

# **Tigase Administration Guide**

**Tigase Team**

---

# **Tigase Administration Guide**

Tigase Team

---

# Table of Contents

1. Scripting support in Tigase .....	1
2. Scripting Introduction - Hello World! .....	2
Loading Script at Run Time .....	2
Executing Script .....	5
Interaction in Scripts .....	7
Automatic Scripts Loading at Startup Time .....	9
3. Tigase Scripting Version 4.4.x Update for Administrators .....	11
4. Tigase and Python .....	15
Installation .....	15
Writing Python Scripts .....	16
5. About Tigase Jabber/XMPP Server .....	18
Security .....	18
Flexibility .....	19
Easy .....	19
6. Quick Start .....	21
7. Installation Using GUI Installer .....	22
Prerequisites .....	22
Download the Installer .....	22
Run the jar File .....	22
Starting the Installation .....	23
JDK Selection .....	24
Installation Type Selection .....	25
Introduction To the Server .....	26
Choice of Base Directory .....	27
Packages selection .....	28
Basic Server Configuration .....	29
Verification of Database Connection and Performing DB Tasks .....	31
Finishing Installation .....	32
Running the Server .....	32
Installation as a Service .....	33
How to Check if the Server is Running .....	34
8. Installing Using Console Installer .....	35
Installation Using the text-mode Installer .....	35
Requirements and Important Notice .....	35
Download the Installer .....	35
Run the jar File .....	35
Installation Steps .....	35
Initial Screen .....	36
JDK Selection .....	36
Actions Selection .....	36
Installer Info .....	37
Server Info .....	37
Server Licence .....	38
Server Location Selection .....	39
Selection of Packs to be Installed .....	39
Installation .....	39
Basic Configuration .....	40
Advanced Configuration .....	41
Database Configuration .....	41
Database Checking and Preparation .....	42
Installation Complete .....	42

Running the Sever .....	42
How to Check if the Server is Running .....	42
9. Manual Installation in Console Mode .....	43
Get the Binary Package .....	43
Unpack the Package .....	43
Prepare Configuration .....	43
Prepare Database .....	45
Start the Server .....	45
Check if it is Working .....	45
10. Tigase Server Binary Updates .....	47
11. Installation Using Web Installer .....	49
Download and Extract .....	49
Start the Server .....	49
Verify Tigase is Running .....	49
Connect to the Web Installer .....	49
Step Through the Installation Process .....	50
12. Tigase Server and Multiple Databases .....	61
13. Basic System Checks .....	63
14. Linux Settings for High Load Systems .....	64
fs.file-max .....	64
net.ipv4.ip_local_port_range .....	64
TCP_keepalive .....	64
/etc/sysctl.conf .....	65
nofile .....	65
su and init script .....	66
15. Generic Documents - Applying to All Tigase Server Versions .....	67
16. Configuration .....	68
17. Tigase XMPP Server Configuration Properties .....	69
18. init.properties .....	73
19. XML Configuration File Description .....	74
20. Old Way - Editing Configuration File Manually .....	76
Admin Accounts .....	76
Hostnames .....	77
Logs .....	78
Each Line Explained: .....	79
21. Startup File for tigase.sh - tigase.conf .....	80
22. Server Certificates .....	81
Documents Describing How To Obtain and Manage Server Certificates .....	81
23. Creating and Loading the Server Certificate in pem Files .....	82
Server Certificate .....	82
Certificate Request .....	82
Certificate From Other Providers .....	82
Self-Signed Certificate .....	83
Installing/Loading Certificate To the Tigase Server .....	83
PEM File .....	83
Tigase Server Configuration .....	84
24. Installing StartCom Certificate in Your Linux System .....	86
Gentoo Linux .....	86
25. Server Certificate Using Keytool and Keystore .....	87
Self Signed Certificate .....	87
Certificate from CA .....	88
26. Tigase and PyMSN-t Transport .....	90
PyMSN-t - /etc/jabber/pymasn-t.xml file .....	90
PyMSN-t - run command .....	91

PyMSN-t - expected output .....	91
Tigase - etc/tigase.conf file .....	92
Tigase - run command .....	92
Tigase - expected output .....	93
27. Two or More SessionManagers .....	94
28. Database Preparation .....	96
29. Prepare the MySQL Database for the Tigase Server .....	97
Basic Setup .....	97
Configuring from MySQL command line tool .....	97
Configuring From the Linux Shell Command Line .....	98
Configuring MySQL for UTF-8 Support .....	99
Other MySQL Setting Worth Considering .....	100
30. Hashed User Passwords in Database .....	101
Shortcut .....	101
Full Route .....	101
Create a Test User Account .....	102
Test User Authentication .....	102
Password Encoding Check .....	102
Password Encoding Change .....	103
31. Prepare the Derby Database for the Tigase Server .....	104
Basic Setup .....	104
General Approach .....	104
32. Prepare the MS SQL Server Database for the Tigase Server .....	105
Basic Setup .....	105
Preparing SQL Server Instance .....	105
Configuring using SQL Server Management Studio .....	105
Using Wizards .....	105
Using Queries .....	105
Import Schema .....	106
Configuring from command line tool .....	106
Tigase configuration - init.properties .....	107
JDBC: jTDS vs MS JDBC driver .....	107
33. Prepare the PostgreSQL Database for the Tigase Server .....	108
Basic Setup .....	108
Configuring from PostgreSQL Command Line Tool .....	108
Configuring From the Linux Shell Command Line .....	108
34. Tigase Database Minor but Useful Schema Change in Version 5.1.0 .....	110
35. Tigase Load Balancing .....	111
Available Implementations .....	111
Configuration Options .....	111
36. StanzaSender .....	113
How it Works .....	113
FileTask .....	113
JDBCTask .....	114
Configuration .....	114
37. Debugging Tigase .....	116
The easy way - init.properties file .....	116
The more difficult but more powerful - tigase.xml file (only applicable to versions before 5.0.0) .....	117
38. Importing User Data .....	118
39. Drupal Authentication Added .....	120
40. Tigase Services .....	121
41. Add and Manage Domain .....	122
Adding a New Domain .....	122

Adding a New User .....	125
What Else? .....	127
42. Presence Forwarding .....	128
43. Register Own XMPP Domain .....	131
44. Server Monitoring .....	133
Setting Up Remote Monitoring in the Server .....	133
What You Need. ....	133
Activation .....	134
Security .....	134
SNMP .....	134
Retrieving statistics from the server .....	135
Retrieving statistics using XMPP .....	135
Retrieving statistics using JMX .....	139
45. Tips and Tricks .....	143
Tigase Tip: Checking the Runtime Environment .....	143
Tigase Tip: Checking Cluster Connections .....	144
Established connections .....	144
Cluster nodes connected (using XMPP) .....	144
46. Tigase Server version 5.x .....	147
47. Tigase 5.1 Database Schema Upgrade .....	148
48. Derby Database Schema Upgrade for Tigase 5.1 .....	149
49. MySQL Database Schema Upgrade for Tigase 5.1 .....	150
50. PostgreSQL Database Schema Upgrade for Tigase 5.1 .....	151
51. Server Configuration 5.x .....	152
52. Configuration Changes in the Tigase Server 5.x .....	153
Reverting To the Old Behaviour .....	153
Default Behaviour .....	153
Storing Configuration in SQL Database .....	154
Going Further .....	155
Message Router Implementation is Configurable Too .....	155
53. Advanced Message Processing - AMP XEP-0079 .....	157
54. Configuration SASL EXTERNAL .....	159
55. Server to Server Protocol Settings .....	160
Number of Concurrent Connections .....	160
Connection Throughput .....	161
Maximum Packet Waiting Time and Connection Inactivity Time .....	161
Custom Plugin Selecting s2s Connection .....	161
56. Tigase Server Version 4.x .....	162
57. MySQL Database Schema Upgrade for Tigase 4.0 .....	163
58. Virtual Components for the Cluster Mode .....	165
59. External Component Configuration .....	166
60. Basic Configuration Options (External Component) .....	167
Simple Case .....	167
More External Components/Domains .....	167
More TCP/IP Ports .....	168
Outgoing Connections .....	168
Specifying Protocol .....	169
Load Balancer Plugin .....	169
61. Tigase as External Component .....	171
A Simple Case - MUC as an External Component .....	171
More Components .....	172
62. Load Balancing External Components in Cluster Mode .....	173
Load Balancing External Component .....	173
External Component and Cluster .....	175

63. Packet Filtering .....	178
Domain Based Packet Filtering .....	178
Introduction .....	178
Configuration .....	179
Administration, Rules Management .....	179
Implementation .....	179
64. Virtual Hosts in the Tigase Server .....	180
65. Specification for ad-hoc Commands Used to Manage Virtual Domains .....	183
Reloading the Domains List from the Database .....	183
Adding a New Domain or Updating Existing One .....	184
Removing a Virtual Domain From the Server .....	185
66. Configuration .....	187
67. Configuring the Tigase Server to Load a Component .....	188
68. Custom Authentication Connectors .....	189
69. Tigase Custom Auth Connector .....	191
70. LDAP Authentication Connector .....	194
71. Tigase Auth Connector .....	195
72. Tigase Server Version 3.x .....	196
73. Connecting the Tigase Server to MySQL Database .....	197
74. Integrating Tigase Server with Drupal .....	199
Connecting to Drupal Database .....	199
75. Integrating Tigase Server With LibreSource .....	201
Short Introduction. ....	201
How to set Tigase up. ....	201
Migration From old Tigase Installation to LibreSource .....	203
Data Migration .....	203
76. Windows Installation .....	205
MySQL Database Installation .....	205
77. Configuration .....	213
78. Configuration Wizards .....	214
79. Tigase Server version 2.x .....	217
80. Installation .....	218
81. MySQL Database Use .....	220
MySQL Database Preparation .....	220
Server Configuration .....	220
User Data Import .....	221
82. PostgreSQL Database Use .....	222
PostgreSQL Database Preparation .....	222
Server Configuration .....	222
83. Configuration .....	223
84. Short Configuration Guide .....	224
Admin Accounts .....	224
Hostnames .....	225
85. Command Line Admin Tools .....	227
86. Configuration Management Tool .....	228
87. Best Practices for Connecting From Web Browser to Tigase XMPP Server .....	230
BOSH .....	230
Tip #1 - BOSH in Cluster Mode Without Load Balancer .....	230
WebSocket .....	230
Tip #1 - Encrypted WebSocket Connection .....	231
Tip #2 - Encrypted WebSocket Connection - Dealing With Multiple VHosts .....	231

---

## List of Figures

44.1. Access service discovery .....	136
44.2. Access statistics component in service discovery .....	137
44.3. Server statistics .....	138
45.1. Access service discovery .....	145
45.2. List of cluster nodes .....	146

---

# Chapter 1. Scripting support in Tigase

Artur Hefczyc <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: <http://tigase.net> :Date: 2010-01-06 20:22

The Tigase server supports scripting languages from the version 4.3.1. These pages describe this feature in details how to create new scripts, upload them to the server and execute them. The guide also contains API description with code examples.

There is one thing I have to highlight before you start digging into the scripting stuff. The Tigase server is known for it very low memory consumption. It successfully runs with less then 10MB of RAM memory. However, if you added scripting support for any non-standard (default) language to the Tigase server it significantly increases memory requirements for the installation. You cannot expect the Tigase server to run on 10MB RAM system if you enabled Python, Scala or any other non-standard language.

---

# Chapter 2. Scripting Introduction - Hello World!

Artur Hefczyc <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2010-01-06 20:18



This document is the first in series describing scripting support in the Tigase server. It shows how to load, install, update and call a script. It contains also an introduction to the scripting API with the first "Hello world!" like example.

Since the Tigase version 4.3.1 the server supports scripting for administrator commands.

In theory many different languages can be used to write scripts and the only requirement is that the support JSR-223 [<http://www.jcp.org/en/jsr/detail?id=223>] for the language. More details can be found on the Java scripting project site [<https://scripting.dev.java.net/>].

In practice some languages are better supported than others. At the moment the most recommended is Groovy [<http://groovy.codehaus.org/>], although following languages are also confirmed to be working: Scala [<http://www.scala-lang.org/>], Python [<http://www.python.org/>] and Ruby [<http://www.ruby-lang.org/en/>]. The Tigase SVN [<http://projects.tigase.org/server/trunk/src/main>] contains a few examples for these languages.

Please note, the default Tigase installation contains only libraries for Groovy. Adding support for a different language is as simple as copying a few JAR files to the Tigase **libs/** directory.

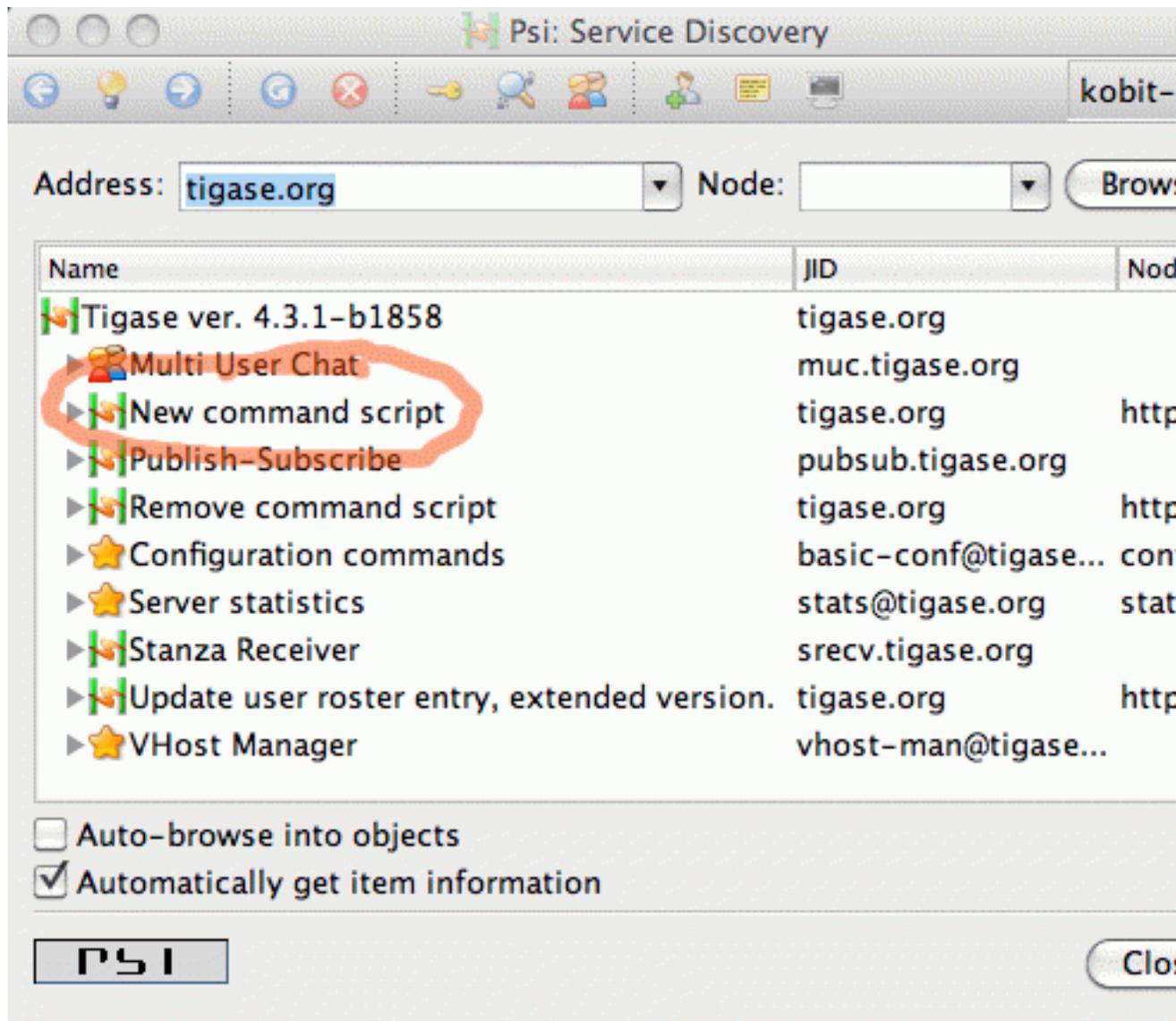
All the examples presented in this guide are also available as ready to use scripts in the Tigase SVN repository in directory: `src/main/groovy/tigase/admin` [<http://projects.tigase.org/server/trunk/src/main/groovy/tigase/admin>].

The scripting utilises only standard XMPP extensions and is by no means specific to any particular solution. I use and prefer Psi client. The whole guide and all the screenshots are created using Psi client. You can, however, use any other client which supports these extensions as well. As the whole thing is based on the service discovery and ad-hoc commands you need an XMPP client with a good support for both features.

To follow the guide and run all the examples you need installed Tigase server in version at least 4.3.1 and you have to connect to the server as administrator.

## Loading Script at Run Time

All the scripting stuff is as usually based on the service discovery and ad-hoc commands in the Tigase server.

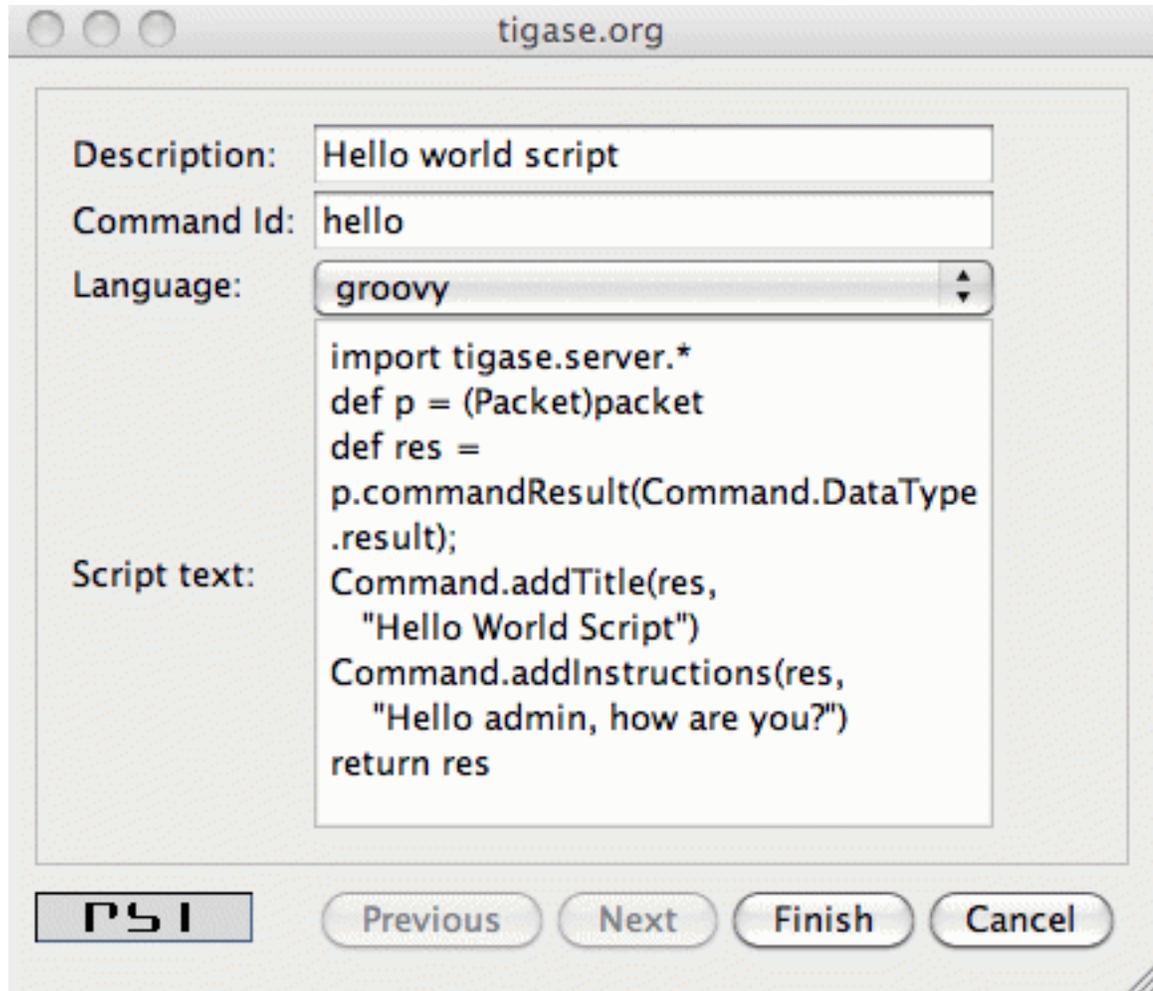


The first thing to do, therefore, is to browse service discovery on the running server. The result you receive depends on your installation and installed components.

The most interesting for us right now are all items with "<http://jabber.org/protocol/admin>" in their node part. You may have a few scripts loaded already but there are two commands used for scripting management. Their names are descriptive enough I hope: "**New command script**" and "**Remove command script**".

The first is for adding a new script or updating existing and the second is for removing script from the server.

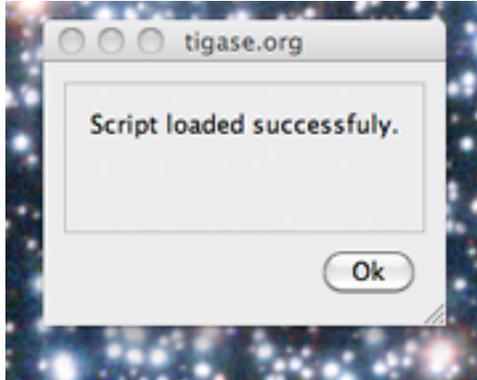
To add a new script you have just to execute "**New command script**". In Psi this is done by double clicking on the element in service discovery list.



The screenshot above shows a couple of options to set for the loaded script:

- **Description** - is what shows as the script name in the service discovery window. There are no special restrictions on what to put there.
- **Command id** - is a unique ID of the script (admin command). This is what shows after the "http://jabber.org/protocol/admin" in node part. This needs to be unique or existing script is overwritten.
- **Language** - a drop down list of all supported scripting languages for your installation. The Tigase automatically detects all libraries for scripting languages and lists them here. So all you need is to select correct language for your script.
- **Script text** - is just your script content.

When your script is ready and all fields are correctly set, simply press "**Finish**" button and you should receive a message confirming that the script has been loaded successfully.



In this guide we are creating a simple script of type "Hello world". The script is written in Groovy. What it does is displaying a window (ad-hoc command result) with a message: *"Hello admin, how are you?"*.

It uses a basic scripting API which is described line by line below:

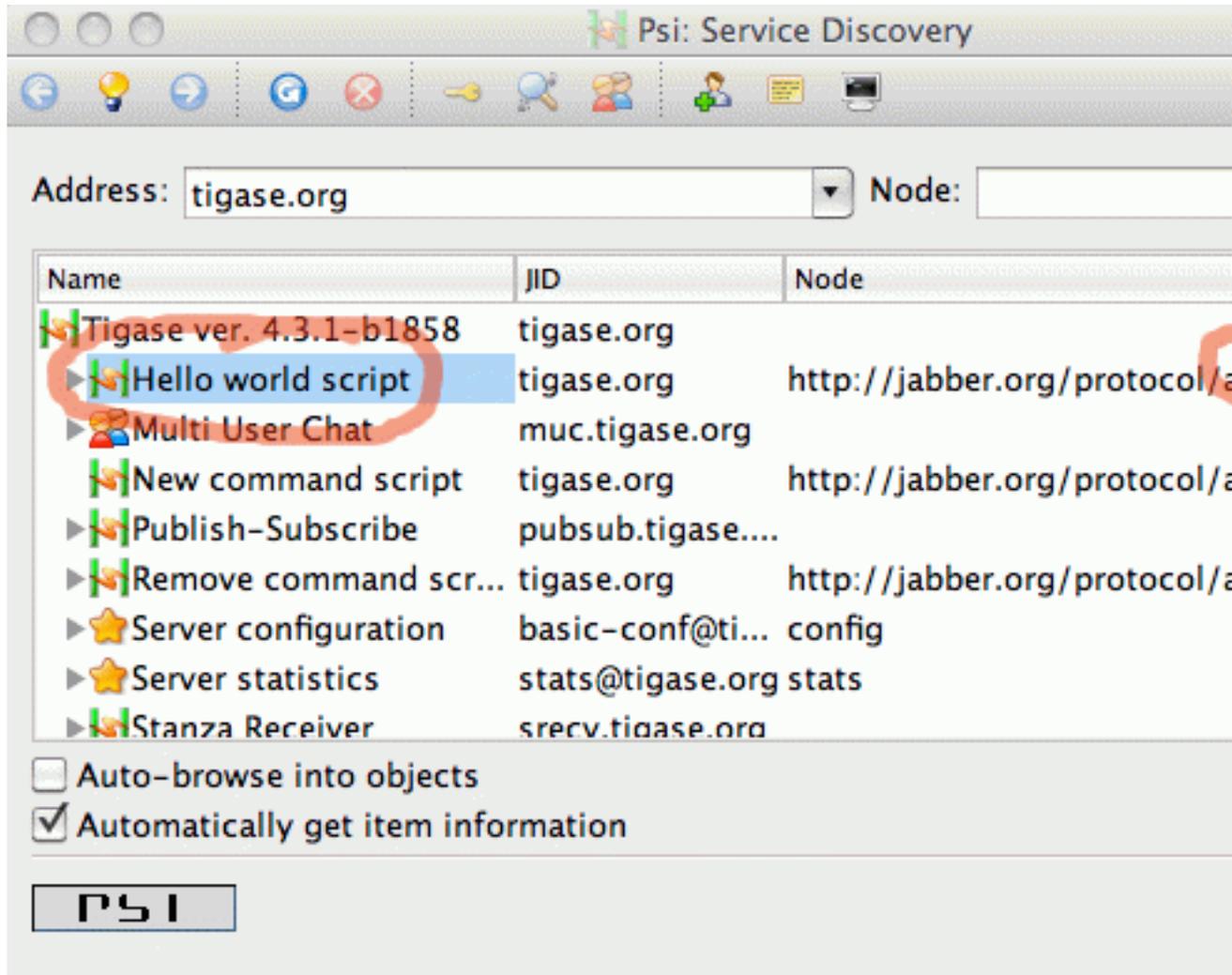
1. It imports basic Tigase classes.
2. Set's a local variable '\p' which points to a '\packet' variable with data received from the client.
3. Creates a '\res' variable which is response sent back to the client (administrator). The response to the client is of type '\result'. Other possible types will be introduced later.
4. We operate on ad-hoc commands here so the script uses Tigase utility class to set/retrieve command parameters. It sets the window title and a simple message displayed to the user (administrator).
5. The last line returns new packet as a script execution result.

The first, very simple version looks like this:

```
import tigase.server.*
def p = (Packet)packet
def res = p.commandResult(Command.DataType.result)
Command.addTitle(res, -"Hello World Script")
Command.addInstructions(res, -"Hello admin, how are you?")
return res
```

## Executing Script

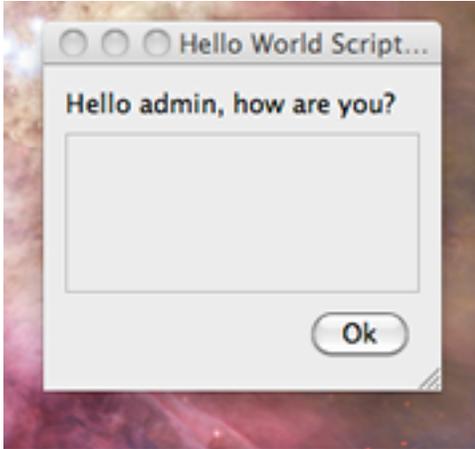
Once the script is successfully loaded you have to reload/refresh the service discovery window which now should display one more element on the list.



As you can see script name is set to what you have entered as "Description" in script loading window - "Hello world script". The command node is set to: "http://jabber.org/protocol/admin#hello" if "hello" is what is set as the script ID.

To execute the script you just have to double click on the script name (or click execute command if you use any other client).

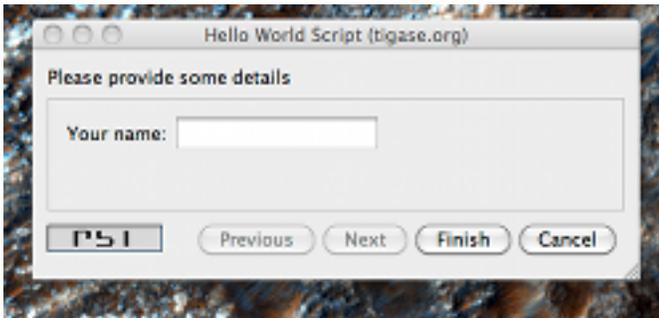
As a result you should see a simple window similar to the screenshot below displaying our message.



## Interaction in Scripts

Displaying just a message is very nice but in most cases not very useful. Normally you need to ask the user for some more data or parameters before you can perform any real processing.

Therefore in most cases the administrator script has to display a new window with input fields asking the user (admin) for some more data. In this document we present very simple examples, just an introduction so let's ask about the administrator name before displaying greetings.



To ask the user for some more information we have to extend example above with some more code:

```
import tigase.server.*

def p = (Packet)packet

def name = Command.getFieldValue(packet, -"name")

if (name == null) {
    def res = p.commandResult(Command.DataType.form);
    Command.addTitle(res, -"Hello World Script")
    Command.addInstructions(res, -"Please provide some details")
    Command.addFieldValue(res, -"name", name -?: -"", -"text-single",
        -"Your name")
    return res
}

def res = p.commandResult(Command.DataType.result)
```

```
Command.addTitle(res, -"Hello World Script")
Command.addInstructions(res, -"Hello ${name}, how are you?")

return res
```

If you compare both scripts you see that they are quite similar. Before displaying greeting, however, the script tries to retrieve data from the 'name' input field. If the name had been provided the greeting is displayed, otherwise the script asks for the user name.



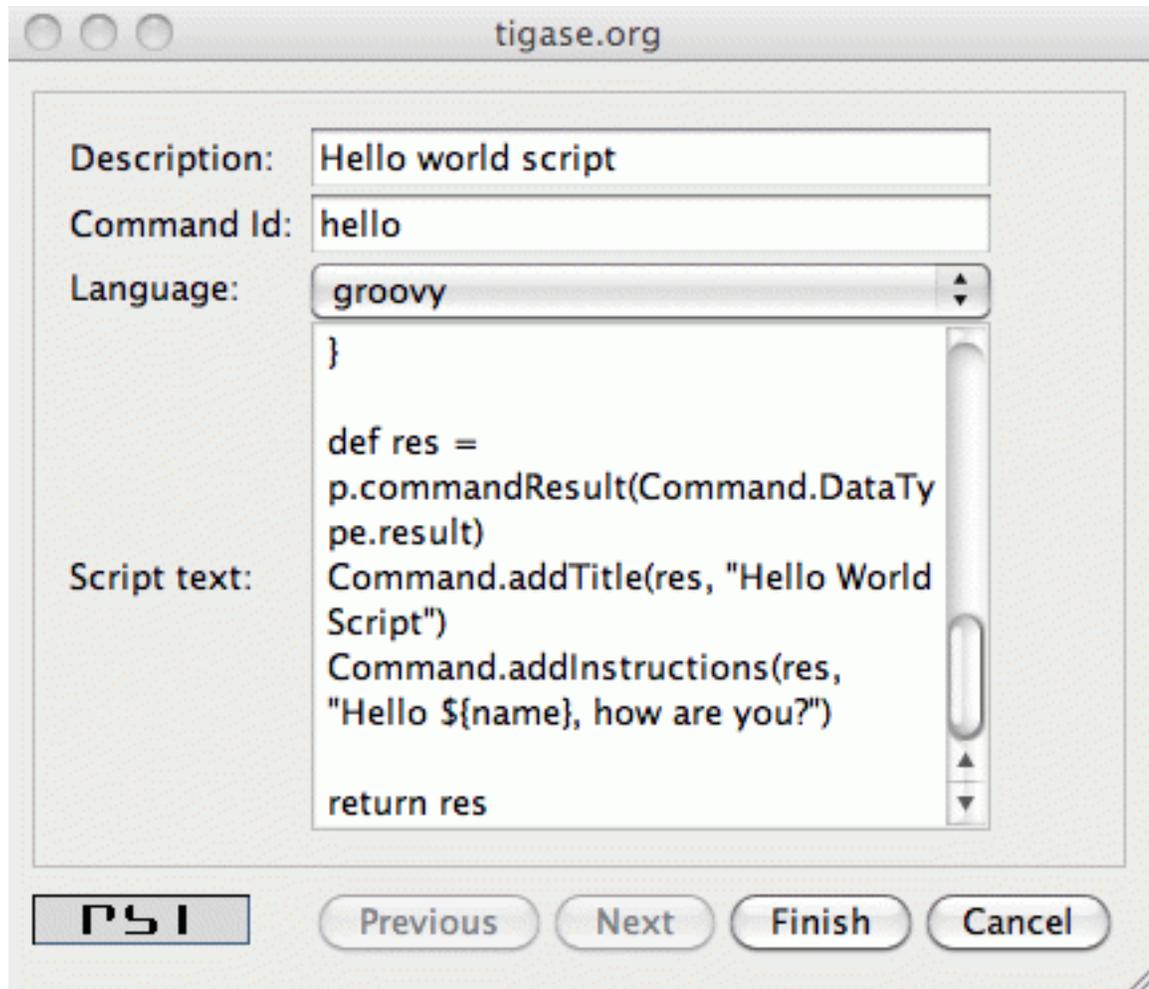
Please note, in this case the packet sent back to the user is of type form instead of result. The practical difference is that the type result displays only **OK** button which is pressed doesn't send any data to the server. The form packet displays more buttons - **Finish** and **Cancel**. Whichever you press some data are sent back to the server.

The script demonstrates use of two new methods from the utility class "Command": `getFieldValue` and `addFieldValue`.

- The first argument to all Command methods is the packet with ad-hoc command.
- The second argument is usually the input field name

These two method parameters are actually enough to read the ad-hoc command data. Methods creating input fields in the ad-hoc command need a few arguments more:

- Next arguments sets a default value displayed to the user. The way how it is set in the example above is specific to Groovy language and is quite useful what will be apparent in later examples.
- After that we have to specify the field type. All field types are defined in the XEP-0004 [<http://xmpp.org/extensions/xep-0004.html#protocol-fieldtypes>].
- The last argument specifies the field label which is displayed to the user.



There are a few other different utility methods in the Command class to set different types of input fields and they will be described in details later on.

To reload the script simply call "New command script" again, enter the script text and make sure you entered exactly the same command ID to replace the old script with the new one.

Or, of course you can enter a new command id to create a new command and make it available on your server.

When the script is loaded on the server, try to execute it. You should get a new dialog window asking for your name as in the screenshot at the beginning of this section. When you entered your name and pressed "Finish" button you see another window and greetings message with your name.

## Automatic Scripts Loading at Startup Time

The last thing described in this guide is how to automatically load your scripts when the Tigase server starts. The ability to load scripts at run time, update them and remove is very useful, especially in emergency cases if something wrong is going on and you want to act without affecting the service.

If you, however have a few dozens scripts you don't want to manually load them every time the server restarts.

The Tigase server automatically loads all scripts at the startup time which are located in the admin scripts directory. Unless you set it differently in the configuration it is: **YourTigaseInstallationDir/scripts/admin/**. All you have to do is to copy all your scripts to this directory and they will be loaded next time the server starts.

But hold on. What about the script parameters: language, description, command id? How are you supposed to set them?

Language is simple. It is detected automatically by the script file extension. So just make sure file extensions are correct and the language is sorted.

The script description and command id needs a little bit more work. You have to include in your script following lines:

```
AS:Description: The command description
AS:CommandId: command-id
AS:Component: comp_name
```

Please note, there must be at least a single space after the "AS:Description:" or "AS:CommandId:" string. Everything rest after that, until the end of the line, is treated as either the script description or command id. Put these in your script file and the loader will detect them and set correctly for your script.

---

# Chapter 3. Tigase Scripting Version 4.4.x Update for Administrators

Artur Hefczyc <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: <http://tigase.net> :Date: 2010-01-06 20:18



Scripting functionality is quite useful in the Tigase server for all sorts of administrator tasks. The possibility to load new scripts or replace old ones at the server runtime opens quite new area for the service maintenance.

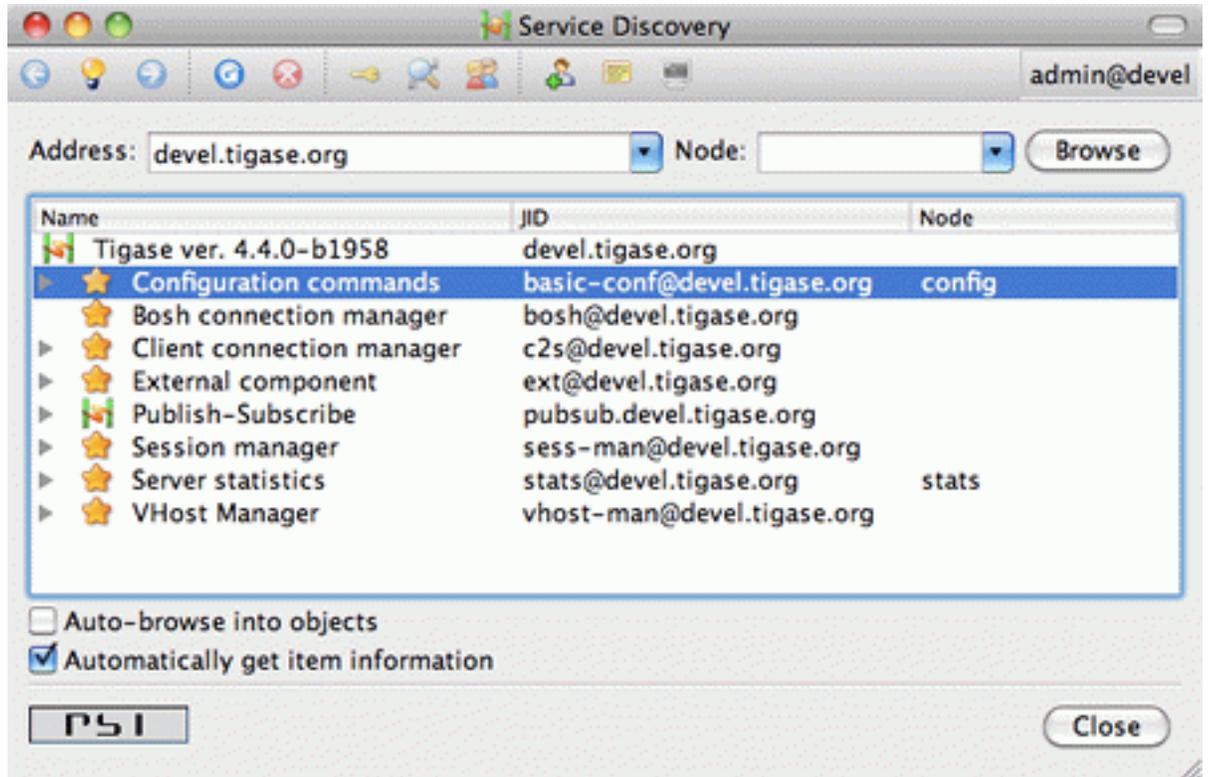
In earlier versions of the Tigase server scripting capabilities was available only in the session manager component while it might be very useful in many other places - connection managers, MUC, PubSub, VHostManager and what even more important in completely new, custom components created for specific needs. It would be quite wasteful to reinvent the wheel every time and implementing scripting capabilities for each component separately.

Therefore the scripting capabilities has been implemented in the core of the Tigase server. It is now part of the API and is automatically available to all components without any additional coding. A detailed developer guide will be published separately.

This document describes changes from the user/administrator perspective because there are some usability changes related to the new implementation.

Please note. The description and screenshots are taken from the Psi client and most likely interface for ad-hoc commands and service discovery on other client looks different. I recommend to do some initial testing and experiments using Psi client and then switch to your preferred application for your day-to-day use.

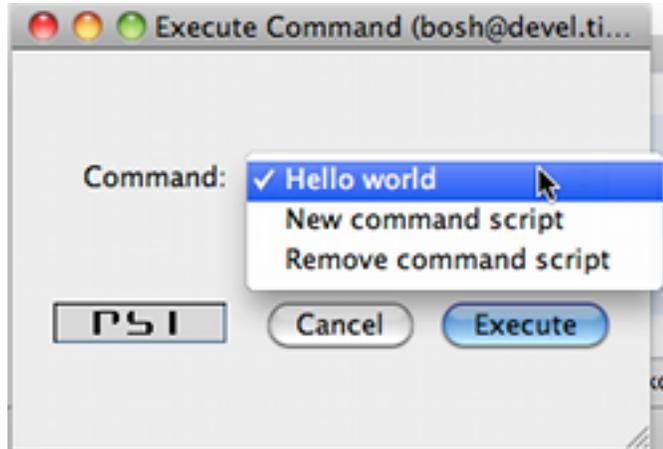
As it always was in the Tigase you can access all the functions via XMPP service discovery on the server. However, as soon as you connect to the server you can see some changes there.



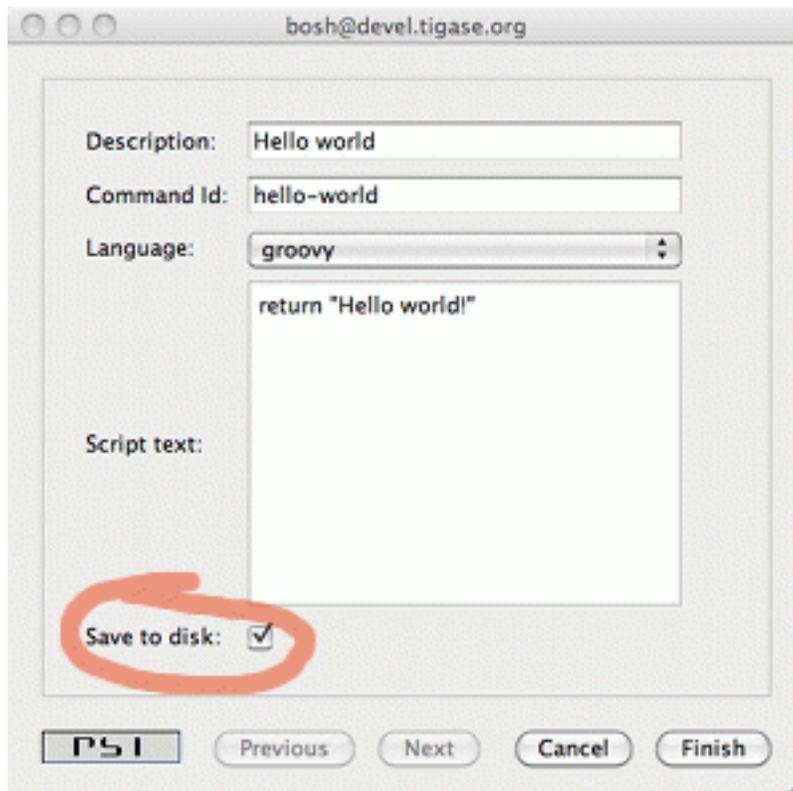
There are no command on the list. They are hidden from the main service discovery list. You can see on the list only the server main components.

This had to be done for many reasons. One of them is, obviously, the cleaner access to the main server stuff. Another, probably more important, is to avoid a long list of commands for different components mixed together. Commands for different components can have the same name/description and they can even do similar things but they are executed on a different server component. To avoid any confusion and minimise opportunities for mistake the commands are now closely tight to their components. To access a list of commands for a particular component you have to double click on the component name on the list or click 'Execute command' icon on top of the window when your component is selected.

A new window should show up with drop-down list of available commands. All the commands are related to the selected component and are executed kind of "inside the component environment". You can of course add new command or delete existing one and of course execute any of the commands showing on the list.



As a reminder, in the window title you can see the component ID and you should check it before running any command to make sure you accidentally don't break your system.



There has been also a small change made to the script adding window. As you can see on the screenshot there is one additional option added - "Save to disk". This means that once you submitted the script to the server it is written to the hard drive and will be automatically loaded at next startup time.

This option is enabled by default as this seems to be a logical choice that the administrator wants to save his new script for later reuse. This, however requires proper configuration of the server and give writing permission to the directory where all scripts are stored. Otherwise the server won't be able to write script files on the hard drive.

As in previous version only users with administrator permissions can execute commands and access all the critical elements on the server. There has been, however, another change made, long time requested by users. In the new version all the administrator specific elements are hidden for the rest of users.

Server components don't show up on the service discovery, the user can't see administrator commands nor he can execute them. This hasn't been implemented to improve the server security but to reduce confusion for general users who would otherwise see a lot of stuff which can't be used by them anyway.

---

# Chapter 4. Tigase and Python

Artur Hefczyc <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: <http://tigase.net> :Date: 2010-01-06 20:22

Looking in the last poll [<http://www.tigase.org/content/what-scripting-language-you-would-use-admin-commands-tigase>] it is clear that the most people are mainly interested in Python support in the Tigase server. As I mentioned in one of previous articles [<http://www.tigase.org/content/scripting-introduction-hello-world>], Tigase virtually supports any scripting language as long as there is JSR-223 [<http://www.jcp.org/en/jsr/detail?id=223>] support for that language.

This article describes how to get Python working as a scripting language for ad-hoc commands in the Tigase server. The first part is installation, and the second shows a few code examples with explanation of the differences between Python usage and some other languages.

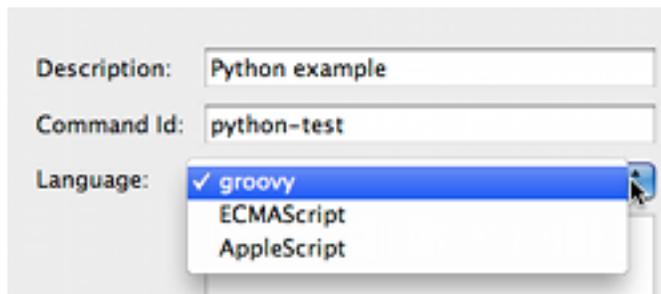
*Please note, I am not a Python developer, and by no means this is Python development guide. In fact I know very little Python. All the code examples are used only to present the API available and there are certainly better ways to do it in the proper Python style. If you have any suggestions or have a better code examples I am happy to include them in the guide.*

## Installation

In short, the installation is extremely simple. Just copy the file attached to this article to your Tigase installation, to the `libs/` directory. Restart the server and... here it is. You can write and execute Python scripts inside the Tigase server right away.

And here it is, the full description.

In theory the Tigase offers scripting support defined in JSR-223 [<http://www.jcp.org/en/jsr/detail?id=223>]. You can use any language for which there is such support for JVM. This includes also stand-alone python implementations and the JSR-223 plugins acts just as a bridge. This, however, does not make much sense as you are not able to interact with JVM code (Tigase API). Therefore you need a language which is executed within JVM and can easily exchange data between the main application (Tigase server) and the script.



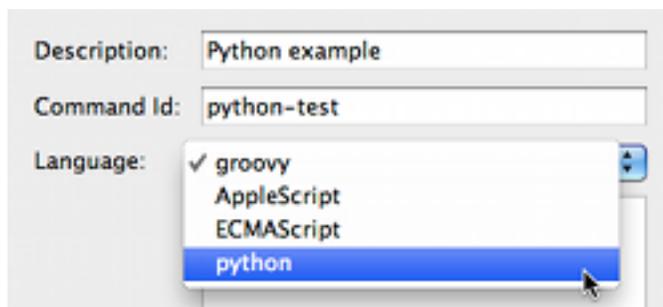
The best way to go is to use Jython implementation. It works very well within JVM and what is very important it perfectly integrates with the Tigase server. I have tested the Tigase server with **Jython-2.2.1** and can confirm it works fine. I, however, recommend version **Jython-2.5.1** and all the examples are executed with this version installed. Please note, *Jython-2.5.0* does not work at all. Both supported versions can be downloaded from the Jython website [<http://wiki.python.org/jython/DownloadInstructions>].

**Version 2.5.1** is a bit simpler to install. When you download and run the Jython installer, you have to find `jython.jar` file in the directory where you installed Jython. Just copy the file to the Tigase's **libs/**

directory and all is ready to go. Please note, this is the same file as the one attached to this article for your convenience.

**Version 2.2.1** needs a little bit more work. The first part is the same. It is not, however enough to copy the `jython.jar` file. One more file is necessary for the Jython to work with the Tigase server. You have to install JSR-223 engine separately. JSR-223 engines binary file can be downloaded from Java scripting project website [<https://scripting.dev.java.net/>]. The binary file has to be unpacked and `jython-engine.jar` file needs to be copied to the Tigase `libs/` directory.

The best way to check if the Jython is installed correctly and support for Python is enabled is by trying to submit a new script to the Tigase server. Browser the server service discovery, select "*Session manager*" component and run "*Execute command*" function. A new window should show with a list of all available ad-hoc commands. Select "*New command script*" item and click "*Execute*". Ad-hoc command dialog windows should show up. One of the field is "*Language*" with pull down list of available scripting languages. If "*python*" is on the list it means everything is ok and support for Python is enabled.



## Writing Python Scripts

Python scripts work, generally in a similar way to Groovy or other languages scripts, except one significant difference. You cannot call "*return*" from the script itself. Hence you cannot simply pass script results by calling "*return*" statement directly from the script.

To overcome the problem Tigase offers another way to pass script execution results. It checks the value of a special variables on the script completion: "*result*" and "*packet*". By assigning value to one of these variables the Python (or any other language) can pass execution results back to the Tigase server.

- "*result*" allows to return simple text (or characters String) from the script.
- "*packet*" allows to return Packet instance which is send back to the user.

The simplest possible Python script may look like this one:

```
result = "Hello world!"
```

For instructions how to load and execute the script, please refer to [introductoryArticle](#), [introductory article](#) for scripting in the Tigase server. There were some minor changes in later the Tigase 4.4.0 and later version, so please have a look at the [newElements](#), [article](#) describing new elements as well.

A bit more advanced script asks the user for providing required parameters for the actual script execution:

```
from java.lang import *
from tigase.server import *

num1 = Command.getFieldValue(packet, -"num1")
```

```
num2 = Command.getFieldValue(packet, -"num2")

if num1 is None or num2 is None:
    res = Iq.commandResultForm(packet)
    Command.addTextField(res, -"Note", -"This is a Python script!")
    Command.addFieldValue(res, -"num1", -" ")
    Command.addFieldValue(res, -"num2", -" ")
    packet = res
else:
    result = num1 + num2
```

Except this rather minor difference, the rest part of scripting in Python for the Tigase administrator commands is the same as for all other languages. As all languages can return execution results via these special variables, it could be argued there is no difference at all.

In next articles, I am going to present the Tigase server API available for scripting framework. My main language is Groovy as it offers the best integration with JVM and Tigase API, however I will try to include Python example code as well.

\_I hope this article encourages you to try the scripting support in the Tigase server. If you have any suggestions or questions please do not hesitate to send me your comments. I have also created a new [tigase scripts](http://www.tigase.org/forums/tigase-scripts) [http://www.tigase.org/forums/tigase-scripts] forum on the website. If you have an interesting script to share or want to discuss some aspects of this functionality do not hesitate to add your post.

- [jython-2.5.1.jar](#) [files/jython-2.5.1.jar] 6.44 MB

---

# Chapter 5. About Tigase Jabber/XMPP Server

Artur Hefczyc <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2010-04-06 16:18

**Tigase Jabber/XMPP Server** is **Open Source and Free (AGPLv3)** Java based server. The goals behind the design and implementation of the server are:

1. Make the server robust and reliable.
2. Make the server secure communication platform.
3. Make flexible server which can be applied to different use cases.
4. Make extensible server which takes full advantage of XMPP protocol extensibility.
5. Make the server easy to setup and maintain.

## **Robust and reliable.**

By robust and reliable server I mean the server which can handle many concurrent requests/connections and can run for a long time. By many concurrent requests/connections I mean as many as it is needed. The server is designed and implemented to handle millions of simultaneous connections.

It is not enough however to design and implement high load server and hope it will run well. The main focus in the project is put in tests. Tests are taken so seriously that dedicated testing framework has been implemented. All server functions are considered as implemented only when they pass testing cycle. The testing cycle consists of 3 fundamental tests:

1. **Functional tests** - checking whether the function works at all.
2. **Performance tests** - checking whether the function performs well enough.
3. **Stability tests** - checking whether the function behaves well in long term run. It must handle hundreds of requests a second in several hours server run.

## **Security**

There are a few elements of the security related to Jabber/XMPP server: secure data transmissions which is met by the implementation of **SSL** or **TLS** protocol, secure user authorization which is met by the implementation of **DIGEST** or **SASL** user authorization and secure deployment which is met by component architecture.

**Secure deployment** is a software installation which doesn't impact network security. Companies usually have their networks divided into 2 parts: **DMZ** which is partially open to outside world and **Private network** which is closed to outside world.

If the Jabber/XMPP server have to provide effective way of communication between company employees regardless they are in company office or outside (perhaps at customer site) it needs to accept connections from outside world. So the natural place for server deployment is a **DMZ** part. There is an issue however with such installation. Each company has normally established network users base and integrated autho-

authorization mechanisms. So it would be very good if Jabber server could use this mechanisms for users authorizations as well. Usually, however, authorization information is not available in **DMZ** and it shouldn't be.

**Tigase server** offers solution for such case. With it's component structure it can be easily deployed on any number machines and from the user point of view it is seen as a one logical Jabber server. So in our case we can install Session Manager module in **private** network part and Client Connection Manager with Server Connection Manager in **DMZ**.

Session Manager connects to **DMZ** and receives all packets from users. Thus it can securely realize users authorization based on company authorization mechanisms.

## Flexibility

There are many different Jabber/XMPP server use cases. The most obvious are:

- Used as bussiness communication platform in small and medium companies server is not under heavy load. Instead for such deployments security is a key feature.
- For huge community websites or internet portals server is on the other side usually under very heavy load and have to support thousands or millions of simultaneous connections and for such deployment we talk about different kind of security as the service is open to public anyway.
- For very small community deployments or for small home networks the key factor is ease to deploy and maintain.

Architecture based on components allows you to run selected modules on separate machines so the server can be easily applied to all scenarios.

For simple installation server generates config file which can be used almost stright away with very few modifications or sometimes even no config editing is required. For complex deployments though you can tweak configuration to your needs and setup Jabber server on as many physical machines as you need.

The world changes all the time so people needs change as well. Jabber/XMPP protocol has been designed to be extensible to make it easy adding new features and apply it to different needs. As a result Jabber is a very effective platform not only for sending messages to mates. It can also be extended for sending instant notifications about events, it can be useful platform for on-line customer service, voice communication and all other cases where sending information instantly to other people is needed.

**Tigase server** has been designed to be extensible as well. Of course modular architecture makes it extensible as you can easily replace component which doesn't fullfill your requirements with another one better fitting your needs. But this is not all. Another factor of extensibility is how easy is to replace component or add new extensions. The great focus has been put in server design API to make it easy for other software developers to create extensions and implement new features.

## Easy

Complex computer networks consisting of many servers with different services are hard to maintain. There is no other way than employing professional staff and looking after the network.

Not all networks are so complex however. Most small companies have just a few servers for their needs with services like e-mail, HTTP server with company website and that's it. They might want to add Jabber server to the collection of their services and don't want to dedicate much resources on setting it up and later maintenance. For such users default configuration is pretty much what they need. If operating system on the server is well configured then Tigase should pickup correct hostname.

Tigase server is designed and implemented to allow dynamic reconfiguration at runtime so there is no need for restarting server each time you want to change a configuration settings.

There are also interfaces and handlers available to make it easy to implement web user interface for server monitoring and configuring. Such user interface will included in one of future releases.

---

# Chapter 6. Quick Start

Artur Hefczyc <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: <http://tigase.net> :Date: 2010-04-06 21:18

This is a set of documents allowing you to quickly start with our software. Every document provides an introduction to a single topic allowing you to start using/developing or just working on the subject. Please have a look at the documents list below to find a topic you are looking for. If you don't find a document for the topic you need please let us know [<http://www.tigase.net/contact>].

- [guiInstaller](#), [Installation using GUI installer](#)
- [consoleInstaller](#), [Installing using console installer](#)
- [manualInstaller](#), [Manual installation in console mode](#)
- [TSBupdates](#), [Tigase server binary updates](#)

---

# Chapter 7. Installation Using GUI Installer

Mateusz Fiolka v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: <http://tigase.net>  
:Date: 2010-04-06 21:18

If you don't want to install Tigase using manual method, you can use the GUI installer. It not only copies server files to preferred place, but also assists with configuration of the most important parameters and database setup. Therefore it is the preferred way to install Tigase.

## Prerequisites

Before you can start the GUI installer you will need to have working Java environment. Although installer only requires **JRE** (Java Runtime Environment), server needs the **JDK** (Java Development Kit). Please do note that **currently minimal JDK version Tigase is capable to run on is 1.6**. If you don't have JDK installed it is the right moment to do it. Visit the Java downloads site [<http://java.sun.com/javase/downloads/index.jsp>] From the list of available packages select newest JDK version (if you don't have a specific need to use J2EE then choose a package without it). After configuring JDK you can download the Tigase GUI installer and start the server installation process. It is also important to set the `JAVA_HOME` environment correctly.

## Download the Installer

You can always find the newest Tigase packages in the download section [<https://projects.tigase.org/projects/tigase-server/files>]. When you enter the page, you will be presented a list of files to choose from. You may be sceptic at the beginning as there are lot of choices, but you don't have to. All Tigase binary packages have conventional names, which help to differentiate them easily. They are of form **tigase-server-x.y.z-bv.ext** where 'x', 'y', 'z' and 'v' are version numbers and they change from a release to release. Ext is the file extension which in the case of our GUI installation program is **.jar**. We recommend you to download the latest version (highest version number) of the server as it contains latest functions and improvements

## Run the jar File

On most systems installing JRE or JDK creates a default association which allows to run the **.jar** file by just clicking/ double clicking on it. However if nothing happens when you do it there is a way to do it manually. Perform the steps in the following order:

1. If you are on **Windows** system you can use the command prompt to run the installer directly using the java command.
  - a. Click on the **Start** menu and choose the **Run action** (You can also use the **Win+R** shortcut).
  - b. You will be presented with a dialog box where you can enter a command. Type "**cmd**" (or "**command**" in the case of windows version older then 2000) and submit the window. If you are on a **Linux** system, you can use a terminal. It should be easily discoverable as it is a standard tool on this platform. Find and run it.
2. Command prompt / terminal will appear. You will be able to check a whether your **Java** environment is working. To do it type the

```
java --version
```

command and press Enter. If the message says that the command is not recognised then your **Java** installation may be corrupt or not configured properly. For correctly setting up **JRE/JDK** including setting the **JAVA\_HOME** environmental variable please check documentation provided on the JDK download site. Also when the command succeeds please check if the printed version number fulfils Tigase requirements. When many versions of **JDK/JRE** are installed on one machine **java** command will need to be invoked with the full path it is placed on.

3. When you have no doubt that you can run the correct java launcher, you may start the installer i.e. for the file **tigase-server-4.1.0-b1315.jar** downloaded to the **c:\download** directory type the following command:

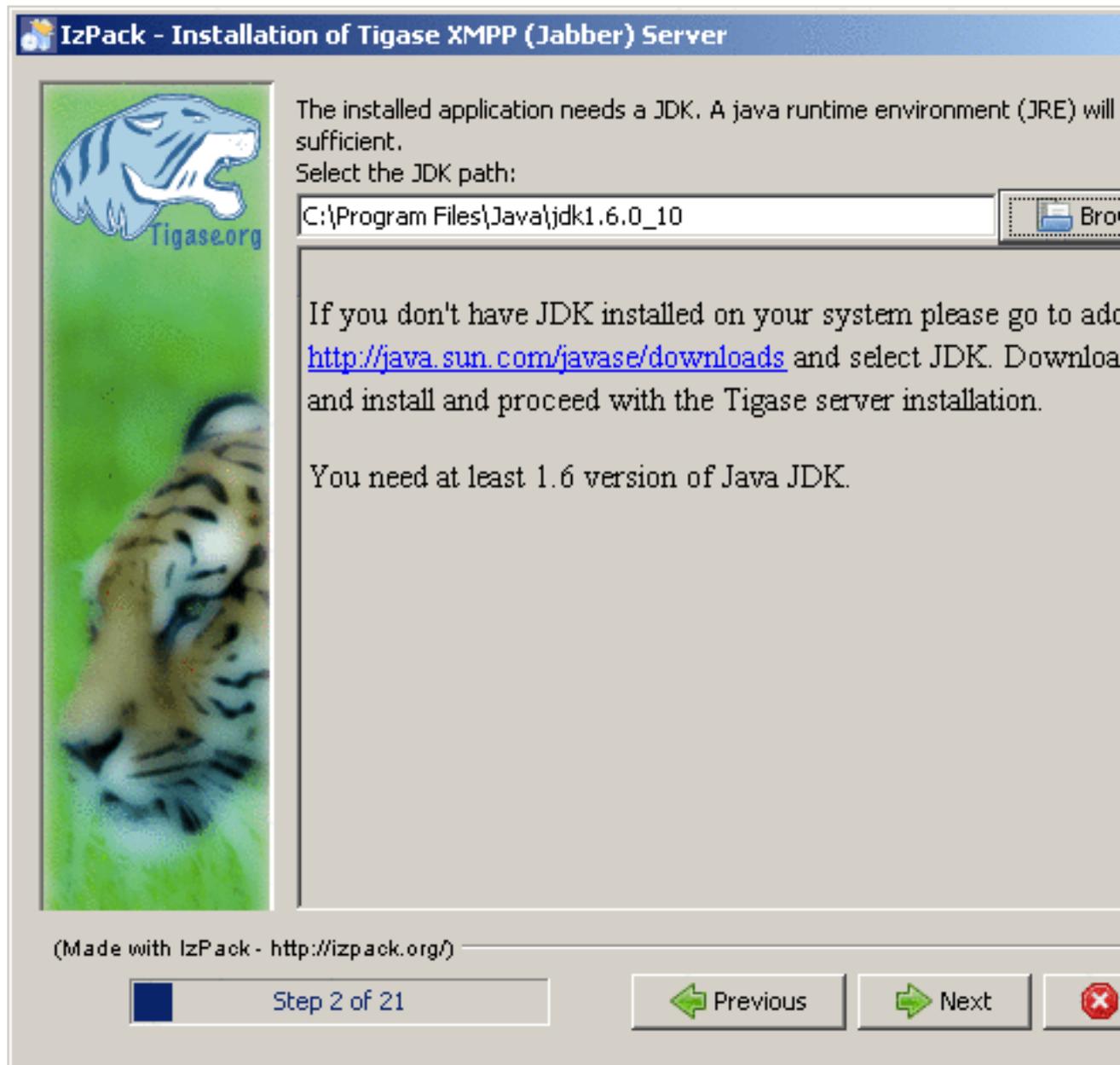
```
java --jar c:\download\tigase-server-4.1.0-b1315.jar
```

This command should start the installer or print an error message explaining what is the cause of problem.

## Starting the Installation

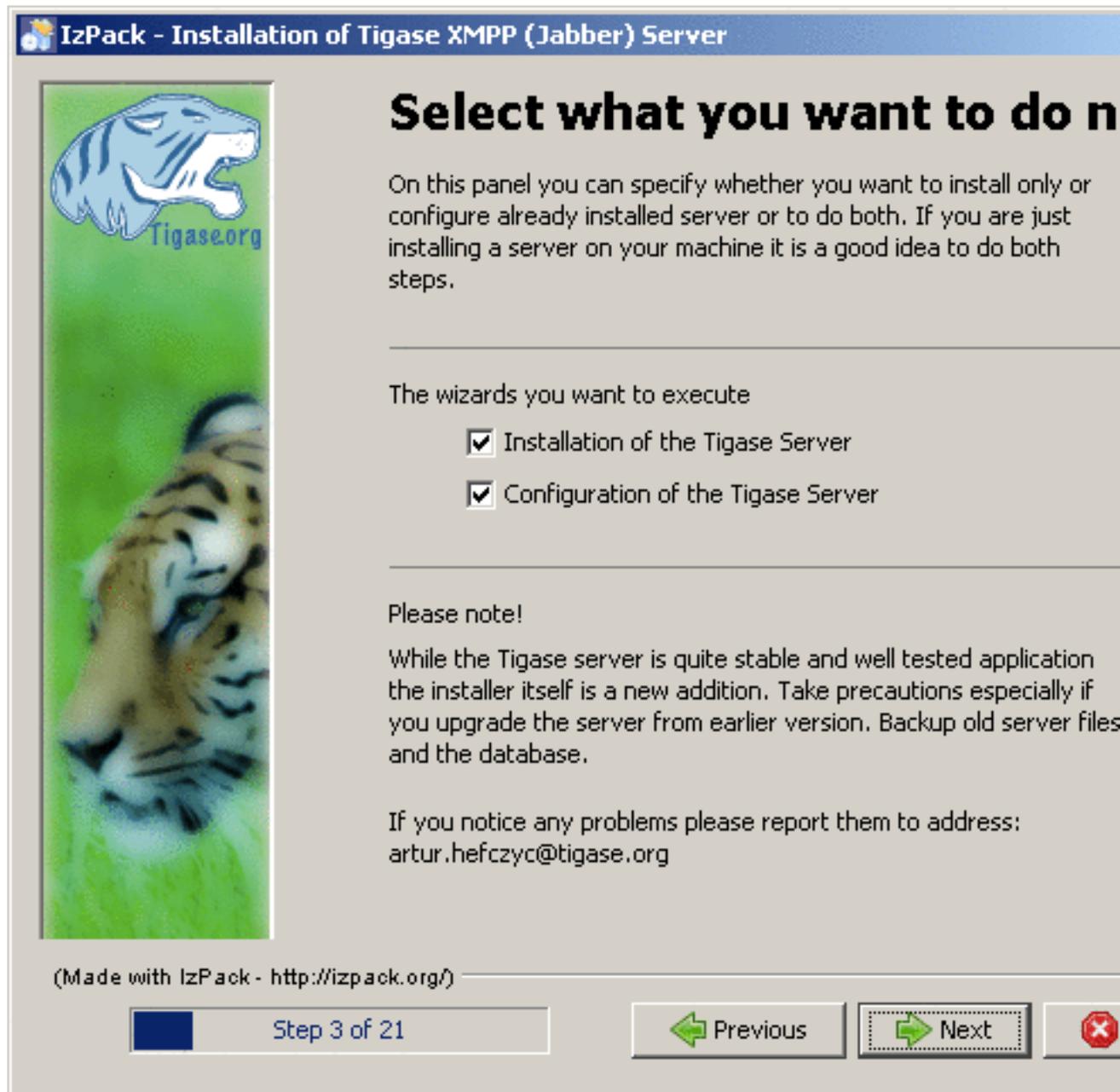
Please note that this tutorial does cover only basic installation mode. Some screens have been omitted because they contain advanced options which are not shown in simple installation mode. Other such as progress of copying files and summary info are on the other hand self explanatory and will also not be described.

## JDK Selection



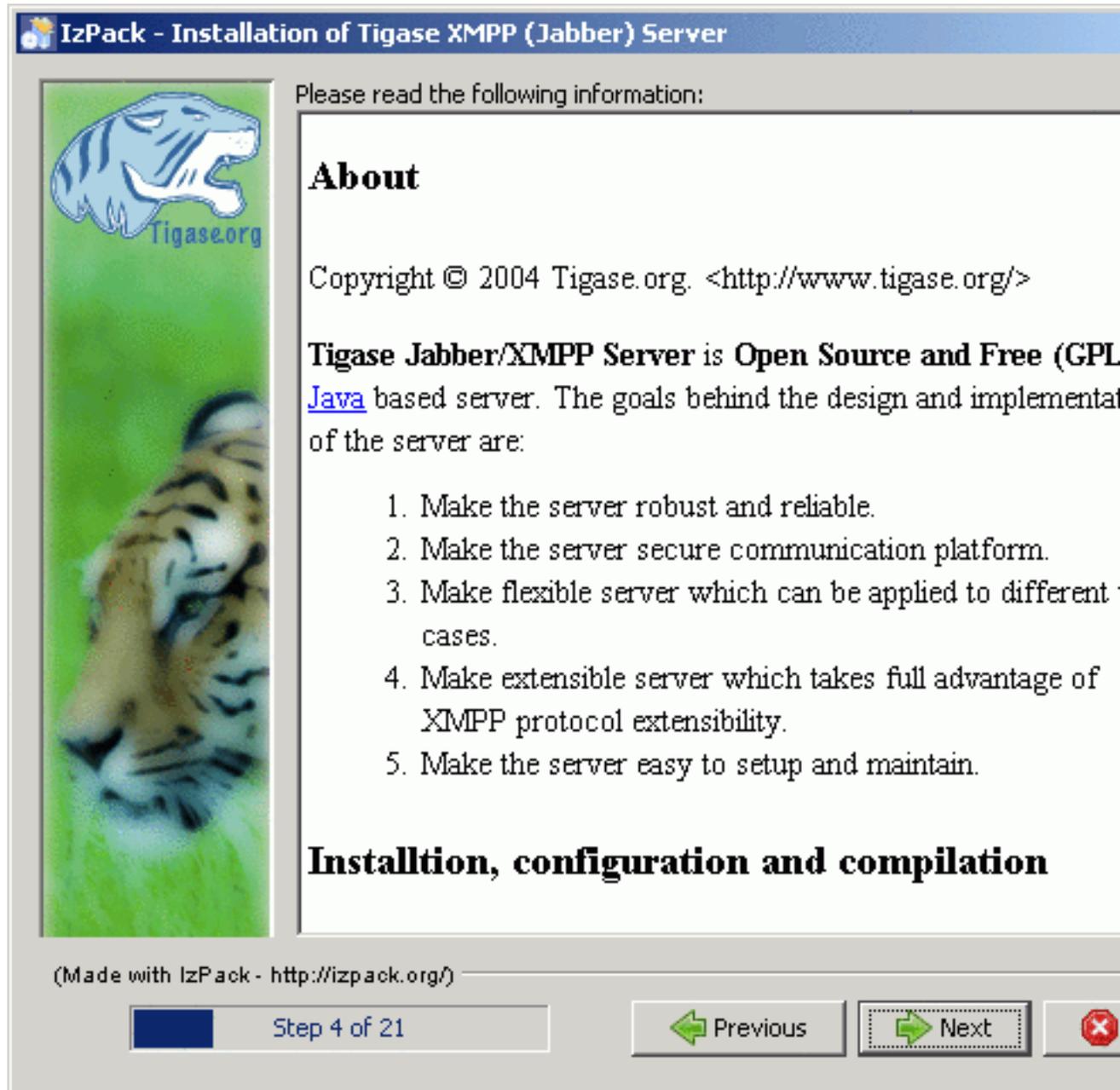
This screen is only shown when JDK has not been selected automatically. When your JAVA\_HOME path is properly set, it will be auto-detected saving you some configuration time. If you are reading this step and still don't have JDK installed, then go back to the `////<<prerequisites,////prerequisites` section where you can find some info on how to prepare your system for Tigase installation. Sometimes your system will be configured in a way that prevents detection of JDK path. This often happens when you install JRE after installing JDK. You will have to find JDK directory yourself. It is by default installed in the **Program Files\Java** directory of your system drive.

## Installation Type Selection



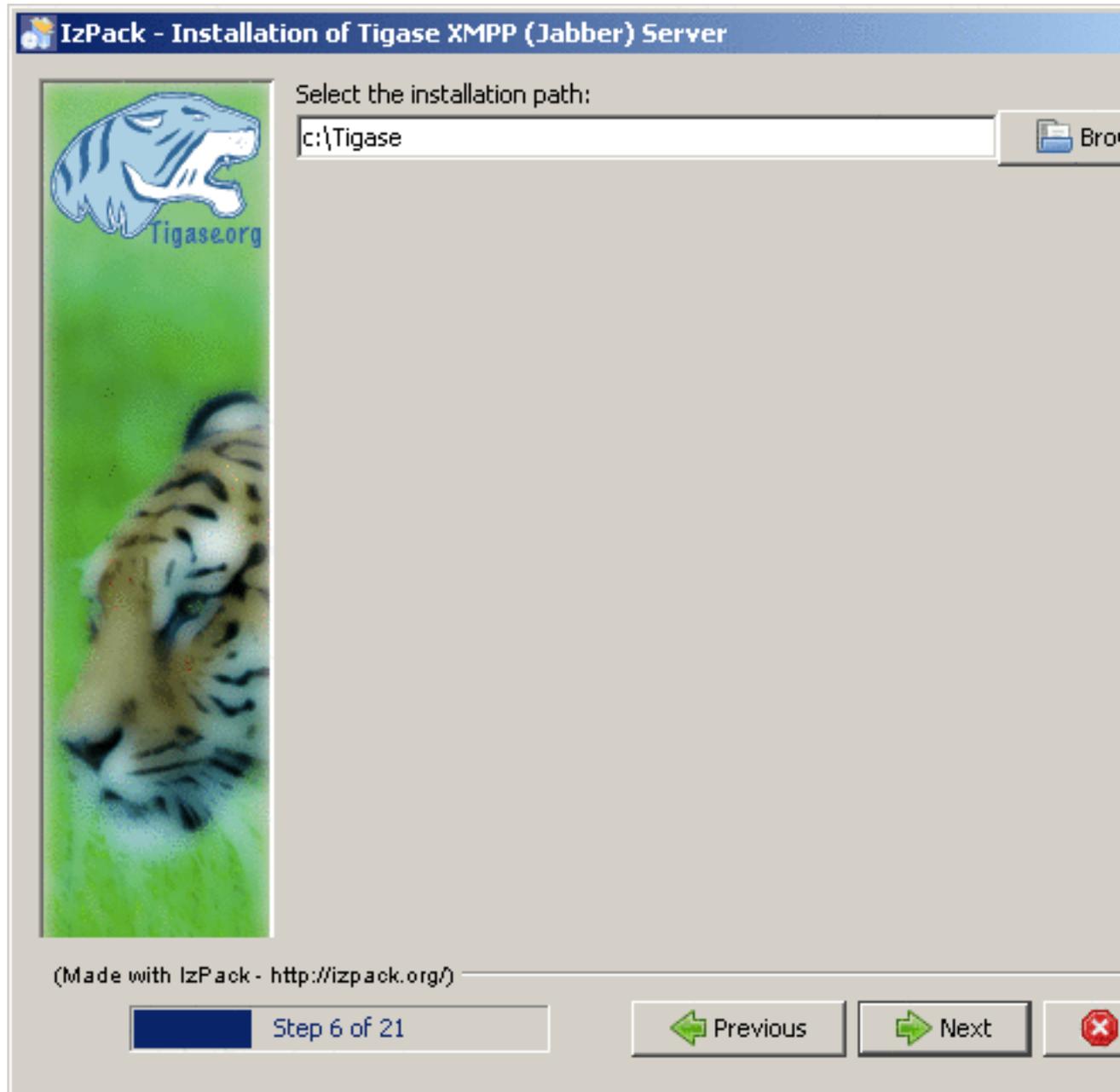
Recommended practice is to choose both installation and configuration of the server as manual configuration is more complicated, time consuming and error-prone.

## Introduction To the Server



This screen shows some information about Tigase which may help you understand what it is and how it can help to take advantage of the **XMPP** protocol. It is important that you read all information's appearing on the installer screens, as they contain valuable hints and most recent information.

## Choice of Base Directory



This is the point where you choose where do you want you server to be installed. Recommended path should not contain spaces, as it may be reason of some strange path problems. In the case of installation on Windows it should be installed on a short path because there is a limit of path lengths. Also note that on Windows Vista there may be some problems with making the server work while installed in the Program Files directory, related to the working of UAC mechanism, so better don't install it there. If you don't want are unsure about where to place the server, you can always leave the default selection.

## Packages selection



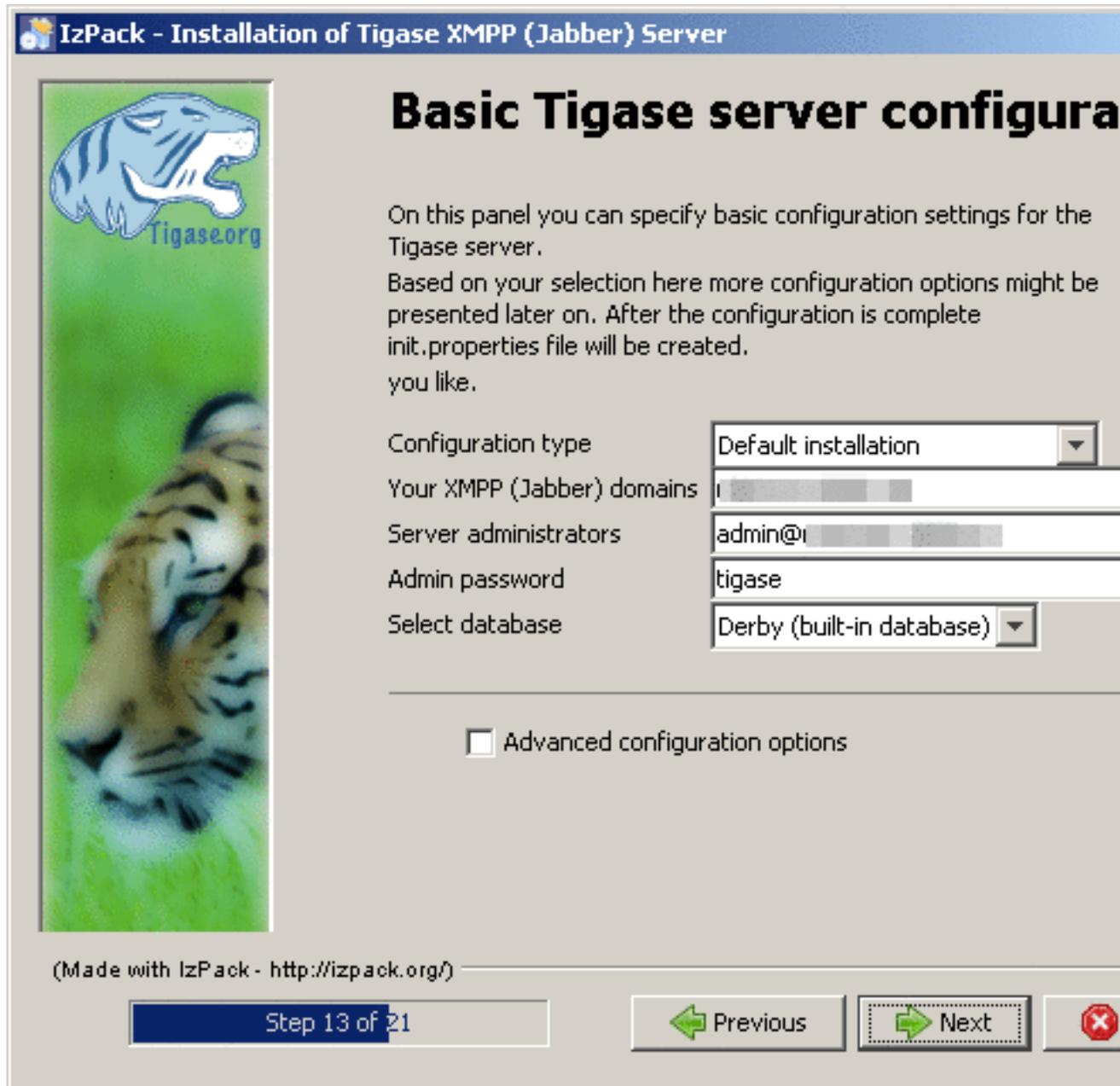
Next important step is package selection. Some choices are grey and you cannot change them as they are essential. All of the rest is optional. It consists of documentation, database drivers, sources of the server and some extras. When you select an item, you will be presented with a short description of it's content.

We recommend you to install documentation. It contains valuable resources which may be very helpful in administration and general use of the server.

If you have a working database platform that you want to use for storing all important user information in, just select appropriate db drivers. If you don't have a database engine, you can use the included derby along with also included drivers.

If you are a developer and you want to be able to check how the server is working or you want to help with the development, you can install also the included source codes.

## Basic Server Configuration



On this screen you will find most important basic configuration options. As this guide covers only non-advanced set up - disable the advanced configuration checkbox.

You can select which components will be installed. For most installations default selection will most appropriate. You can expand the list to check if any of the other options will better suit your needs.

It is very important that you enter your domain name correctly here.

- On **Linux** like system you can use the **hostname** command and extract the domain part from the output. If you use the **-f** parameter then you will get the fully qualified domain name.
- On **Windows** use the standard **System** control panel applet. You will find your domain (computer name) in the **Computer name** tab.

On the other hand if you want to use Tigase virtual domain support and you have your **DNS** system configured properly, then you can put your virtual domains list here. Just separate them by comma characters. For example if your server is seen from the outside as **veloci.tigase.org**, **mammoth.tigase.org** and **tigase.org** then you can use Tigase instance as if it were three separate instances. In reality it will be one server, however **admin@veloci.tigase.org** will be a different user then **admin@tigase.org**. This feature allows to use one server for separating user groups, for example different organizations.

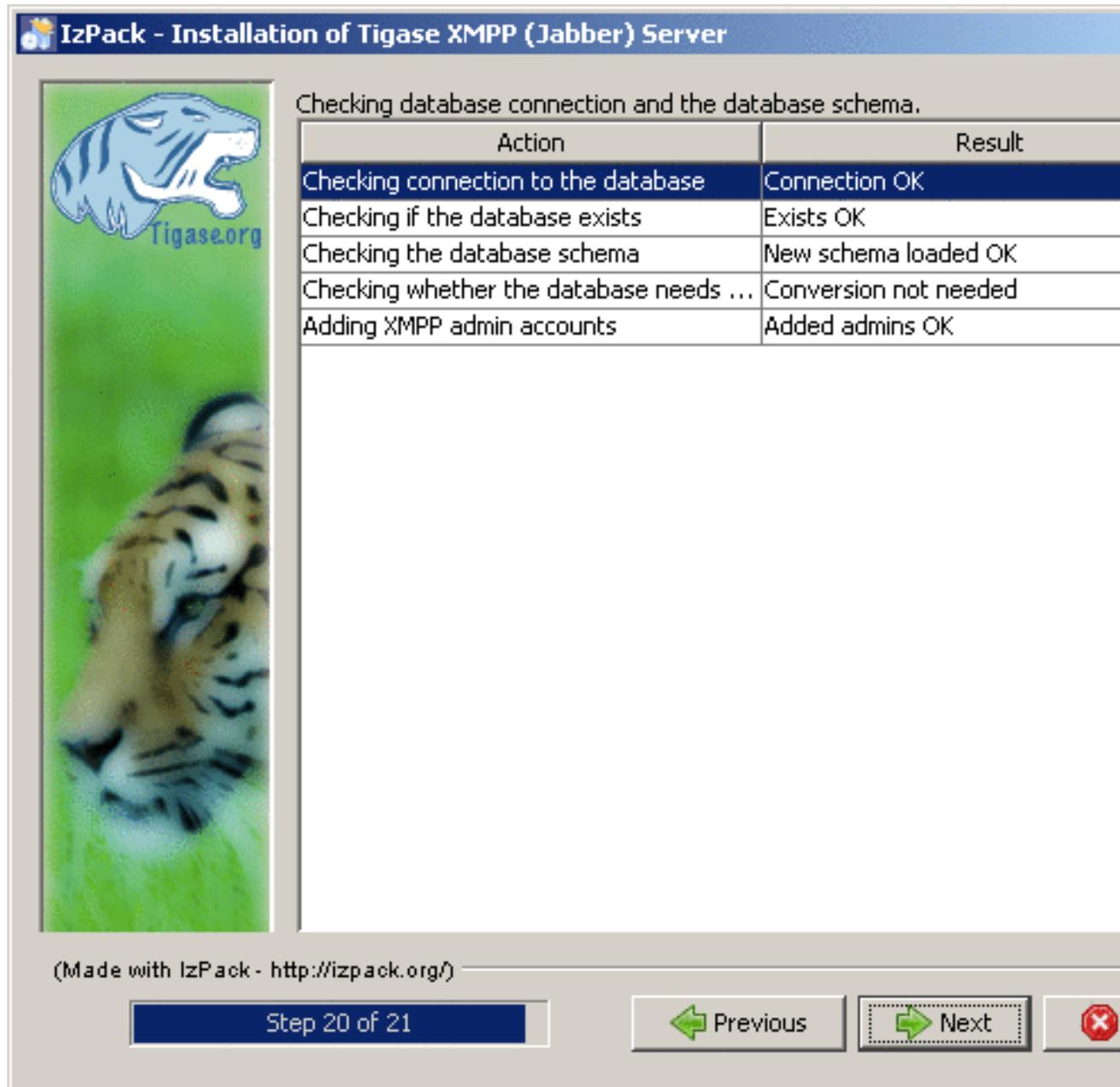
When you will have your domain name just enter it in the domain text box. Next parameter will be the JID of server administrator. Standard practice is to give him name of **admin**, however you may choose any name you like. For example for domain **tigase.org** full admin name would be **admin@tigase.org**. Just stick your chosen name and domain together using the **@** character as separator.

Starting from this version your XMPP admin will be automatically added to the database, so after installation you can just login into the server without registering admin manually.

You should also select a database which will be used for storing user info. Default is the **Derby** database, if you don't need anything special just leave it as it is. Just select a new password as the default one may be easy to guess for a hacker.

**Important notice: Tigase installer doesn't contain the actual databases, only drivers allowing db access. One exception is Derby database, which is included in JDK. It is automatically configured by installer, in case of other databases you will need to configure them by yourself.**

## Verification of Database Connection and Performing DB Tasks



When you switch to this screen an automatic test of database configuration will be started. It consists of few steps which will be executed in order. After testing connection and configuring schema, admin users will be added.

What to do if any of the tests will fail?

- If you decided to use your own database, check if you entered correct password and whether your database is running.

- If you use the embedded **Derby** database then probably your problem is more complicated. An error may indicate a bug in the installer. You may report it to one of Tigase developers.

If you cannot go beyond this step after trying to resolve database problems you may try manual installation mode.

## Finishing Installation

When you perform all those steps altogether with choosing Start Menu location and other basic actions you will be informed that installation process is complete. You can now use your Tigase server. There are some post installation actions you may want to perform. They are briefly presented below.

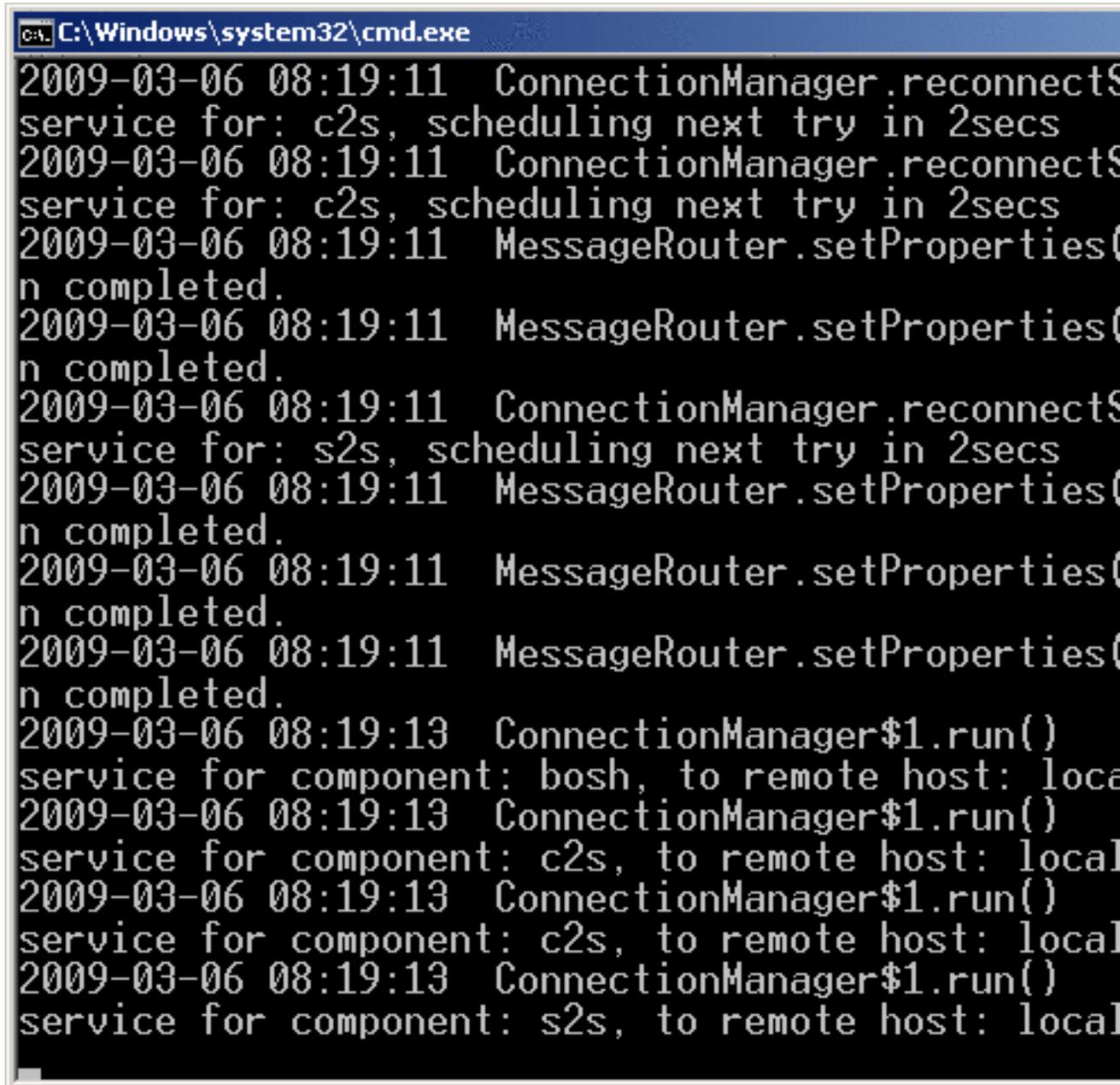
## Running the Server

Part of the installation process is selection of Tigase base directory. This is where you can find all important server files. Installer will create some configurable shortcuts in the **Start Menu**. You can navigate to the menu and use it to start the server. To run the server manually:

- On a **Linux** system you may start the server using the **tigase.sh** file found in the **scripts** sub-directory of Tigase server base directory. In the root server directory type the following command:

```
./scripts/tigase.sh start etc/tigase.conf
```

- Of course if you have a custom config file then change last command appropriately.
- On a **Windows** platform you can use a **bat** file to run the server. There is a **run.bat** file in the Tigase root directory. Just double click it in **Explorer** or run it from command line to start the server. A window with server log output will pop-up.

A screenshot of a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The window displays a series of log messages from Tigase services. The messages include timestamps (2009-03-06 08:19:11 and 08:19:13) and descriptions of service actions such as reconnection attempts for 'c2s' and 's2s' components, and property setting for 'MessageRouter'. The output is as follows:

```
C:\Windows\system32\cmd.exe
2009-03-06 08:19:11 ConnectionManager.reconnectS
service for: c2s, scheduling next try in 2secs
2009-03-06 08:19:11 ConnectionManager.reconnectS
service for: c2s, scheduling next try in 2secs
2009-03-06 08:19:11 MessageRouter.setPropertyes(
n completed.
2009-03-06 08:19:11 MessageRouter.setPropertyes(
n completed.
2009-03-06 08:19:11 ConnectionManager.reconnectS
service for: s2s, scheduling next try in 2secs
2009-03-06 08:19:11 MessageRouter.setPropertyes(
n completed.
2009-03-06 08:19:11 MessageRouter.setPropertyes(
n completed.
2009-03-06 08:19:11 MessageRouter.setPropertyes(
n completed.
2009-03-06 08:19:13 ConnectionManager$1.run()
service for component: bosh, to remote host: local
2009-03-06 08:19:13 ConnectionManager$1.run()
service for component: c2s, to remote host: local
2009-03-06 08:19:13 ConnectionManager$1.run()
service for component: c2s, to remote host: local
2009-03-06 08:19:13 ConnectionManager$1.run()
service for component: s2s, to remote host: local
```

## Installation as a Service

On **Windows** you can install Tigase as a service. To do it use the **InstallTigaseService.bat** batch file found also in server root directory.

In this mode service will be running in background and will be controllable from the **Services** management snapshot. To launch the tool right click on the **Computer** icon on the desktop. Choose the **Manage** action. It will run the **Computer management** graphical configuration program. On the left side choose the **Services** item. You will be shown with a list of services. Here you can find Tigase service when it will be installed.

To uninstall Tigase service use the **UninstallTigaseService.bat** file from Tigase server root directory.

## How to Check if the Server is Running

Checking if the server is running is quite easy. Just try to connect to it by using one of may available **XMPP** clients.

---

# Chapter 8. Installing Using Console Installer

Mateusz Fiolka v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: <http://tigase.net>  
:Date: 2010-04-06 21:18

## Installation Using the text-mode Installer

When you install Tigase server on a desktop machine or a server having a graphical user interface - you can use the `guiInstaller` GUI installer. However servers are often administered using **SSH** text-mode connection because of low connection requirements or user preferences. Because of a popular users demand Tigase development team has decided to implement a console installer for the server.

**Note!** The console installer is available for the Tigase server from version 4.1.5 and later.

## Requirements and Important Notice

Before trying the installer please keep in mind:

- This is first - alpha rate version of the text-mode installation - meant to install the server and do some basic configuration. We will be very happy to see bug reports and overall feedback about this feature. Please send your remarks to Artur Hefczyc [<mailto:artur.hefczyc@tigase.net>] or Mateusz Fiolka.
- You will still need to perform some additional steps before running the installer. Main requirement is to download and install **Java JDK of minimal 1.6 version**. This guide is aimed at advanced users - thus downloading and configuring JDK is left to the reader. You can find more info at the Java downloads site [<http://java.sun.com/javase/downloads/index.jsp>].

## Download the Installer

You can always find the newest Tigase packages in the download section [<https://projects.tigase.org/projects/tigase-server/files/>]. When you enter the page, you will be presented a list of files to choose from. All Tigase binary packages have conventional names, which help to differentiate them easily. They are of form **tigase-server-x.y.z-bv.ext** where 'x', 'y', 'z' and 'v' are version numbers and they change from a release to release. Ext is the file extension which in the case of our console installation program is **.jar**. We recommend you to download the latest version (highest version number) of the server as it contains latest functions and improvements.

## Run the jar File

In terminal (SSH or Windows cmd prompt if you use on) type:

```
java --jar nameOfTheDownloadedJarFile.jar --console
```

## Installation Steps

Now you are ready to install the server.

Some tips for working with the console installer:

- Please remember to write down your JDK location because you will need to type it during the installation process.
- Installer consists of number of screens which relate to different configuration aspects. Most of the panels end with a question whether you want to redisplay the panel or quit installer. When you make a mistake you will be probably asked later if you want re-enter the data again.
- To quit the installer use standard termination key specific to your platform. For example on a **Linux** system it is the Ctrl+C key combination. Keep in mind that if you quit the installer after it copied some files - it may leave them in the place and you might have to remove them manually.
- In current version the installer is of beta quality and using advanced configuration is not recommended. It might work, however it is not much tested and will be improved later.

## Initial Screen

On this screen you will find server version info which will be useful if you would to suggest something to Tigase developers.

```
Welcome to the installation of Tigase XMPP (Jabber) Server X.Y.Z!
```

```
The homepage is at: http://www.tigase.org/  
press 1 to continue, 2 to quit, 3 to redisplay
```

## JDK Selection

Your system may contain more then one JDK installation, thus you will have to make an explicit decision which one to use. Currently console mode installer doesn't contain any auto-detection code nor validation. You will have to enter the path correctly or you may have problems with running the server. For example on **Ubuntu Linux** you can find the JDK in the `/lib/jvm/java-6-sun` directory. For the current Tigase server JDK of version at least **1.6** is required.

The installed application needs a JDK. A java run-time environment (JRE) will be not sufficient.

```
Enter path: -/lib/jvm/java-6-sun
```

## Actions Selection

Choose whether you want to configure the server in addition to install it.

```
*** Select what you want to do next:
```

```
On this panel you can specify whether you want to install  
only or configure already installed server or to do both. If  
you are just installing a server on your machine it is a  
good idea to do both steps.
```

```
The wizards you want to execute  
Installation of the Tigase Server  
[on, off]  
on
```

Configuration of the Tigase Server  
[on, off]  
on

## Installer Info

Introduction to the installer.

Please note!

While the Tigase server is quite stable and well tested application the installer itself is a new addition. Take precautions especially if you upgrade the server from earlier version. Backup old server files and the database.

If you notice any problems please report them to address:  
Artur Hefczyc

press 1 to continue, 2 to quit, 3 to redisplay

## Server Info

If you don't know what exactly is Tigase server, you can find some basic introduction on this screen.

Tigase XMPP (Jabber) server ver 4.1.5-bDEV

About

Copyright (C) 2004 Tigase.org. <<http://www.tigase.org/>>

Tigase Jabber/XMPP Server is  
Open Source and Free (AGPLv3)  
Java based server. The goals behind the design and  
implementation of the server are:

Make the server robust and reliable.  
Make the server secure communication platform.  
Make flexible server which can be applied to different use  
cases.  
Make extensible server which takes full advantage of XMPP  
protocol extensibility.

--- Press ENTER to continue ----

Make the server easy to setup and maintain.

Installation, configuration and compilation

The most recent documentation on all these topics is always  
available in the project website: [www.tigase.org](http://www.tigase.org). Please  
refer to the website for all the details and always up to

date guides.

You would probably want to start with Quick Start:  
<http://www.tigase.org/content/quick-start> documentation.

The website also contains lots of other useful information like load tests results, user discussions and on-line support and help always available to you.

This is 4.1.5-bDEV release of the server. Please include the exact version number in all correspondence regarding the server.

press 1 to continue, 2 to quit, 3 to redisplay

## Server Licence

This is a licence that you have to agree to use Tigase server. Please read it carefully. Take note, that in this manual only part is shown in order to decrease guide length.

Please read the following license agreement carefully:

GNU General Public License -- GNU Project -- Free Software  
Foundation (FSF)

GNU GENERAL PUBLIC LICENSE  
Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc.  
Everyone is permitted to copy and distribute verbatim copies  
of this license document, but changing it is not allowed.

Preamble

...  
... Fragment cut out  
...

You should also get your employer (if you work as a  
programmer) or school, if any, to sign a -"copyright  
disclaimer" for the program, if necessary. For more  
information on this, and how to apply and follow the  
AGPL, see ---- Press ENTER to continue ----

<<http://www.gnu.org/licenses/agpl-3.0.en.html>>.

The GNU General Public License does not permit incorporating  
your program into proprietary programs. If your program is  
a subroutine library, you may consider it more useful to

permit linking proprietary applications with the library.  
If this is what you want to do, use the GNU Lesser General  
Public License instead of this License. But first, please  
read

<<http://www.gnu.org/philosophy/philosophy.html>>.

1. I accept the terms of this license agreement.  
2. I do not accept the terms of this license agreement.  
Choose number (1-2):  
1  
press 1 to continue, 2 to quit, 3 to redisplay

## Server Location Selection

Enter where do you want the server to be installed. If you have administrator rights you can place it in a standard location where all your applications reside. If you don't have write rights for this place, you can always install the server in your home directory.

Select target path  
[/home/user/tigase] -/home/user/tigase-server

press 1 to continue, 2 to quit, 3 to redisplay 1

## Selection of Packs to be Installed

Some packs are optional and you can disable/enable them. In the following screen they have an [x] option before them. To switch their state enter item number and ENTER. When done press d and ENTER.

Select the packs you want to install:

1 => Base, The base files  
2 => Unix Files, Files needed to run the server on Unix like systems  
3 => [x] Docs, The documentation  
4 => [x] Extras, Extras libraries, MUC, PubSub...  
5 => [x] Derby Database, Derby database and JDBC driver  
6 => [x] MySQL Database, MySQL JDBC driver (MySQL has to be  
installed separately)  
7 => [x] PostgreSQL Database, PostgreSQL JDBC driver  
(PostgreSQL has to be installed separately)  
8 => [x] SQL Server Database, SQL Server JDBC driver (SQL  
Server has to be installed separately)  
9 => [ -] Sources, The server source files, tools and  
libraries sources are not included  
r => Redisplay menu  
d => Done

Choose action: d  
press 1 to continue, 2 to quit, 3 to redisplay

## Installation

During extracting and copying server files to their target you will be presented with the process progress.

```
[ Starting to unpack -]
[ Processing package: Base (1/9) -]
[ Processing package: Unix Files (2/9) -]
[ Processing package: Windows Files (3/9) -]
[ Processing package: Docs (4/9) -]
[ Processing package: Extras (5/9) -]
[ Processing package: Derby Database (6/9) -]
[ Processing package: MySQL Database (7/9) -]
[ Processing package: PostgreSQL Database (8/9) -]
[ Processing package: SQL Server Database (9/9) -]
[ Unpacking finished -]
```

## Basic Configuration

This panels contains most important configuration options for the Tigase server. You can choose which components should be configured to be used when running server, add XMPP admin users and enter their password (many admins, comma separated, initially having the same password). Choose different password from the default one. Then select preferred database. If you don't have a standalone DB which you would like to use, you can choose the included Derby DB.

**Important notice: Tigase installer doesn't contain the actual databases, only drivers allowing db access. One exception is Derby database, which is included in JDK. It is automatically configured by installer, in case of other databases you will need to configure them by yourself.**

\*\*\* Basic Tigase server configuration

On this panel you can specify basic configuration settings for the Tigase server.

Based on your selection here more configuration options might be presented later on. After the configuration is complete `init.properties` file will be created.

You can optionally restart the server at the end of the process if you like.

```
0 [x] Default installation
1 [ -] Default plus extra components
2 [ -] Session Manager only
3 [ -] Network connectivity only
input selection:
0
Your XMPP (Jabber) domains [my-laptop]

Server administrators [admin@my-laptop]

Admin password [tigase]

0 [x] Derby (built-in database)
1 [ -] MySQL
2 [ -] PostgreSQL
3 [ -] SQLServer
4 [ -] Other...
```

```
input selection:  
1
```

## Advanced Configuration

**Please note: in this version advanced configuration is not supported. Although it may work it has not been tested and thus is not recommended. Please enter off to not use it.**

```
Advanced configuration options  
[on, off]  
off  
press 1 to continue, 2 to quit, 3 to redisplay
```

## Database Configuration

Depending on which database you did select, you will be presented with related options to configure its connectivity options. As you will see, the parameters have default values.

\*\*\* Database configuration:

You have selected MySQL database. This database needs additional configuration parameters. Please enter all required information.

MySQL super user account will be used only to create and configure database for the Tigase server. It will not be used by the Tigase server later on.

```
Super user account name: [root]
```

```
WARNING: password will be visible while entering  
Super user password: mysecretpassword  
WARNING: password will be visible while entering  
Retype password: mysecretpassword
```

MySQL database details. It will be created automatically if it does not exist.

```
Database account: [tigase]
```

```
Account password: [tigase12]
```

```
Database name: [tigasedb]
```

```
Database host or IP: [localhost]
```

```
Additional database parameters: []
```

```
press 1 to continue, 2 to quit, 3 to redisplay
```

## Database Checking and Preparation

After entering all database information an automatic test of connection and database setup is performed. If everything is ok installer tries to convert database schema to required version and finally adds XMPP administrators to it.

```
Performing DB tasks
```

```
Checking connection to the database
Connection OK
Checking if the database exists
Exists OK
Checking the database schema
New schema loaded OK
Checking whether the database needs conversion
Conversion not needed
Adding XMPP admin accounts
Added admins OK
```

## Installation Complete

Now you can run the server and use it.

```
Install was successful
application installed on -/home/user/tigase-server
[ Console installation done -]
```

## Running the Sever

You can start the server using the `tigase.sh` file found in the `scripts` sub-directory of Tigase server base directory. In the root server directory type the following command:

```
./scripts/tigase.sh start etc/tigase.conf
```

Of course if you have a custom config file then change last command appropriately. On a Windows platform you can use a bat file to run the server. There is a `run.bat` file in the Tigase root directory. Just double click it in Explorer or run it from command line to start the server. A window with server log output will pop-up.

## How to Check if the Server is Running

Checking if the server is running is quite easy. Just try to connect to it by using one of may available XMPP clients. Application: Tigase Server

---

# Chapter 9. Manual Installation in Console Mode

Artur Hefczyc <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2010-04-06 21:18

Our preferred way to install the Tigase server is using GUI installer and configuration program which comes with one of the binary packages. Please pick up the latest version of the JAR file in our download section [<http://www.tigase.org/filebrowser/tigase-server>].

In many cases however this is not always possible to use the GUI installer. In many cases you have just an ssh access or even a direct access in console mode only. We are going to provide a text-only installer in one of the next releases but for the time being you can use our binary packages to install the server manually. Please continue reading to learn how to install and setup the server in a few easy steps...

If you have an old version of the Tigase server running and working and you intend to upgrade it please always backup the old version first.

## Get the Binary Package

Have a look at our download area [<http://www.tigase.org/filebrowser/tigase-server>]. Always pick the latest version of the package available. For manual installation either **zip** or **tar.gz** file is available. Pick one of files with filename looking like: **tigase-server-x.y.z-bv.tar.gz** or **tigase-server-x.y.z-bv.zip** where 'x', 'y', 'z' and 'v' are version numbers and they change from a release to release.

## Unpack the Package

Unpack the file using command for the tar.gz file:

```
$ tar --xzvf tigase-server-x.y.z-bv.tar.gz
```

or for the zip file:

```
$ unzip tigase-server-x.y.z-bv.zip
```

A new directory will be created: **tigase-server-x.y.z-bv/**.

Sometimes after unpacking package on unix system startup script doesn't have execution permissions. To fix the problem you have to run following command:

```
$ chmod u+x ./scripts/tigase.sh
```

## Prepare Configuration

If you look inside the new directory, it should like this output:

```
$ ls -l
total 316K
-rw-r--r--  1 265K 2008-12-15 22:24 ChangeLog
-rw-r--r--  1  37K 2008-12-15 22:24 License.html
```

```

-rw-r--r--  1 1.1K 2008-12-15 22:24 README
drwxr-xr-x  6  204 2009-02-03 13:25 certs/
drwxr-xr-x 22  748 2009-02-03 13:25 database/
drwxr-xr-x  3  102 2008-12-15 22:24 docs/
drwxr-xr-x  4  136 2009-02-03 13:25 etc/
drwxr-xr-x  3  102 2009-02-03 13:25 jars/
drwxr-xr-x 12  408 2009-02-03 13:25 libs/
drwxr-xr-x  2   68 2008-12-15 22:24 logs/
-rw-r--r--  1 1.5K 2008-12-15 22:24 package.html
drwxr-xr-x  7  238 2009-02-03 13:25 scripts/

```

At the moment the most important is the etc/ directory with 2 files:

```

$ ls -l etc/
total 8.0K
-rw-r--r-- 1  97 2008-12-15 22:24 init.properties
-rw-r--r-- 1 333 2008-12-15 22:24 tigase.conf

```

Small change in the tigase.conf file is needed. Find a line setting correct **JAVA\_HOME**:

```
JAVA_HOME="${JDKPath}"
```

and replace **\${JDKPath}** with a path to Java installation on your system.

You need also to edit the init.properties file. It contains initial parameters normally set by the configuration program. As you do the installation manually you have to edit this file yourself. It contains already a few lines:

```

$ cat etc/init.properties
config-type=--gen-config-def
--admins=admin@$HOST_NAME
--virt-hosts = $HOST_NAME
--debug=server

```

You have to replace **\$HOST\_NAME** with a domain name used for your XMPP (Jabber) installation. Let's say this is **\jabber.your-great.net**. Your init.properties should look like this then:

```

$ cat etc/init.properties
config-type=--gen-config-def
--admins=admin@jabber.your-great.net
--virt-hosts = jabber.your-great.net
--debug=server

```

You can also use multiple virtual domains if you want. Please have a look at the detailed description for **--virt-hosts** property in the [initPropertiesGuide](#), [init.properties](#) guide and also more detailed information in the [VHtigaseServerGuide](#), [Virtual Hosts in the Tigase Server](#) guide.

Unfortunately this is not all. You also need to configure connection to the database. First you have to decide what database you want to use: **Derby**, **MySQL** or **PostgreSQL**. Then there are to more properties you have to add to the **init.properties**: **--user-db** and **--user-db-uri**. The first property specifies the database type you use and the second the database connection string. For simplicity let's assume you want to use **Derby** database with files located in directory **/var/lib/tigase/derby**. 2 more lines need to be added to the **init.properties** file:

```

$ cat etc/init.properties
config-type=--gen-config-def

```

```
--admins=admin@jabber.your-great.net
--virt-hosts = jabber.your-great.net
--debug=server
--user-db=derby
--user-db-uri=jdbc:derby:/var/lib/tigase/derby
```

This is enough basic configuration to have your Tigase server installation running.

## Prepare Database

Normally the database is prepared for you during the installation process. Now you are on your own. As in section above we prepare your first installation to run with the **Derby** database. Creating and preparing the Derby database is actually quite easy if you use a helper script: `./scripts/db-create-derby.sh` [<https://projects.tigase.org/projects/tigase-server/repository/revisions/master/entry/scripts/db-create-derby.sh>]. The file might not be in your **scripts/** directory if you have an earlier version of the package. Simply download it from the link provided if it is missing and put it in the **scripts/** directory and execute it with the database location as the parameter:

```
$ ./scripts/db-create-derby.sh -/var/lib/tigase/derby
```

There will be lots of output but if there is no error at the end of the output it means your database has been created and it is ready to use and you are ready to....

There might be filesystem access restrictions for the directory: `/var/lib/` and you might want/need to select a different location.

## Start the Server

Starting the server is the easiest part. Simply execute following command:

```
$ ./scripts/tigase.sh start etc/tigase.conf
```

and you should get the output like this:

```
Starting Tigase:
nohup: redirecting stderr to stdout
Tigase running pid=18103
```

## Check if it is Working

The server is started already but how do you know if it is really working and there were no problems. Have a look in the **logs/** directory. There should be a few files in there:

```
$ ls -l logs/
total 40K
-rw-r--r-- 1 20K 2009-02-03 21:48 tigase-console.log
-rw-r--r-- 1 16K 2009-02-03 21:48 tigase.log.0
-rw-r--r-- 1 0 2009-02-03 21:48 tigase.log.0.lck
-rw-r--r-- 1 6 2009-02-03 21:48 tigase.pid
```

2 first files are the most interesting for us: **tigase-console.log** and **tigase.log.0**. The first one contains very limited information and only the most important entries. Have a look inside and check if there are any **WARNING** or **SEVERE** entries. If not everything should be fine.

Now you can connect with a Jabber (XMPP) client of your choice. The first thing to do would be registering the first account - the admin account from your `init.properties` file: `admin@jabber.your-great.net` [<mailto:admin@jabber.your-great.net>]. Refer to your client documentation how to register a new account.

---

# Chapter 10. Tigase Server Binary Updates

Artur Hefczyc <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2010-04-06 21:18

Most projects try to make sure that the SVN trunk code always compiles correctly. In the Tigase server case this is not enough however. Lots of installations out there we know of run just straight from out SVN trunk line. This puts on us an extra responsibility. Therefore our general approach is to run all functional tests before each code commit to ensure it works correctly. Of course this does not guarantee it will work correctly and efficiently in all cases and on all systems.

Some people like to be on the bleeding edge and always use the source code repository trunk some others prefer to stick to stable public releases. There is however a lot of other who would like to be able to use something from the middle - have the most recent features and new bug fixes but at least in "**Beta**" or "**Release-Candidate**" state.

If you look at the **Monitor** on the right hand side of the Tigase website you can see that the currently running version is always higher then the version available for download. If you are interested to update your server to the more recent version without much hassle please continue to read.

Installing Tigase from the public binary packages is pretty straightforward especially if you use the installer. Using SVN trunk sources to compile them for your own installation is not simple thing, however people who decide to do so normally don't need much help or instructions. This document describes how to update your system using so called **Betas** or **RC** versions we release from time to time.

For releasing our Betas or RC versions we use Maven [<http://maven.apache.org/>] tool. All our projects and packages are always available in the Tigase Maven repository [<http://maven.tigase.org/tigase/>]. You can find there both sources, documentation and binary packages ready to download and use.

If you look for example into the Tigase server directory [<http://maven.tigase.org/tigase/tigase-server/>] you can see a list of versions already published. The version with **SNAPSHOT** in name is something we are working on right now or we used to work in the past. The code used to generate the version is very close to what is in SVN repository.

Therefore if you want to use the most recent **SVN** code but don't want to compile it yourself you can download the most recent **SNAPSHOT** and replace your current libraries with it. Please note there might be many snapshots for a single version number. Just pick the last one. The file name consists of the **version number**, the **creation date**, **creation time** and snapshot number: **tigase-server-4.1.1-20090211.142252-8.jar**.

Those who are interested in using more stable version should pick the most recent version **without** the **SNAPSHOT** in its name. There is always only one binary release for each final version. Looking in the version directory you can find 2 more binary files for each final version: one with documentation and another with source codes. Each final release is also tagged [<https://svn.tigase.org/repos/tigase-server/tags/>] in our SVN repository so you can checkout or browse the source code if you need. There is a nicer source code browser [<http://projects.tigase.org/server/trac/browser/tags>] available too.

A few things to remember when you do the update from our maven repository:

1. If you update to the most recent version of the server you **MUST** also update all libraries to the most recent version and you **MUST** also update all other elements like MUC and PubSub. Only this way you can be sure all parts are compatible with each other and will work correctly.

2. Please make sure you **REPLACE** old libraries with new files. A common mistake is to copy new libraries to a directory and leave old files too. This leads to unpredictable problems. Note, tigase server library for example is stored in **jars/tigase-server.jar** file. From the repository however you would download: **tigase-server-4.1.4.jar** file. Make sure you don't have both loaded at runtime.

The instruction may not be accurate or complete. If you run in to any problems or find something wrong with the instruction please let me know. I am always open for suggestions and comments.

---

# Chapter 11. Installation Using Web Installer

Eric Dzewa v1.0, January 2015: New Web Installer! :toc: :numbered: :website: <http://tigase.net> :Date: 2015-01-15

When Tigase XMPP Server starts up, it looks for the default configuration file: **etc/init.properties**. If this file has not been modified you can run the web installer. Which will step you through the process of configuring Tigase.

## Download and Extract

First download Tigase XMPP Server and extract it. You can download the official binaries [<https://projects.tigase.org/projects/tigase-server/files>], or the latest and greatest nightly builds [<http://build.xmpp-test.net/nightlies/dists/>].

```
$ wget http://build.xmpp-test.net/nightlies/dists/2015-01-12/tigase-server-7.0.0-S
$ tar --xf tigase-server-7.0.0-SNAPSHOT-b3752-dist-max.tar.gz
$ cd tigase-server-7.0.0-SNAPSHOT-b3752
```

Please do not run as root.

## Start the Server

```
scripts/tigase.sh start
```

## Verify Tigase is Running

You should see a list of listening ports.

```
$ lsof --i --P
COMMAND  PID  USER  FD  TYPE  DEVICE  SIZE/OFF  NODE  NAME
java     18387  tigase  141u  IPv6  22185825  0t0    TCP  *:8080 (LISTEN)
java     18387  tigase  148u  IPv6  22185834  0t0    TCP  *:5222 (LISTEN)
java     18387  tigase  149u  IPv6  22185835  0t0    TCP  *:5223 (LISTEN)
java     18387  tigase  150u  IPv6  22185836  0t0    TCP  *:5290 (LISTEN)
java     18387  tigase  151u  IPv6  22185837  0t0    TCP  *:5280 (LISTEN)
java     18387  tigase  152u  IPv6  22185838  0t0    TCP  *:5269 (LISTEN)
```

## Connect to the Web Installer

Some points before you can connect:

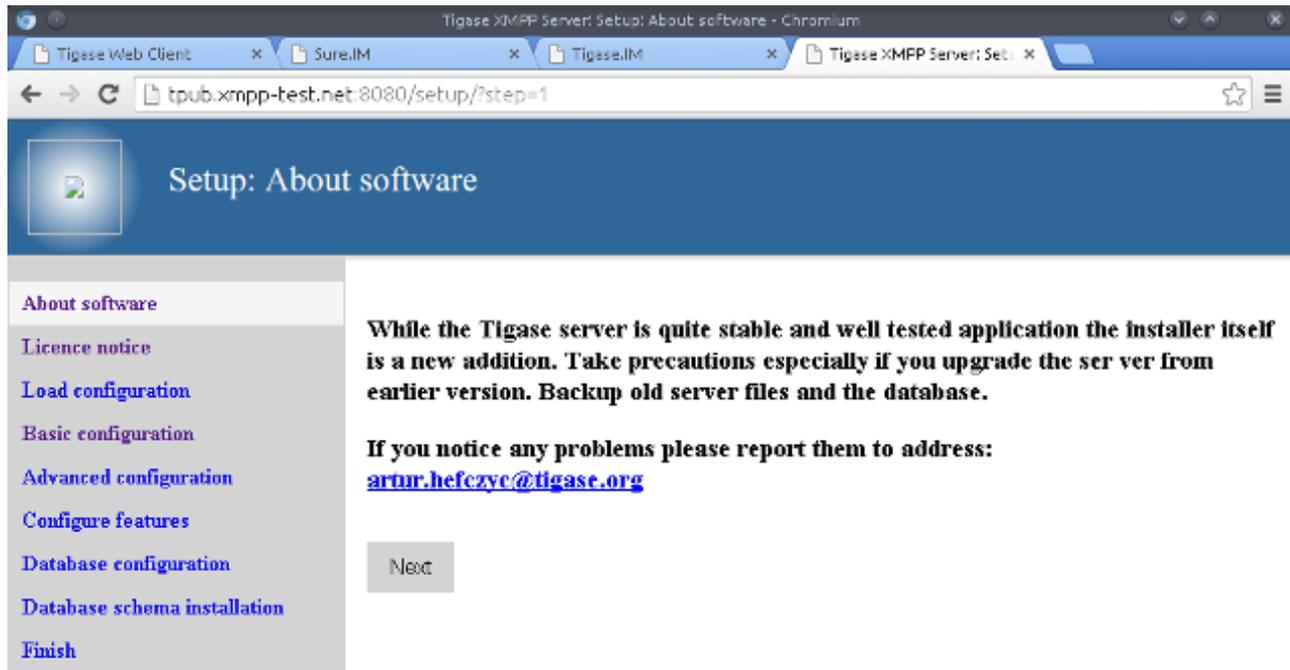
1. If AuthRepository is not configured then access is allowed (initial installation).
2. If it is accessed on localhost then access is allowed.
3. If user is authenticated by jid and password as user from admins of Tigase XMPP Server then access is allowed.

4. In other case access is denied.

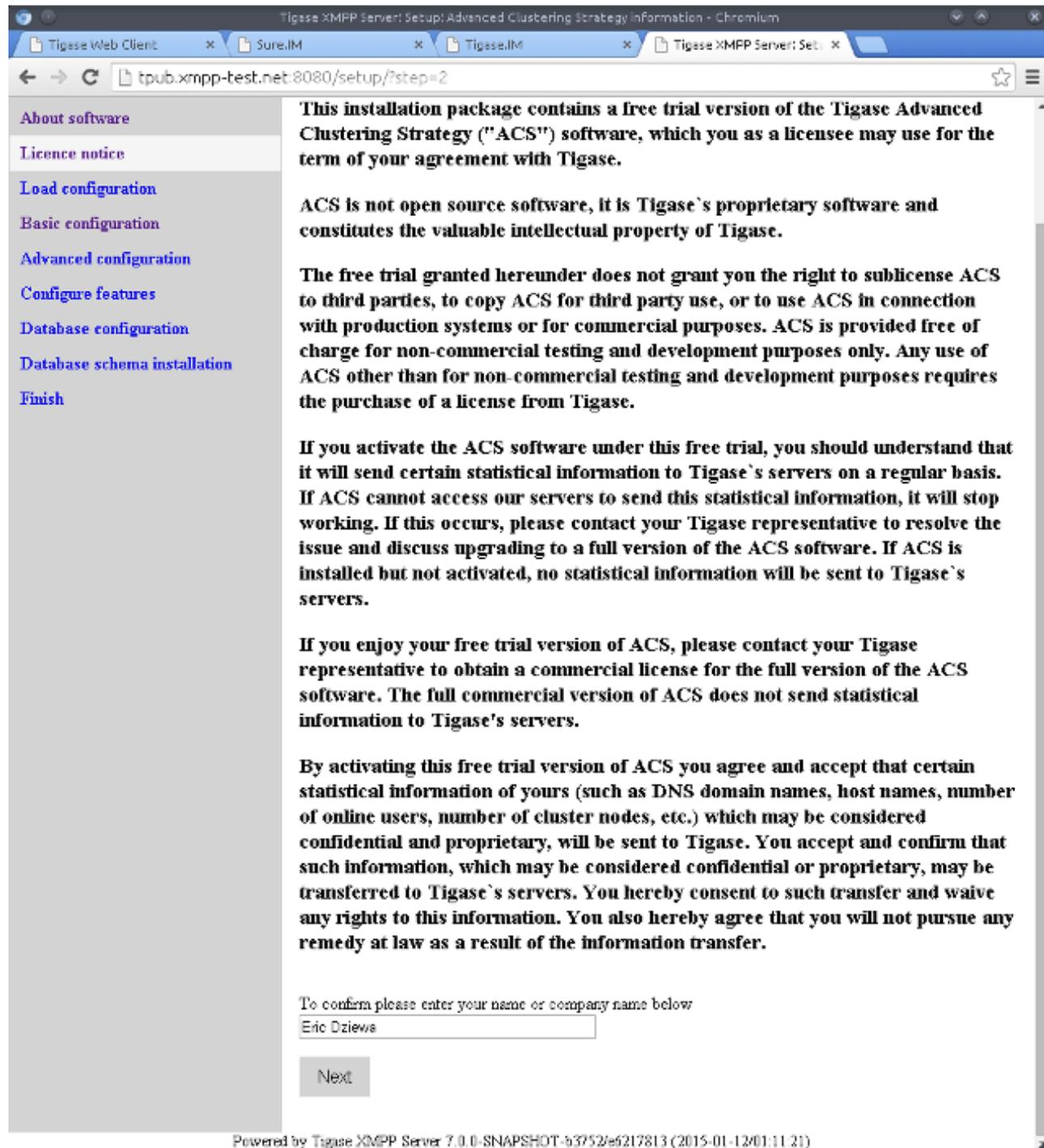
Point your browser to **http://localhost:8080/setup/** unless you are working remotely. Then you can use the domain name, or IP address. I used <http://tpub.xmpp-test.net:8080/setup/> for this guide.

## Step Through the Installation Process

You will be greeted by the following "About software" page.



Read it and then click "Next"



The screenshot shows a web browser window with the title "Tigase XMPP Server: Setup: Advanced Clustering Strategy Information - Chromium". The address bar shows the URL "tpub.xmpp-test.net:8080/setup/?step=2". The page content is as follows:

**About software**

**Licence notice**

**Load configuration**

**Basic configuration**

**Advanced configuration**

**Configure features**

**Database configuration**

**Database schema installation**

**Finish**

**This installation package contains a free trial version of the Tigase Advanced Clustering Strategy ("ACS") software, which you as a licensee may use for the term of your agreement with Tigase.**

**ACS is not open source software, it is Tigase's proprietary software and constitutes the valuable intellectual property of Tigase.**

**The free trial granted hereunder does not grant you the right to sublicense ACS to third parties, to copy ACS for third party use, or to use ACS in connection with production systems or for commercial purposes. ACS is provided free of charge for non-commercial testing and development purposes only. Any use of ACS other than for non-commercial testing and development purposes requires the purchase of a license from Tigase.**

**If you activate the ACS software under this free trial, you should understand that it will send certain statistical information to Tigase's servers on a regular basis. If ACS cannot access our servers to send this statistical information, it will stop working. If this occurs, please contact your Tigase representative to resolve the issue and discuss upgrading to a full version of the ACS software. If ACS is installed but not activated, no statistical information will be sent to Tigase's servers.**

**If you enjoy your free trial version of ACS, please contact your Tigase representative to obtain a commercial license for the full version of the ACS software. The full commercial version of ACS does not send statistical information to Tigase's servers.**

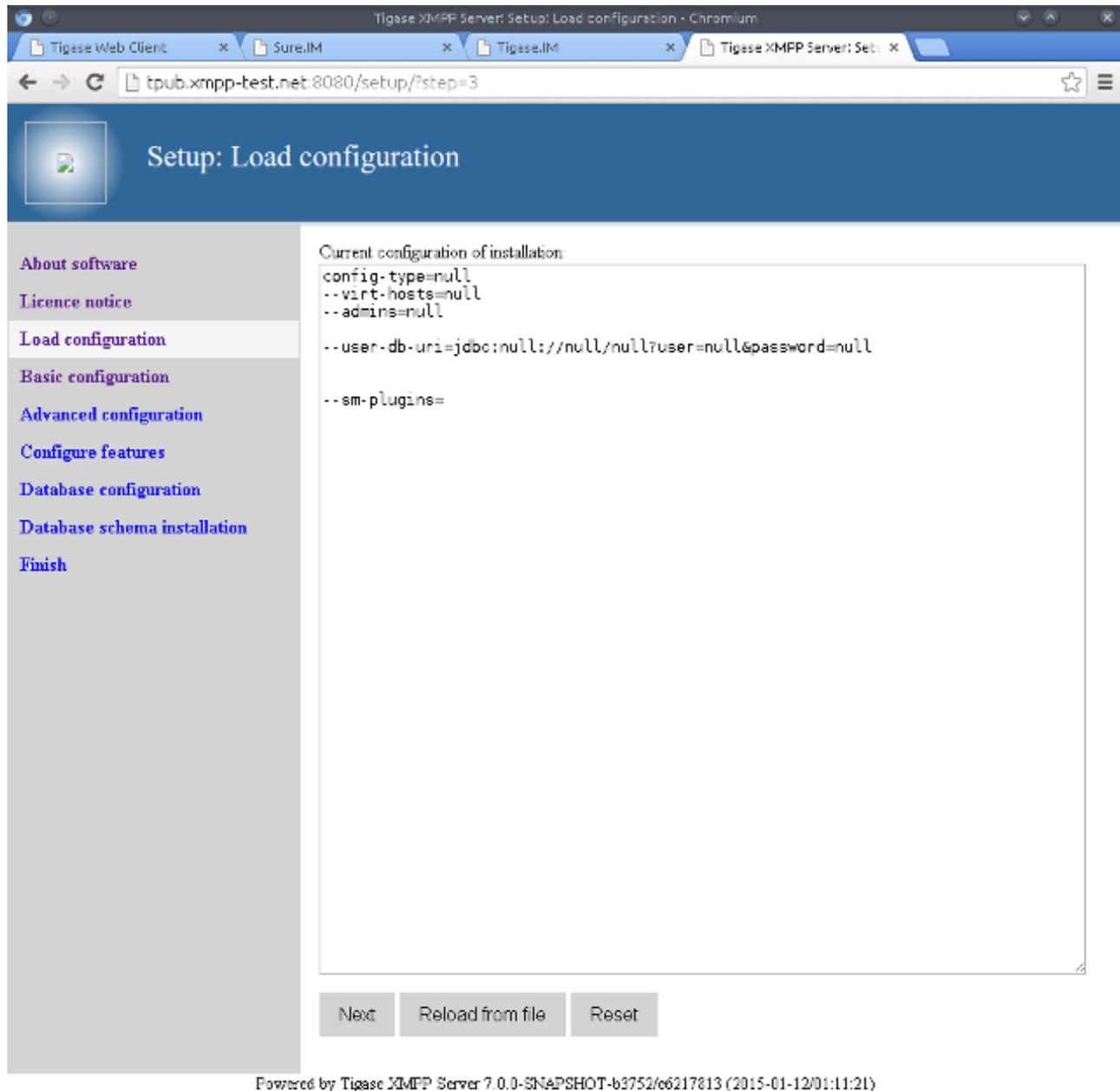
**By activating this free trial version of ACS you agree and accept that certain statistical information of yours (such as DNS domain names, host names, number of online users, number of cluster nodes, etc.) which may be considered confidential and proprietary, will be sent to Tigase. You accept and confirm that such information, which may be considered confidential or proprietary, may be transferred to Tigase's servers. You hereby consent to such transfer and waive any rights to this information. You also hereby agree that you will not pursue any remedy at law as a result of the information transfer.**

To confirm please enter your name or company name below

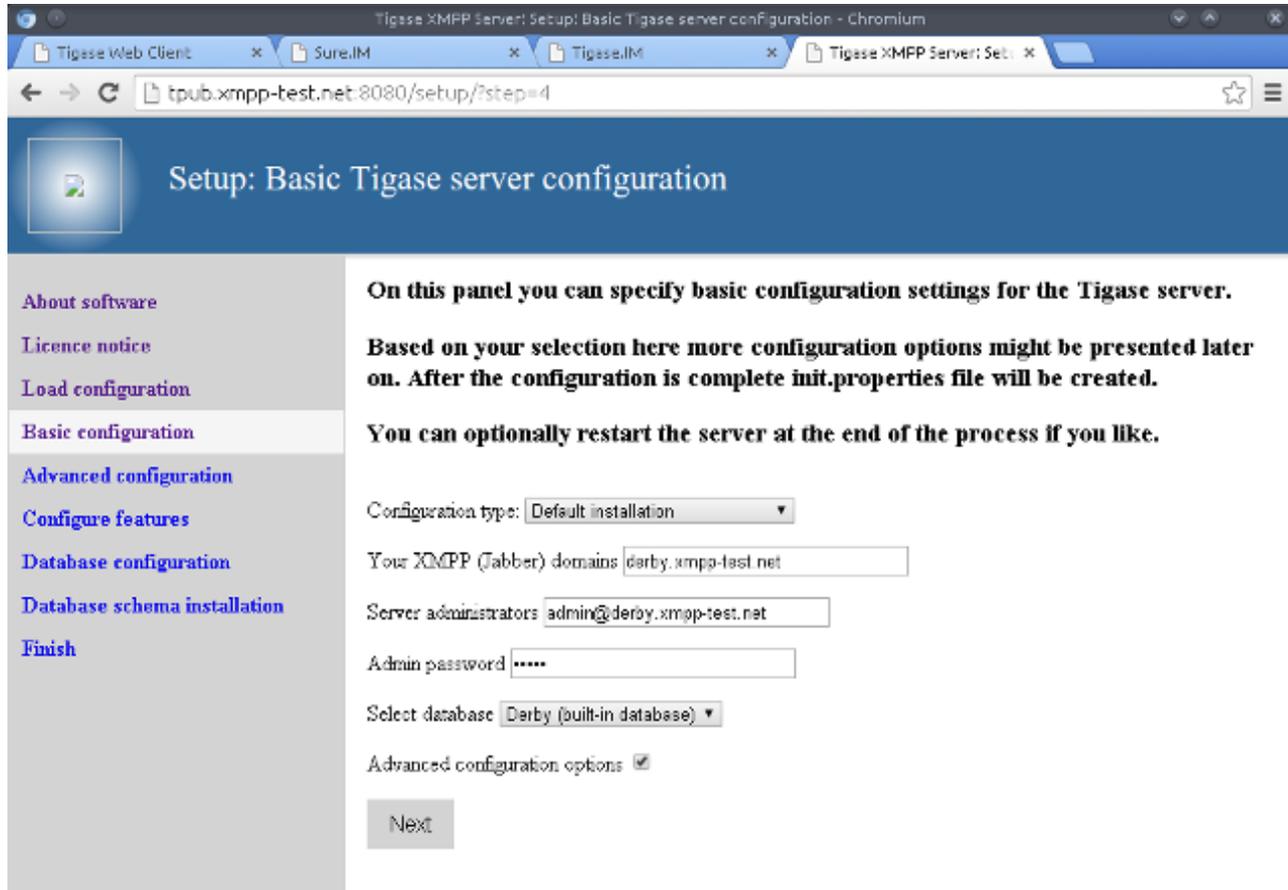
Next

Powered by Tigase XMPP Server 7.0.0-SNAPSHOT-63752e6217813 (2015-01-12/01:11:21)

Here we have the licensing page. Please read though the agreement, type your name and click "Next".

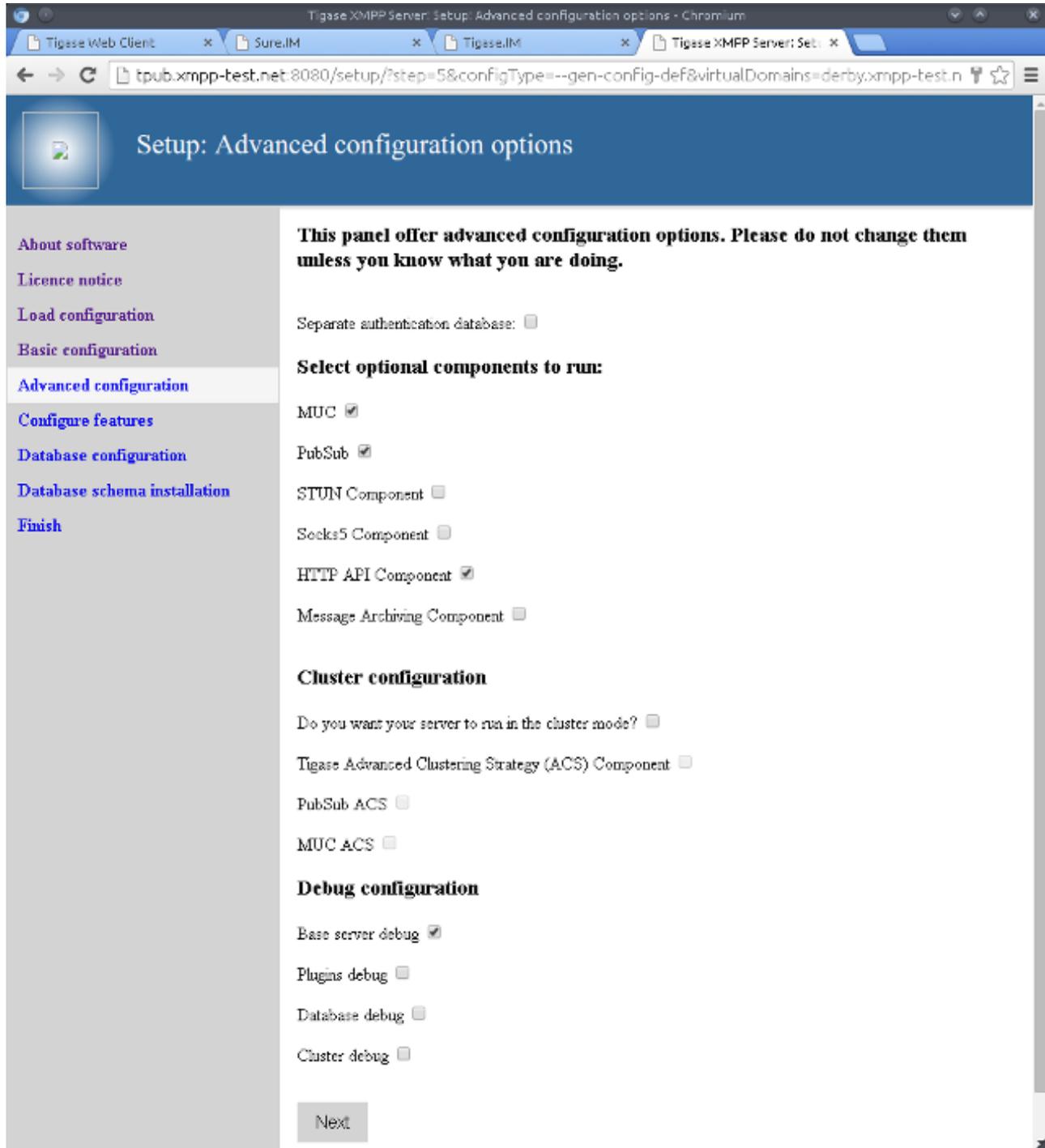


The software reads your current **etc/init.properties** file and presents it. Here we have a bare bones new installation. Click "Next".

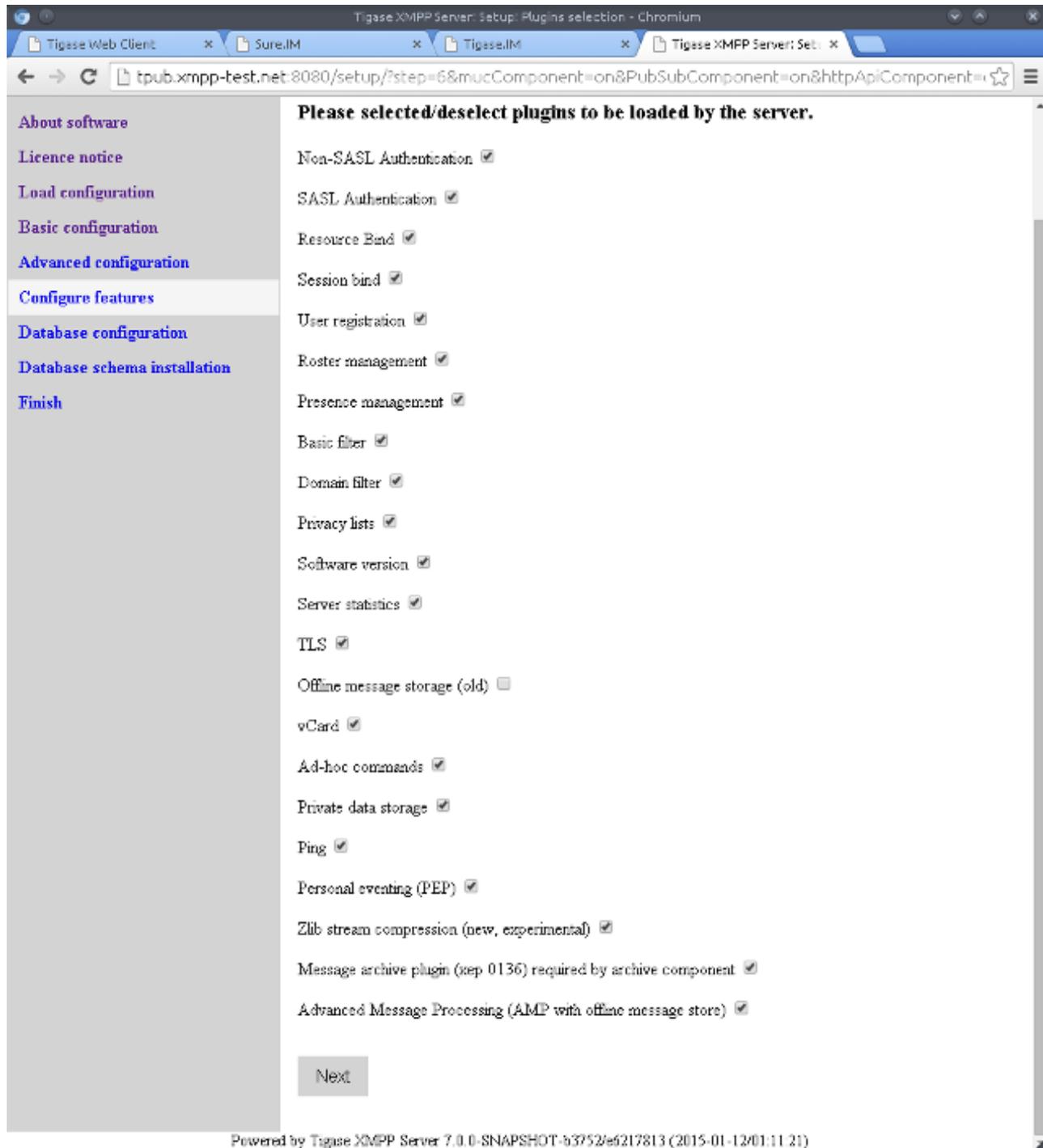


Powered by Tigase XMPP Server 7.0.0-SNAPSHOT-b3752/e6217813 (2015-01-12/01:11:21)

Here we have the domain, database type, and admin account configuration screen. I've chosen Derby. Possible options are MySQL, PostgreSQL, and SQLServer. Tick the advanced box if you think you want a non standard installation.



The Advanced configuration page. Select what you need.



Plugins which will be loaded by the server.

The screenshot shows a web browser window with the title "Tigase XMPP Server: Setup: Database configuration for Derby - Chromium". The address bar shows the URL "tpub.xmpp-test.net:8080/setup/?step=7&jabber%3Aiq%3Aauth=on&urn%3Aietf%3Aparams%3Axml%3Ans%". The page has a blue header with the title "Setup: Database configuration for Derby" and a navigation menu on the left. The main content area contains a message: "You have selected Derby database. This database needs additional configuration parameters. Please enter all required information." Below this are input fields for "Super user account name" (root), "Super user account password" (masked with dots), "Database account" (tigase), "Account password" (tigase12), "Database name" (tigasedb), "Database host or IP" (localhost), and "Additional database parameters". A "Next" button is at the bottom.

**Setup: Database configuration for Derby**

**You have selected Derby database. This database needs additional configuration parameters. Please enter all required information.**

Super user account name:

Super user account password:

**Derby database details. It will be created automatically if it does not exist.**

Database account:

Account password:

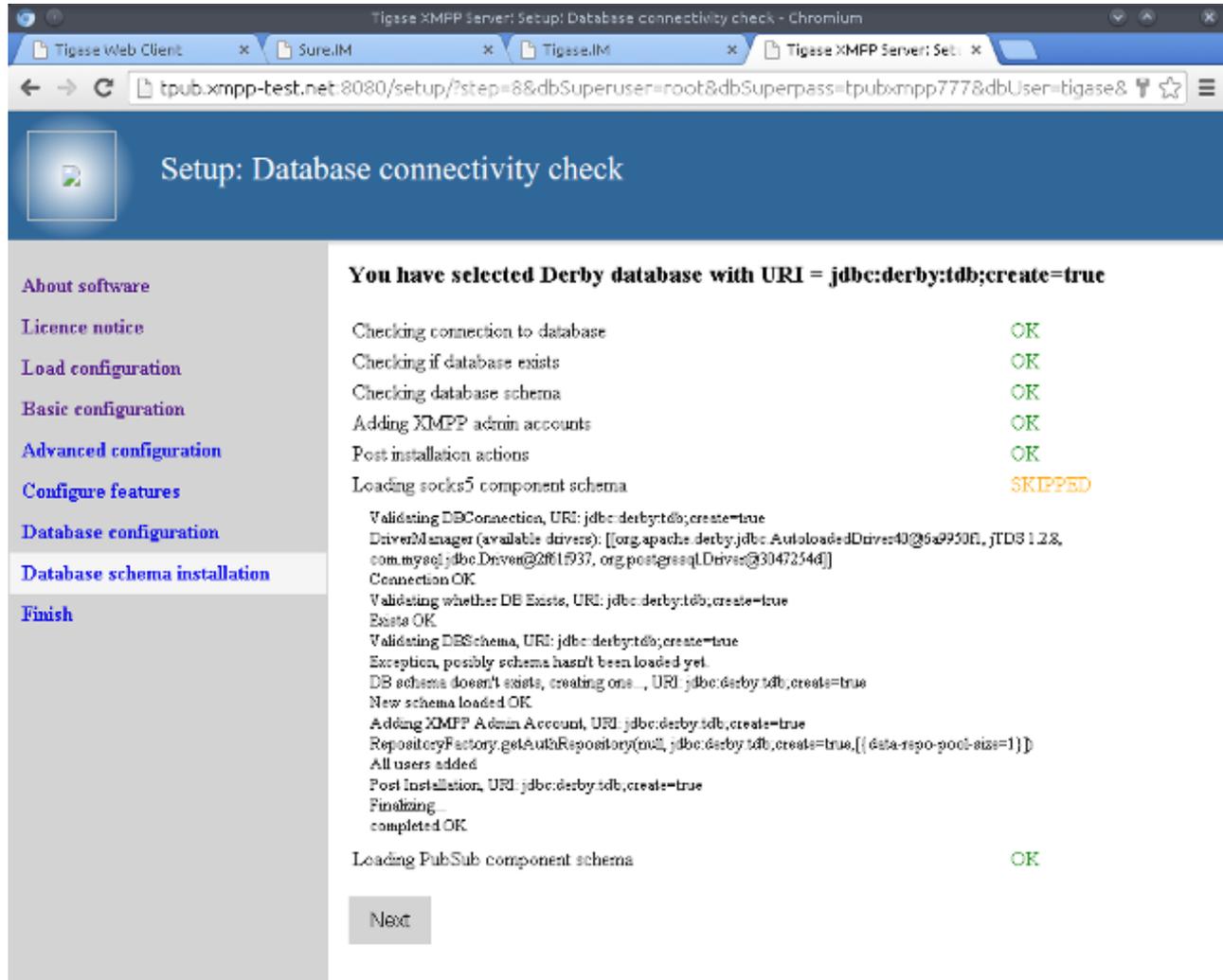
Database name:

Database host or IP:

Additional database parameters:

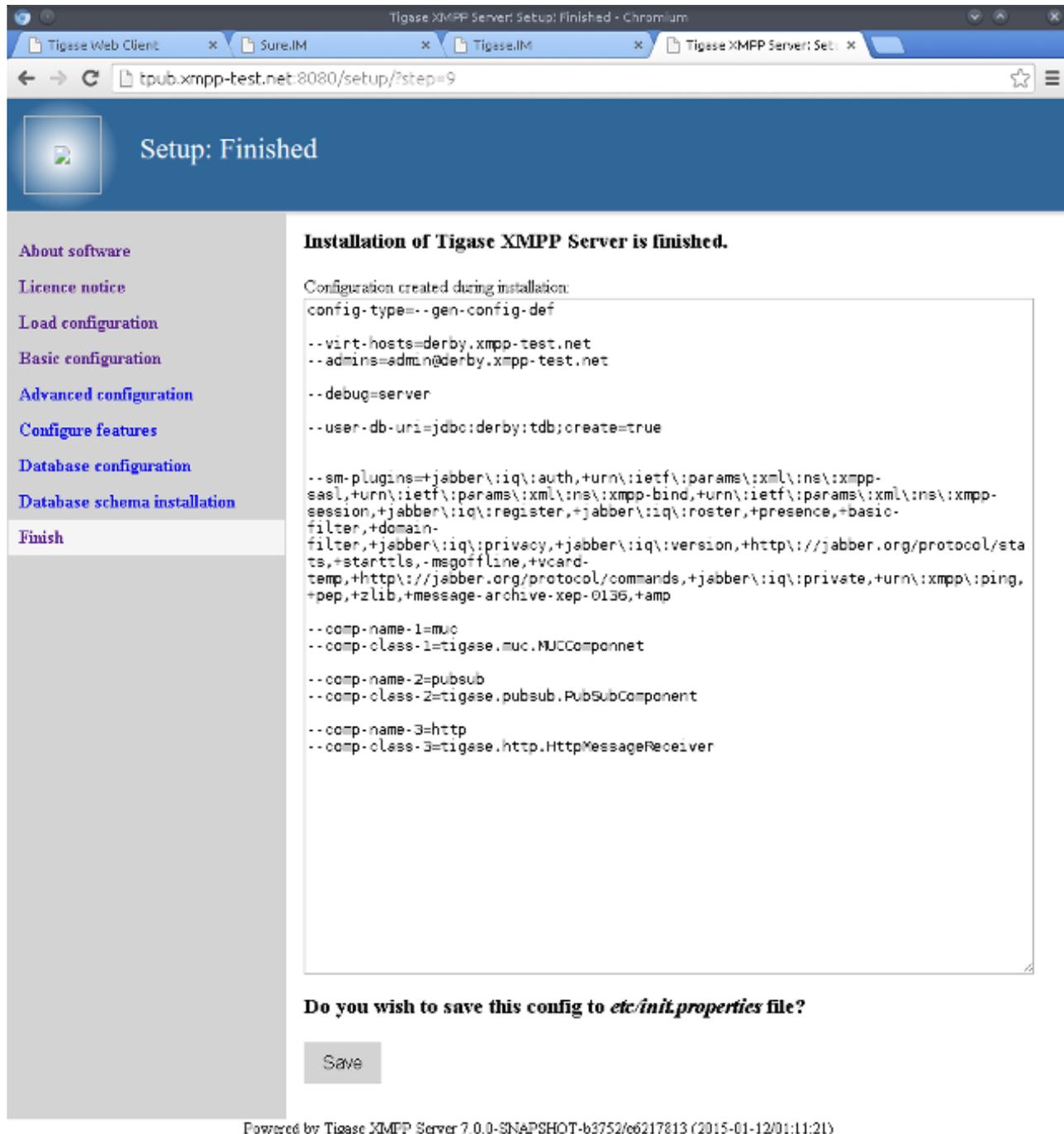
Powered by Tigase XMPP Server 7.0.0-SNAPSHOT-b3752/e6217813 (2015-01-12/01:11:21)

This is where the database schema is installed.



Powered by Tigase XMPP Server 7.0.0-SNAPSHOT-b3752/e6217813 (2015-01-12/01:11:21)

You should see a page like this after success.



Setup: Finished

[About software](#)  
[Licence notice](#)  
[Load configuration](#)  
[Basic configuration](#)  
[Advanced configuration](#)  
[Configure features](#)  
[Database configuration](#)  
[Database schema installation](#)  
[Finish](#)

**Installation of Tigase XMPP Server is finished.**

Configuration created during installation:

```
config-type--gen-config-def
--virt-hosts=derby.xmpp-test.net
--admins=admin@derby.xmpp-test.net

--debug=server

--user-db-uri=jdbc:derby:tdb;create=true

--sm-plugins=+jabber\;iq\;auth,+urn\;ietf\;params\;xml\;ns\;xmpp-
easl,+urn\;ietf\;params\;xml\;ns\;xmpp-bind,+urn\;ietf\;params\;xml\;ns\;xmpp-
session,+jabber\;iq\;register,+jabber\;iq\;roster,+presence,+basic-
filter,+domain-
filter,+jabber\;iq\;privacy,+jabber\;iq\;version,+http://jabber.org/protocol/sta-
ts,+starttls,-megooffline,+vcard-
temp,+http://jabber.org/protocol/commands,+jabber\;iq\;private,+urn\;xmpp\;ping,
+pep,+zlib,+message-archive-xep-0136,+amp

--comp-name-1=muc
--comp-class-1=tigase.muc.MUCComponnet

--comp-name-2=pubsub
--comp-class-2=tigase.pubsub.PubSubComponent

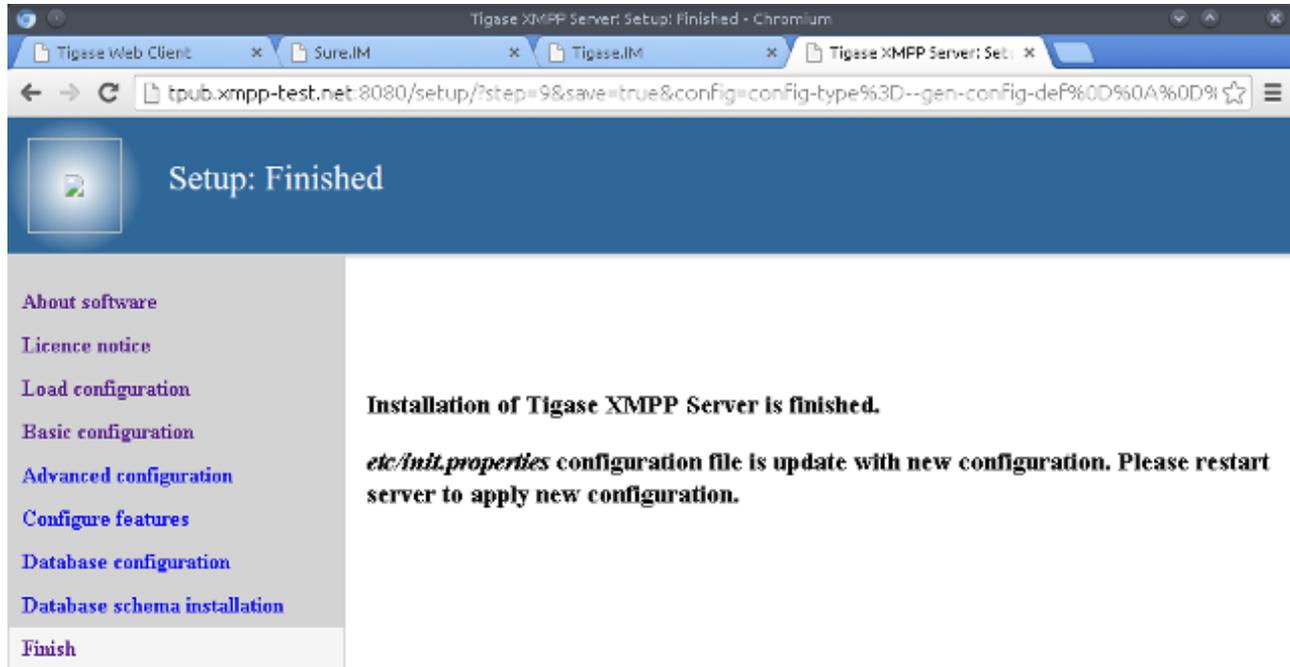
--comp-name-3=http
--comp-class-3=tigase.http.HttpMessageReceiver
```

**Do you wish to save this config to *etc/init.properties* file?**

Save

Powered by Tigase XMPP Server 7.0.0-SNAPSHOT-b3752/e6217813 (2015-01-12/01:11:21)

The installation is complete and this is what the new **init.properties** will look like. Click "Save" to write the file to disk.



Powered by Tigase XMPP Server 7.0.0-SNAPSHOT-b3752fe6217813 (2015-01-12/01:11:21)

Setup has finished. Go back into the installation directory and restart.

```
$ scripts/tigase.sh stop
$ scripts/tigase.sh start
```

To further fine tune the server you should edit `etc/tigase.conf`. Ensure **JAVA\_HOME** path is correct, and increase memory if needed using **JAVA\_OPTIONS** `-Xmx` (max), and `-Xms` (initial). You will need to direct Tigase to read settings from this file on startup as follows.

```
$ scripts/tigase.sh start etc/tigase.conf
```

Everything should be running smooth at this point. Check the logfiles in logs/ if you experience any problems.

---

# Chapter 12. Tigase Server and Multiple Databases

Artur Hefczyc <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2012-10-02 04:23

Splitting user authentication data from all other XMPP information such as roster, vcards, etc... was almost always possible in Tigase XMPP Server. Possible and quite simple thing to configure. Also it has been always possible and easy to assign a different database for each Tigase component (MUC, PubSub, AMP), for recording the server statistics. Almost every data type or component can store information in a different location, simple and easy to setup through the configuration file.

However it is much less known that it is also possible to have a different database for each virtual domain. This applies to both the user repository and authentication repository. This allows for very interesting configuration such as user database sharding where each shard keeps users for a specific domain, or physically split data based on virtual domain if each domain refers to a different customer or group of people.

How can we do that then?

This is very easy to do through the Tigase's configuration file.

Typically the well known lines:

```
--auth-db=tigase-custom
--auth-db-uri=jdbc:mysql://db1.tigase/dbname?user&password
--user-db=mysql
--user-db-uri=jdbc:mysql://db2.tigase/dbname?user&password
```

Define just a default databases for both user repository and authentication repository. Default means it is used when there is no repository specified for a particular virtual domain. However, you can have a separate, both user and authentication repository for each virtual domain.

Here is, how it works:

```
# First, let's define our default database for all VHosts
--auth-db=tigase-custom
--auth-db-uri=jdbc:mysql://db1.tigase/dbname?user&password
--user-db=mysql
--user-db-uri=jdbc:mysql://db2.tigase/dbname?user&password

# Now, we have VHost: domain1.com
# User authentication data for this VHost is stored in Drupal database
--auth-db[domain1.com]=drupal
--auth-db-uri[domain1.com]=jdbc:mysql://db7.tigase/dbname?user&password
# All other user data is stored in Tigase's standard database in MySQL
--user-db[domain1.com]=mysql
--user-db-uri[domain1.com]=jdbc:mysql://db4.tigase/dbname?user&password

# Next VHost: domain2.com
# User authentication is in LDAP server
--auth-db[domain2.com]=tigase.db.ldap.LdapAuthProvider
# Pretty standard Tigase's definition for the database (repository)
# connection string
```

```
--auth-db-uri[domain2.com]=ldap://ldap.domain2.com:389
# Now is something new, we have a custom authentication repository
# settings for a single domain.
# Please note how we define the VHost for which we set custom parameters
basic-conf/auth-repo-params/domain2.com/user-dn-pattern=cn=,ou=,dc=,dc=
# All other user data is stored in the same as default repository
--user-db[domain2.com]=mysql
--user-db-uri[domain2.com]=jdbc:mysql://db2.tigase/dbname?user&password

# Next VHost: domain3.com
# Again user authentication is in LDAP server but pointing to
# a different LDAP server with different access credentials
--auth-db[domain3.com]=tigase.db.ldap.LdapAuthProvider
# Pretty standard Tigase's definition for the database
# (repository) connection string
--auth-db-uri[domain3.com]=ldap://ldap.domain3.com:389
# Now is something new, we have a custom authentication
# repository settings for a single domain
# Please note how we define the VHost for which we set custom parameters
basic-conf/auth-repo-params/domain3.com/user-dn-pattern=cn=,ou=,dc=,dc=
# All other user data is stored on the domain3 server in PostgreSQL database
--user-db[domain3.com]=pgsql
--user-db-uri[domain3.com]=jdbc:pgsql://db.domain3.com/dbname?user&password

# For VHost: domain4.com all the data, both authentication and
# user XMPP data are stored on a separate
# MySQL server with custom stored procedures for both user
# login and user logout processing.
--auth-db[domain4.com]=tigase-custom
--auth-db-uri[domain4.com]=jdbc:mysql://db14.domain4.com/dbname?user&password
basic-conf/auth-repo-params/domain4.com/user-login-query={ call UserLogin(?, -?) -
basic-conf/auth-repo-params/domain4.com/user-logout-query={ call UserLogout(?) -}
basic-conf/auth-repo-params/domain4.com/sasl-mechs=PLAIN,DIGEST-MD5
--user-db[domain4.com]=mysql
--user-db-uri[domain4.com]=jdbc:mysql://db14.domain4.com/dbname?user&password
```

As you can see, it requires some writing but flexibility is very extensive and you can setup as many separate databases as you need or want. If one database (recognized by the database connection string) is shared among different VHosts, Tigase still uses a single connection pool, so it won't create an excessive number of connections to the database.

I hope this helps with more complex setups and configuration cases.

---

# Chapter 13. Basic System Checks

Artur Hefczyc <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: <http://tigase.net> :Date: 2010-03-06 20:18

A while ago, I have written an article about Linux settings for high load systems. This is a description for just very basic settings which are essential to successfully run XMPP service for hundred of thousands online users.

Of course, high load and high traffic systems require much more tuning and adjustments. If you use selinux you have to be careful as it can interfere with the service while it is under a high load. Also some firewall settings may cause problems as the system may decide it is under a DOS attack and can start blocking incoming connections or throttle the traffic.

In any case, there are some basic checks to do every time you deploy XMPP service to make sure it will function properly. I am trying to keep the article mentioned above up to date and add all the settings and parameters I discover while working with different installations. *If you have some suggestions for different values or different parameters to add, please let me know.*

The article, while helpful, seems to be not enough though.

If you want to run a service on a few cluster nodes (5 or even 10), then manually checking every machine and adjusting these settings is quite annoying and it is very easy to forget about something.

To overcome this problem I started to work on a shell script which would run all the basic checks and report problems found. Ideally it should be also able to adjust some parameters for you.

I have just committed to the Tigase server trunk/scripts/ [<https://svn.tigase.org/repos/tigase-server/trunk/scripts/>] repository a script called machine-check.sh. It performs all the basic checks from the article and also tries to adjust them when necessary. Have a look at the code [<https://svn.tigase.org/anon-web-svn/filedetails.php?reponame=tigase-server&path=%2Ftrunk%2Fscripts%2Fmachine-check.sh>] or check it out [<https://svn.tigase.org/repos/tigase-server/trunk/scripts/machine-check.sh>] and run for yourself.

Any comments or suggestions, as usually, very much appreciated.

---

# Chapter 14. Linux Settings for High Load Systems

Artur Hefczyc <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2010-04-06 21:18

There are a few basic settings you have to adjust for high load systems to make sure the server have enough resources to handle big number of network connections.

The main parameter is a maximum number of opened files allowed for the process to keep at the same time. Each network connection uses a file handler therefore if the limit is too low you can quickly run out of handlers and the server can not accept any more connections.

This limit is set on 2 levels - on the kernel level (**fs.file-max**) and on the system level (**nofile**).

Another kernel property which can be important in certain configurations (like transports installations or when you use proxy for Bosh connections) is: **net.ipv4.ip\_local\_port\_range**. This parameter can be set the same way as the fs.file-max property.

## fs.file-max

The **fs.file-max** kernel property is set via sysctl command. You can see current settings executing command:

```
# sysctl fs.file-max
fs.file-max = 358920
```

If you plan to run high load service with big number of server connections then this parameter should be at least as twice big as the number of network connections you expect to support. You can change this setting executing command:

```
# sysctl --w fs.file-max=360000
fs.file-max = 360000
```

## net.ipv4.ip\_local\_port\_range

You can see current settings executing command:

```
# sysctl net.ipv4.ip_local_port_range
net.ipv4.ip_local_port_range = 32768 61000
```

You can change this setting executing command:

```
# sysctl --w net.ipv4.ip_local_port_range="1024 65000"
net.ipv4.ip_local_port_range = 1024 65000
```

## TCP\_keepalive

According to [blog.kolargol.eu](http://blog.kolargol.eu) [<http://blog.kolargol.eu/2006/06/tcpkeepalive.html>] or [www.gnugk.org/](http://www.gnugk.org/) [<http://www.gnugk.org/keepalive.html>] some keepalive settings should be changed to improve reliability.

```
# sysctl --w net.ipv4.tcp_keepalive_time="60"
net.ipv4.tcp_keepalive_time = 60
# sysctl --w net.ipv4.tcp_keepalive_probes="3"
net.ipv4.tcp_keepalive_probes = 3
# sysctl --w net.ipv4.tcp_keepalive_intvl="90"
net.ipv4.tcp_keepalive_intvl = 90
```

## /etc/sysctl.conf

Above commands let the system remember new settings until the next system restart. If you want to make the change permanent you have to edit file: /etc/sysctl.conf and add the property at the end of the file:

```
fs.file-max=360000
net.ipv4.ip_local_port_range=1024 65000net.ipv4.tcp_keepalive_time=60
net.ipv4.tcp_keepalive_probes=3
net.ipv4.tcp_keepalive_intvl=90
```

It will be automatically loaded next time you start the server. Command:

```
# sysctl --p
```

Causes the /etc/sysctl.conf to be reloaded which is useful when you added more parameters to the file and don't want to restart the server.

## nofile

This is the property used by the system limits. For example running the command `ulimit -a` shows you all limits set for the current user:

```
# ulimit --a
core file size          (blocks, --c) 0
data seg size           (kbytes, --d) unlimited
file size               (blocks, --f) unlimited
pending signals         (-i) 38912
max locked memory       (kbytes, --l) 32
max memory size         (kbytes, --m) unlimited
open files              (-n) 40960
pipe size               (512 bytes, --p) 8
POSIX message queues    (bytes, --q) 819200
stack size              (kbytes, --s) 8192
cpu time                (seconds, --t) unlimited
max user processes      (-u) 38912
virtual memory          (kbytes, --v) unlimited
file locks              (-x) unlimited
```

To make it even more interesting and more complex there are 2 types of system limits: **soft limit** which can be temporarily exceeded by the user and **hard limit** which can not be exceeded. To see your **hard limit** execute command:

```
# ulimit --a --H
core file size          (blocks, --c) unlimited
data seg size           (kbytes, --d) unlimited
file size               (blocks, --f) unlimited
pending signals         (-i) 38912
```

```
max locked memory      (kbytes, --l) 32
max memory size        (kbytes, --m) unlimited
open files              (-n) 40960
pipe size               (512 bytes, --p) 8
POSIX message queues   (bytes, --q) 819200
stack size              (kbytes, --s) unlimited
cpu time                (seconds, --t) unlimited
max user processes     (-u) 38912
virtual memory          (kbytes, --v) unlimited
file locks              (-x) unlimited
```

The hard limits are usually bigger then the soft limits or sometimes the same.

For us the most important parameter is: **open files**. You can change the property in file: `/etc/security/limits.conf`. You have to append 2 following lines to the end of the file:

```
jabber                soft    nofile    350000
jabber                hard    nofile    350000
```

Where the **jabber** is the user name of the account running you IM service. You can also set the limits for all users on the machine in a following way:

```
*                soft    nofile    350000
*                hard    nofile    350000
```

For those changes to make an effect you have to logout from the modified account and login again. New limits should be applied.

## su and init script

If one intends to use init scripts for startup purposes (or simply wants to be able to start the server utilizing su command it's necessary to adjust PAM configuration by modifying `/etc/pam.d/su` file and uncomment following line:

```
session    required    pam_limits.so
```

Afterwards that init scripts will respect configured limits.

---

# Chapter 15. Generic Documents - Applying to All Tigase Server Versions

Artur Hefczyc <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2010-04-06 21:18

This section keeps set of documents which apply to all the Tigase server version and contain more generic or introductory information.

- `////<<genericConfiguration,////Configuration`
- `////<<genericServerCertificates,////Server certificates`
- `////<<genericPymnsn-t,////Tigase and PyMSN-t transport`
- `////<<genericTwoSessionManagers,////Two or more SessionManagers`
- `////<<genericdatabasePreperation,////Database preparation`
- `////<<genericLoadBalancing,////Tigase Load Balancing`
- `////<<genericSchemaChange51,////Tigase database minor but useful schema change in version 5.1.0`
- `////<<genericStanzaSender,////StanzaSender`
- `////<<genericDebuggingTigase,////Debuging Tigase`
- `////<<genericImportingData,////Importing user data`
- `////<<genericDrupalAuthentication,////Drupal authentication added`

---

# Chapter 16. Configuration

Artur Hefczyc <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2010-04-06 21:18

The main and actually the only configuration for the Tigase server is kept in the XML file. Let's call it **tigase.xml** for further discussion.

When the user tries to setup the client for the first time he comes across 2 other configuration files: **tigase.conf** and **init.properties** which might be confusing. Here is a brief explanation what all those files are about and in other sections you can learn all the details needed to configure the server.

1. `////<<tigaseXMLconfiguration,////tigase.xml` is the only Tigase server configuration file. It stores all the runtime settings and if it is missing the server generates a new file with whole configuration with some default settings + settings read from the **init.properties** file. You may edit the file manually to adjust settings but this is not recommended as manual editing the XML is error prone. Another way of changing this file and changing the configuration is to use ad-hoc commands which allow you to modify configuration at run-time and cause updating the XML file too. Ad-hoc commands method is not polished yet thus it is also not recommended. The safest way to tweak server runtime parameters is to put them in the `init.properties` file.
2. `////<<initPropertiesGuide,////init.properties` file is a simple text file with server parameters in form: **key = value**. When the XML configuration file is missing the Tigase server reads **init.properties** file and uses parameters found there as defaults for generation of the XML file. Therefore if you change the **init.properties** file you normally have to stop the server, remove the XML file and start the server again. All the settings from the **init.properties** are read and applied to the XML configuration. The properties file is easy to read and very safe to modify. At the moment this is the recommended way change the server configuration.
3. `////<<genericTigaseConf,////tigase.conf` is the Tigase server startup configuration. It is actually not used by the server itself. It rather contains operating system settings and environment parameters to correctly run the Java Virtual Machine [<http://java.sun.com/>]. It is only useful on the unix-like systems with Bash shell. If you run the server on MS Windows systems **tigase.bat** and **wrapper.conf** files are used instead. The **tigase.conf** file is read and loaded by the `scripts/tigase.sh` shell script which also scans the operating system environment for Java VM and other tools needed.

---

# Chapter 17. Tigase XMPP Server Configuration Properties

Artur Hefczyc <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2013-02-09 03:54

init.properties is a little bit extended version of the Java properties file with (key, value) pairs.

Comment line has it's first non-white space ASCII character either '#' or '!'.

The key starts with first non-white space ASCII character and ends on either first white space ASCII character or either of '=' or ':'. Therefore if your key contains any of '=', ':', or white space characters you have to escape them with backslash '\': \: or \=.

All of examples below specify 'vhosts' as a key and 'test-a, test-b, test-c' as a value:

```
vhosts=test-a, test-b, test-c
vhosts -: test-a, test-b, test-c
    vhosts      =      test-a, test-b, test-c
```

Possible types are:

- **[S]** (or nothing) - Characters string: 'abcdef'
- **[s]** - String array: 'abcdef, ghaijk, lmnopq'
- **[B]** - Boolean: 'true' or 'false'
- **[b]** - Boolean array: 'true, true, false'
- **[L]** - Long number: 1234567890
- **[I]** - Long array: '12334, 45435, 45645'
- **[I]** - Integer number: 123456
- **[i]** - Integer array: '123, 456, 678'

There are a bunch of parameters which have broader meaning than just one property. Some of them affect many configuration settings or can generate whole sections in the XML file. Most of them starts with '\--' double hyphen. Please note, each property put in the init.properties file starting with '\--' becomes a JVM system property (without '\--' at the beginning). Here is a list of all those parameters with description:

**Property name:** ///<<admins,///--admins

**Property name:** ///<<auth-db,///--auth-db

**Property name:** ///<<auth-db-uri,///--auth-db-uri

**Property name:** ///<<auth-domain-repo-pool,///--auth-domain-repo-pool

**Property name:** ///<<auth-repo-pool,///--auth-repo-pool

**Property name:** ///<<auth-repo-pool-size,///--auth-repo-pool-size

**Property name:** `////<<bind-ext-hostnames,////--bind-ext-hostnames`

**Property name:** `////<<bosh-close-connection,////--bosh-close-connection`

**Property name:** `////<<bosh-extra-headers-file,////--bosh-extra-headers-file`

**Property name:** `////<<cl-conn-repo-class,////--cl-conn-repo-class`

**Property name:** `////<<client-access-policy-file,////--client-access-policy-file`

**Property name:** `////<<cluster-connect-all,////--cluster-connect-all`

**Property name:** `////<<cluster-mode,////--cluster-mode`

**Property name:** `////<<cluster-nodes,////--cluster-nodes`

**Property name:** `////<<cm-ht-traffic-throttling,////--cm-ht-traffic-throttling`

**Property name:** `////<<cm-see-other-host,////--cm-see-other-host`

**Property name:** `////<<cm-traffic-throttling,////--cm-traffic-throttling`

**Property name:** `////<<cmpname-ports,////--cmpname-ports`

**Property name:** `////<<comp-class,////--comp-class`

**Property name:** `////<<comp-name,////--comp-name`

**Property name:** `////<<cross-domain-policy-file,////--cross-domain-policy-file`

**Property name:** `////<<data-repo-pool-size,////--data-repo-pool-size`

**Property name:** `////<<debug,////--debug`

**Property name:** `////<<debug-packages,////--debug-packages`

**Property name:** `////<<domain-filter-policy,////--domain-filter-policy`

**Property name:** `////<<elements-number-limit,////--elements-number-limit`

**Property name:** `////<<ext-comp,////--ext-comp`

**Property name:** `////<<extcomp-repo-class,////--extcomp-repo-class`

**Property name:** `////<<external,////--external`

**Property name:** `////<<hardened-mode,////--hardened-mode`

**Property name:** `////<<max-queue-size,////--max-queue-size`

**Property name:** `////<<monitoring,////--monitoring`

**Property name:** `////<<net-buff-high-throughput,////--net-buff-high-throughput`

**Property name:** `////<<net-buff-standard,////--net-buff-standard`

**Property name:** `////<<new-connections-throttling,////--new-connections-throttling`

**Property name:** `////<<nonpriority-queue,////--nonpriority-queue`

**Property name:** `////<<queue-implementation,////--queue-implementation`

**Property name:** `////<<roster-implementation,////--roster-implementation`

**Property name:** `////<<s2s-ejabberd-bug-workaround-active,////--s2s-ejabberd-bug-workaround-active`

**Property name:** `////<<s2s-secret,////--s2s-secret`

**Property name:** `////<<s2s-skip-tls-hostnames,////--s2s-skip-tls-hostnames`

**Property name:** `////<<script-dir,////--script-dir`

**Property name:** `////<<sm-cluster-strategy-class,////--sm-cluster-strategy-class`

**Property name:** `////<<sm-plugins,////--sm-plugins`

**Property name:** `////<<sm-threads-pool,////--sm-threads-pool`

**Property name:** `////<<ssl-certs-location,////--ssl-certs-location`

**Property name:** `////<<ssl-container-class,////--ssl-container-class`

**Property name:** `////<<ssl-def-cert-domain,////--ssl-def-cert-domain`

**Property name:** `////<<stats-history,////--stats-history`

**Property name:** `////<<stringprep-processor,////--stringprep-processor`

**Property name:** `////<<test,////--test`

**Property name:** `////<<tigase-config-repo-class,////--tigase-config-repo-class`

**Property name:** `////<<tigase-config-repo-uri,////--tigase-config-repo-uri`

**Property name:** `////<<tls-jdk-nss-bug-workaround-active,////--tls-jdk-nss-bug-workaround-active`

**Property name:** `////<<trusted,////--trusted`

**Property name:** `////<<user-db,////--user-db`

**Property name:** `////<<user-db-uri,////--user-db-uri`

**Property name:** `////<<user-domain-repo-pool,////--user-domain-repo-pool`

**Property name:** `////<<user-repo-pool,////--user-repo-pool`

**Property name:** `////<<user-repo-pool-size,////--user-repo-pool-size`

**Property name:** `////<<vhost-anonymous-enabled,////--vhost-anonymous-enabled`

**Property name:** `////<<vhost-max-users,////--vhost-max-users`

**Property name:** `////<<vhost-message-forward-jid,////--vhost-message-forward-jid`

**Property name:** `////<<vhost-presence-forward-jid,////--vhost-presence-forward-jid`

**Property name:** `////<<vhost-register-enabled,////--vhost-register-enabled`

**Property name:** `////<<vhost-tls-required,////--vhost-tls-required`

**Property name:** `////<<virt-hosts,////--virt-hosts`

**Property name:** `////<<watchdog_delay,////--watchdog_delay`

**Property name:** `////<<watchdog_ping_type,////--watchdog_ping_type`

**Property name:** `////<<watchdog_timeout,////--watchdog_timeout`

**Property name:** `////<<config-type,////config-type`

---

# Chapter 18. init.properties

Artur Hefczyc <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2010-04-06 21:18

Document has been moved to a new location. Please have a look at the new page with the server properties description: Tigase XMPP Server [initPropertiesGuide](#), [configuration properties](#).

---

# Chapter 19. XML Configuration File Description

Artur Hefczyk <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2010-04-06 21:18

If you look inside configuration file you can see that this is just normal XML file with a few top-level separate sections. These sections are called components.

This is it. Tigase server consists of components and without components there is no application at all. This is why the only configuration settings you can find there are only under some component level.

More precisely top level element in this XML file is called: `<tigase-config>`; and it doesn't contain any top level configuration settings.

Under the top level element there are at least 2 or more `<component>` elements. Each component can be distinguished from others by it's 'name'. That is, 'name' attribute is mandatory and must be distinct within configuration file. It is just component ID. Each component can be named any way you like it doesn't need to mean anything. It is just easier to manage configuration if you select sensible names.

`<component>` elements keep configuration settings for server modules.

Example 1:

```
<tigase-config>
  <component name="basic-conf">
    -.... settings
  </component>
  <component name="message-router">
    -.... settings
  </component>
</tigase-config>
```

Configuration settings are kept in "simple" maps like structures (key, values, type) triplets.

- **key** is a configuration parameter identifier or a name of the parameter.
- **values** are just values of the parameter identified by the key. Usually this is just a single value but in some cases there can be more than just one value.
- **type** all configuration parameters have a type. In most cases this is just a String. Other possible types are: Boolean, Integer, Long and corresponding array types: String[], Boolean[], int[], long[].

Configuration settings are stored in `<map>` element which contains list of `<entry>` elements. If there are multiple values for a parameter they are stored as a list of `<item>` elements.

Example 2:

```
<map>
  <entry value="localhost" type="String" key="remote-host"/>
  <entry value="false" type="Boolean" key="demo-mode"/>
  <entry value="1000" type="Integer" key="max-queue-size"/>
  <entry type="String[]" key="hostnames">
    <item value="test-a"/>
```

```
    <item value="localhost" />
  </entry>
  <entry type="int[]" key="ports">
    <item value="5222" />
    <item value="5223" />
  </entry>
</map>
```

Configuration settings can be organised hierarchically using `<node>` elements:

Example 3:

```
<component name="c2s">
  <map>
    <entry type="String[]" key="hostnames">
      <item value="test-d" />
      <item value="localhost" />
    </entry>
  </map>
  <node name="connections">
    <map>
      <entry type="int[]" key="ports">
        <item value="5222" />
        <item value="5223" />
      </entry>
    </map>
    <node name="5222">
      <map>
        <entry value="localhost" type="String" key="remote-host" />
        <entry value="plain" type="String" key="socket" />
        <entry value="accept" type="String" key="type" />
      </map>
    </node>
  </node>
</component>
```

Detailed description for all possible settings is split to per-component chapter. Please look for particular component description for details.

---

# Chapter 20. Old Way - Editing Configuration File Manually

Artur Hefczyk <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2010-04-06 21:18

Options you most likely have to change at deployment time are:

1. **Admin accounts** - account names where all admin messages are sent.
2. **Hostnames** - real and virtual hostnames your server has to serve for.
3. **Logs** - setting related to log file location and how much information should be logged there.

Please let me know if you think more options should be described here.

At the moment the only way to change configuration is to manually edit XML config file. So be prepared for tough times. The good news is that more user friendly interfaces are scheduled for next release. And it will be also reconfigure server at runtime without need to restart service.

## Admin Accounts

This is the most likely thing to change after you install server and generate default configuration. Actually it is also the easiest option to customize.

Open `tigase-config.xml` in you favorite text editor and search for string: "**admins**". You should find section looking like this:

```
<entry type="String[]" key="admins">
  <item value="admin%40your.hostname.com"/>
  <item value="admin%40localhost"/>
</entry>
```

Characters "%40" stand for '@'. So assuming you have just 1 admin account on your installation which is: "**frank@jabber.example.com**" you need to replace above code with:

```
<entry type="String[]" key="admins">
  <item value="frank%40jabber.example.com"/>
</entry>
```

And yes, you can just remove second entry with admin account: "**admin@localhost**" unless you really want to keep it. Be aware though all system messages will be sent to **ALL** admin accounts.

Well, if the account does not exists the message is discarded and a warning is be printed in log file. Again, read it again, the previous sentence...

It means that the admin account has to be also created in normal way on the Jabber server. Just register it using your Jabber client. The admin accounts setting works just as a forward instruction. So as a result all system and admin messages are forwarded to all admin accounts if they exist.

Obviously you can have admin accounts as many as you like:

```
<entry type="String[]" key="admins">
```

```
<item value="frank%40jabber.example.com" />
<item value="lucy%40jabber.example.com" />
<item value="mark%40jabber.example.com" />
<item value="brenda%40jabber.example.com" />
<item value="luck%40jabber.example.com" />
</entry>
```

## Hostnames

This one might be a little bit more tricky than previous as **hostnames** setting has to be changed in a few places. Don't ask why now as this is a "Short Configuration Guide", you remember. Here we focus on **how** not on **why**.

You have to search configuration file for string "**hostnames**". There are more than one such sections and you have to find **ALL** sections looking like:

```
<entry type="String[]" key="hostnames">
  <item value="your.hostname.com" />
  <item value="localhost" />
</entry>
```

It may also look like:

```
<entry type="String[]" key="hostnames">
  <item value="localhost" />
</entry>
```

Depending how successful was mechanism for automatic hostname detection. It of course does not depends on your luck. It depends on network configuration on your server.

The first form is more useful as it includes also hostname recognized in network environment. If it is correct then you can just leave it as it is. If it is incorrect you have to change it. Please remember, if you want your server to be able to communicate with other *Jabber/XMPP* servers the hostname you put there must resolve in DNS to your Jabber server machine IP address. In other words. If you try to connect from the Internet to machine with this hostname the connection should reach your Jabber server.

And remember your Jabber server users' *JIDs* (*Jabber IDs*) can include only those hostnames which are included in the configuration. So for our case you can use only *JIDs*: "*user2@your.hostname.com*", "*user1@your.hostname.com*" and so on.

If you server have more Internet addresses (virtual domains) assigned to it your Jabber server can use them all. So your configuration may look like:

```
<entry type="String[]" key="hostnames">
  <item value="your.hostname.com" />
  <item value="clien1.hostname.com" />
  <item value="another.hostname.com" />
  <item value="jabber.sample-domain.com" />
  <item value="jabber.some-project.org" />
  <item value="localhost" />
</entry>
```

In such case users' *JIDs* on your Jabber server may include any of defined above domains like: "*user1@your.hostname.com*", "*user1@clien1.hostname.com*", "*user1@jabber.sample-domain.com*". Each of these 3 sample *JIDs* refer to different user account.

Your server will accept connections only for domains defined in configuration file.

In majority cases it does not matter whether you leave **"localhost"** or remove it. It is sometimes better to leave it though. So if you are not sure if you can remove it in your environment just leave it as is.

## Logs

Logging mechanism is very flexible in Tigase server. You can adjust separate logging level for each single component. You can also direct logging to many different destinations like console, file, network socket and so on. Unfortunately it also mean it is a bit complex. The general idea however is quite simple so once you understand it it shouldn't be difficult for you anymore. This guide however describes logging very briefly. Look at full configuration documentation for detailed explanation.

In standard sever configuration you usually want to turn off all logging to console and all warning and more serious notices directed to log file. Let's say logs will be written to `/var/log/tigase-server.log` which shouldn't get bigger than 10MB and 5 old logs will be preserved. Here are instructions how to set options.

Open `tigase-config.xml` in you favorite text editor and search for string: **"logging"**. You should find section looking like this:

```
<node name="logging">
  <map>
    <entry value="FINE" type="String" key=".level"/>
    <entry value="java.util.logging.ConsoleHandler+java.util.logging.FileHandler"
    <entry value="tigase.util.LogFormatter" type="String" key="java.util.logging.C
    <entry value="WARNING" type="String" key="java.util.logging.ConsoleHandler.lev
    <entry value="true" type="String" key="java.util.logging.FileHandler.append"/>
    <entry value="5" type="String" key="java.util.logging.FileHandler.count"/>
    <entry value="tigase.util.LogFormatter" type="String" key="java.util.logging.F
    <entry value="ALL" type="String" key="java.util.logging.FileHandler.level"/>
    <entry value="100000" type="String" key="java.util.logging.FileHandler.limit"/
    <entry value="logs%2Ftigase.log" type="String" key="java.util.logging.FileHand
    <entry value="true" type="String" key="tigase.useParentHandlers"/>
  </map>
</node>
```

Assuming we make this guide easy and straightforward let me show how this section should look like after modification. So you could just copy and paste it to your config file without going into details. After the configuration code I will briefly explain what each line means so you should be able to further adjust settings for your needs.

```
<node name="logging">
  <map>
    <entry value="WARNING" type="String" key=".level"/>
    <entry value="java.util.logging.ConsoleHandler+java.util.logging.FileHandler"
    <entry value="tigase.util.LogFormatter" type="String" key="java.util.logging.C
    <entry value="tigase.util.LogFormatter" type="String" key="java.util.logging.F
    <entry value="OFF" type="String" key="java.util.logging.ConsoleHandler.level"/
    <entry value="true" type="String" key="java.util.logging.FileHandler.append"/>
    <entry value="5" type="String" key="java.util.logging.FileHandler.count"/>
    <entry value="ALL" type="String" key="java.util.logging.FileHandler.level"/>
    <entry value="10000000" type="String" key="java.util.logging.FileHandler.limit
    <entry value="%2Fvar%2Flog%2Ftigase-server.log" type="String" key="java.util.l
    <entry value="true" type="String" key="tigase.useParentHandlers"/>
  </map>
</node>
```

```
</map>  
</node>
```

## Each Line Explained:

```
<entry value="WARNING" type="String" key=".level"/>
```

Effectively we set WARNING level for all possible logs for all possible components. So more detailed logging information will be discarded. All possible log levels are: OFF, SEVERE, WARNING, INFO, CONFIG, FINE, FINER, FINEST, ALL.

```
<entry value="java.util.logging.ConsoleHandler+java.util.logging.FileHandler" type="String" key="java.util.logging.ConsoleHandler.handlers" />
```

We set 2 handlers for logging information: console and file handler. As we are going to turn off logging to console we could remove all configuration settings for console handler as well. It would simplify configuration file. I don't recommend it though. If there are any problems with your installation switching console logging on might be very helpful and if you remove these settings from config file it may be difficult to bring them back. Hm... maybe not with such excellent documentation.... ;-)

```
<entry value="tigase.util.LogFormatter" type="String" key="java.util.logging.ConsoleHandler.formatter" />  
<entry value="tigase.util.LogFormatter" type="String" key="java.util.logging.FileHandler.formatter" />
```

We set here log formatter for console and file handler. Standard Java handlers print each log message in 2 lines. Tigase formatter prints all logging info in 1 line which make it much easier to filter logs by log type, logging component or log level or whatever you wish. You can just use simple sed command and that's it.

```
<entry value="OFF" type="String" key="java.util.logging.ConsoleHandler.level"/>
```

Here we just switch console handler off. To switch it on back set any different level from the list above.

```
<entry value="true" type="String" key="java.util.logging.FileHandler.append"/>
```

This settings is to controll whether we want to append logs into old log file or we want to create new log file (removing old content) each time server is restarted.

```
<entry value="5" type="String" key="java.util.logging.FileHandler.count"/>
```

Sets number of old log files to preserve to 5.

```
<entry value="ALL" type="String" key="java.util.logging.FileHandler.level"/>
```

This line sets the logging level for file handler. Here we set that we want all possible logs to be written to the file. The global level setting however says that only WARNING logs will be generated. So if you want to have more detailed logs you need to adjust global logging level.

```
<entry value="10000000" type="String" key="java.util.logging.FileHandler.limit"/>
```

Log file maximum size set to 10MB. After reaching this size the log file is closed and new file is created.

```
<entry value="/var/log/tigase-server.log" type="String" key="java.util.logging.FileHandler.fileName" />
```

Location of the log file and file name: /var/log/tigase-server.log. Please note %2F instead of \ character.

```
<entry value="true" type="String" key="tigase.useParentHandlers" />
```

This setting requires going into more details so it is explained in comprehensive configuration guide.

---

# Chapter 21. Startup File for `tigase.sh` - `tigase.conf`

Artur Hefczyk <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2010-04-06 21:18

Property file name for `tigase.sh` startup script is a second parameter for the startup script. It can be skipped if environmental variables are set in different place or in different way.

Config file for startup script simply sets number of environment variables with location of required components. Possible variables to set in this file are:

- **JAVA\_HOME** - location of Java installation home directory. **Must be set.**
- **TIGASE\_HOME** - location of Tigase installation home directory. *By default script try to find this location by searching directories from the location where the script has been run.*
- **TIGASE\_CONSOLE\_LOG** - file to which all console messages will be redirected if server is run in background. By default it will be: `TIGASE_HOME/logs/tigase-console.log`. **If this file/directory is not writable by Tigase process all console messages will be redirected to /dev/null**
- **TIGASE\_PID** location of the file with server PID number. By default it will be `TIGASE_HOME/logs/tigase.pid`.
- **TIGASE\_CONFIG** - location of the Tigase server config file. This is main config XML file. Not to be confused with startup script parameters file. *If not set script trys to find it in following locations in given order: /etc/conf.d/tigase-server.xml, /etc/tigase-server.xml, /etc/tigase/tigase-server.xml or finally in TIGASE\_HOME/etc/tigase-server.xml*
- **JAVA\_OPTIONS** - options for JVM like size of RAM allocated for the JVM, properties and so on.
- **TIGASE\_OPTIONS** - additional options for Tigase server program. You can tweak here initial parameters for your environment.

Sample file to run **Tigase** with **PostgreSQL** database may look like:

```
ENC="-Dfile.encoding=UTF-8 --Dsun.jnu.encoding=UTF-8"
DRV="-Djdbc.drivers=org.postgresql.Driver"
JAVA_OPTIONS="${ENC} ${DRV} --server --Xms100M --Xmx100M -"
CLASSPATH=""
TIGASE_CONFIG="tigase-pgsql.xml"
TIGASE_OPTIONS=" ---property-file etc/init.properties -"
```

Please note encoding settings. I have received several requests about encoding problems. JVM by default uses encoding set in operating system environment. XMPP protocol, however uses UTF-8 for all data processing. So the above settings enforces UTF-8 encoding for all operations.

Another significant setting is `CLASSPATH`. It is intentionally set to empty string. The `tigase.sh` startup script builds the `CLASSPATH` on it's own from files found in `jars/` and `libs/` directories. I advice to set the `CLASSPATH` to the empty string because the Tigase server scans all available classes to find all components and plugins implementation. If the `CLASSPATH` contains lots of libraries which are not used anyway it cases long startup time and lots of memory consumption.

---

# Chapter 22. Server Certificates

Artur Hefczyc <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2010-04-06 21:18

## Documents Describing How To Obtain and Manage Server Certificates

- [////<<serverCertificatesPEM,////Creating and Loading the Server Certificate in pem Files](#)
- [////<<startcomCertificate,////Installing StartCom Certificate in Your Linux System](#)
- [////<<keytoolKeystore,////Server Certificate Using Keytool and Keystore](#)

---

# Chapter 23. Creating and Loading the Server Certificate in pem Files

Artur Hefczyk <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2010-04-06 21:18

## Server Certificate

Server certificate is needed when you use secure socket connections - SSL/TLS.

For secure socket connection proper certificate is needed. You can either generate your own, self-signed certificate or obtain certificate from trusted third party organization.

Here are steps how to obtain certificate from a trusted organization.

## Certificate Request

Before you can obtain/generate server certificate you need to have a private key and generate certificate request which contain essential information about you (the entity requesting certificate) and the domain for which you want the certificate.

1. Generate the certificate request:

```
openssl req --nodes --new --newkey rsa:2048 --keyout yourdomain.com.key --out yo
```

You will be asked series of questions:

```
Country Name (2 letter code) [AU]:AU
State or Province Name (full name) [Some-State]:Somestate
Locality Name (eg, city) []:Your city name
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Company name
Organizational Unit Name (eg, section) []:Department or any unit
Common Name (eg, YOUR name) []:*.yourdomain.com
Email Address []:your_email_address@somedomain.com
```

```
Please enter the following -'extra' attributes
to be sent with your certificate request
```

```
A challenge password []:
```

```
An optional company name []:
```

2. The command will generate 2 files for you:

- a. yourdomain.com.csr

- b. yourdomain.com.key

## Certificate From Other Providers

There is number of certificate providers offering certificates either for free or for money. You can use any of them, however you have to be aware that sometimes certificates might not be recognized by all other XMPP servers, especially if this is a new provider. Here is an example list of providers:

- CAcert [<https://www.cacert.org/>] - free certificates with an excellent Web GUI for managing generated certificates and identities.
- StartCom [<https://www.startssl.com/>] - both free and paid certificates, class 1, 2 and 3. Very good GUI for managing certificates and identities.
- Verisign [<https://www.verisign.com/>] - very expensive certificates comparing to above provides but the provider is recognized by everybody. If you have a certificate from Verisign you can be sure it is identified as a valid certificate.
- Comodo Certificate Authority [<http://www.comodo.com/business-security/digital-certificates/ssl-certificates.php>] offers different kind of commercial certificates

To obtain certificate from such a third party authority you have to go to its Website and request the certificate using certificate request generated above. I can not provide any instructions for this as each of above providers have a different offer and a different user interface.

## Self-Signed Certificate

Another option, **highly not recommended** is to use self-signed certificate. Such a certificate won't be recognized by any entity on the Internet but your own. So if any other Jabber server allows for connections only with valid certificate your installation won't be able to connect. On the other hand self-signed certificate is a very good way for testing environment where you can create many artificial domains and you can test your service for supporting many virtual domains. You can generate as many certificates as you want and load them onto your server and play around.

Tigase, version 5.0.0 and newer automatically generates self-signed certificates for each Virtual Host configured. Alternatively such certificate can be created by hand - just execute following command using the same as above **openssl** tool:

```
openssl x509 --req --days 365 --in yourdomain.com.csr --signkey yourdomain.com.key
```

You can, of course put there any number of days you like.

## Installing/Loading Certificate To the Tigase Server

From the version **3.1.0-b802** of Tigase server installing/loading certificates is very easy. The server can load all certificates directly from **pem** files. You just need to create a separate pem file for each of your virtual domains and put the file in a directory accessible by the server. Tigase server can automatically load all **pem** files found in given directory.

### PEM File

**PEM** file is a file containing server certificate, certificate private key and certificate chain if it exists. For self-signed certificates there is no certificate chain so creating pem file is very simple:

```
cat yourdomain.com.crt yourdomain.com.key > yourdomain.com.pem
```

If the certificate is issued by third-party authority you have also to attach certificate chain, that is certificate of the authority who has generated your certificate. You normally need to obtain certificates for your chain from the authority who has generated your certificate. For example, of you have a certificate from XMPP federation you need to download StartCom root certificate [<http://www.startssl.com/certs/ca.pem>] and in-

intermediate ICA certificate [<http://www.startssl.com/certs/sub.class1.server.ca.pem>]. In such case pem file is created using following command:

```
cat yourdomain.com.crt yourdomain.com.key sub.class1.xmpp.ca.crt ca.crt > yourdomain.com.pem
```

Result file should look similar to:

```
-----BEGIN CERTIFICATE-----
MIIG/TCCBeWgAwIBAgIDA0wZMA0GCSqGSIb3DQEEBQUAMIGMMQswCQYDVQQGEwJJ
.
.
.
pSLqW/PmSLSmUNIr8yQnhy4=
-----END CERTIFICATE-----
-----BEGIN RSA PRIVATE KEY-----
WW91J3JlIGtpZGRpbmchISEKSSBkb24ndCBzaG93IHlvdSBvdXlhcHJpdmF0ZSBr
.
.
.
ZXkhISEhCkNyZWF0ZSB5b3VyIG93bIA7KSA7KSA7KQo=
-----END RSA PRIVATE KEY-----
-----BEGIN CERTIFICATE-----
MIIHyTCCBbGgAwIBAgIBATANBgkqhkiG9w0BAQUFADB9MQswCQYDVQQGEwJJTDEW
.
.
.
xV/stleh
-----END CERTIFICATE-----
```

For Tigase server and many other servers (Apache 2.x) the order is following: your domain certificate, your private key, authority issuing your certificate, root certificate.

**Note! Tigase requires full certificate chain in PEM file (described above)! Different applications may require pem file with certificates and private key in different order. So the same file may not be necessarily used by other services like Web server or e-mail server. Currently Tigase can automatically sort certificates in PEM file while loading it.**

## Tigase Server Configuration

Starting from version 5.1.0 and newer it's not needed to use external libraries nor extra configuration in `init.properties` file. With this version Tigase uses, loaded by default thus no need to configure it, following class:

```
--ssl-container-class=tigase.io.SSLContextContainer
```

Older versions require different configuration. In order to be able to load server certificates directly from **pem** files you need to have **tigase-extras** package installed in your server **libs/** directory in version at least **0.1.0**. If you use Tigase server binary package other than **mini** this library is included by default. If you haven't changed manually anything in your XML configuration file put following line in your `initial.properties` file:

```
--ssl-container-class=tigase.extras.io.PEMSSLContextContainer
```

Copy all your **pem** files with certificates into `certs/` subdirectory in Tigase server installation, stop the server, remove XML configuration file and start the server. XML configuration will be automatically

regenerated with the new SSLContainer used by all components and all certificates will be automatically loaded.

If you have changed your XML configuration file manually and you can not lost those changes you have now to manually change the existing SSLContainer class with the new one. Just replace all occurrences of the default SSLContainer - `tigase.io.SSLContextContainer` with the new - `tigase.extras.io.PEMSSLContextContainer`, copy all your **pem** files with certificates into `certs/` subdirectory in Tigase server installation and restart the server.

---

# Chapter 24. Installing StartCom Certificate in Your Linux System

Artur Hefczyk <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: <http://tigase.net> :Date: 2010-04-06 21:18

The third party authority for free XMPP server certificates is Startcom [<http://cert.startcom.org/>]. Startcom root certificate is not normally known to your system as a valid certificate and appear as a self-signed certificate.

To make it known to your system as valid you have to install it in your system.

In any case or any operating system, you have to download the certificate from the issuer web site [<http://cert.startcom.org/?lang=en&app=110>].

## Gentoo Linux

1. Copy downloaded ca.crt file to `/etc/ssl/certs/starcom_ca.crt` file.
2. Run command:

```
update-ca-certificates
```

All done. To test it run following command:

```
openssl s_client --connect tigase.org:5223 --CApath -/etc/ssl/certs
```

Scroll the output up and look for something like:

```
verify return:1
```

Which means certificate verification was successful. If you find however:

```
verify return:0
```

Look one line up for an error message which may look like this:

```
verify error:num=19:self signed certificate in certificate chain
```

Which means the root certificate is still not recognized in your system.

---

# Chapter 25. Server Certificate Using Keytool and Keystore

Bartosz Malkowski <bmalkowski@tigase.pl [mailto:bmalkowski@tigase.pl]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2010-04-06 21:18

To allow secure connections through SSL or TLS channel you need SSL certificate.

The main purpose of SSL certificate is to provide connecting entity with a proof of identity of your server. Significant role in proving identity of your server plays trusted third party - usually the issuer of the certificate.

Certificate issued by trusted third party usually cost you a money. You can also use self signed certificate which works as well but gives authentication warning on client software at the connecting time.

Tigase server binary package and sources repository contain so called "dummy" certificate which doesn't refer to any real site name. This certificate is temporary. Should be used only for initial testing of your server. It should be replaced with real one as soon as possible. By real certificate I mean either self signed certificate or issued by trusted third party organization.

Here are instructions how to install real certificate for Tigase server.

*Please note! You have to use **keytool** program included in JDK-1.6 or later version. The utility included in earlier versions can not import third party signed certificates correctly.*

## Self Signed Certificate

If you don't have third party signed certificate you should generate self-signed certificate.

Some clients don't works correctly with DSA keys, so we need to use RSA algorithm. To generate private and public keypair you should use keytool:

```
keytool --genkey --alias yourdomain --keystore rsa-keystore \  
--keyalg RSA --sigalg MD5withRSA
```

Where yourdomain is a domain part of *JIDs* on your *Jabber/XMPP* server. If you want to have TLS support for virtual domains you have to create certificate for each virtual domain. If you have just one domain or for some reason you have to use one certificate for all domains use default as an alias.

Now, enter the secret password to protect keystore:

```
Enter keystore password: 123456
```

The keytool asks several questions about certificate data. First question is very important! You must enter a hostname of your XMPP server!!

```
What is your first and last name?  
[Unknown]: jabber.myserver.org  
What is the name of your organizational unit?  
[Unknown]:  
What is the name of your organization?  
[Unknown]:  
What is the name of your City or Locality?
```



```
blreotVHqUpK9Wv42JlTDd0UmqEvo/kCwbCeK1njh4CHW4hebZzIaGTF1P/9AgMBAAEwDQYJKoZI  
hvcNAQEEBQADgYEAQqRPdkbc/pdDcPIWOTHn2XPP0gitPkXq89ZM1mb0Pib1OISj9ekjqhEz0UA  
cI6g1XtttpY6hKi6Gg+mRbwiHNvbkDLame2UIcVJ1wBtowYeOcV1CcLn1j91ScMKNhfd5ebQL+be  
tWwRjX3ep+80kF/NdVkc7htGOhLebopp8SQ=  
-----END CERTIFICATE-----
```

You should save the Certificate to disk file.

If you already have third-party signed certificate you have to import it with **keytool** program to server certificate storage.

*Note! You must have a root CA certificate!!! You can download the cert from CA (ie.: root.crt) and import:*

```
keytool --import --keystore rsa-keystore --file root.crt \  
--alias root
```

Last step is import Certificate to your keystore:

```
keytool --import --alias yourdomain --keystore rsa-keystore \  
--file your-certificate.cer
```

Where yourdomain is a domain name for which this certificate has been generated. If you need support for multiple virtual domains you have to import certificate for each domain separately and assign proper alias to certificate. If you have just one domain or for some reason you have to use one certificate for all domains use default as an alias.

It's also good to import root CA certificate to this keystore. You must find it on CA website.

```
keytool --import --keystore rsa-keystore --file rootCA.cer
```

It may also happen that you have generated certreq using openssl for use in other services like WWW. In such case you may have your private key and certificate in separate files. Let's say private key is in ssl.key file and certificate is in ssl.crt file. To put them together use following command:

```
openssl pkcs12 --export --inkey ssl.key --in ssl.crt \  
--out mycert.pfx --name "default"
```

And now you can load certificate with private key to your keystore:

```
keytool --importkeystore --srckeystore mycert.pfx \  
--srcstoretype pkcs12 --destkeystore rsa-keystore \  
--srcalias default --destalias yourdomain \  
--destkeypass your_keystore_pass
```

**Note!** Please note *-destkeypass* parametr. Your keys password must be the same as keystore password. Otherwise it won't work.

Now you have to copy file rsa-keystore to directory certs/ in your tigase server installation. The file could also be installed in different location but then corrections to config file are required. Refer to configuration documentation for details.

The final note - don't forget to modify tigase server configuration file and set proper password for your keystore.

---

# Chapter 26. Tigase and PyMSN-t Transport

Artur Hefczyk <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2010-04-06 21:18

Any Jabber server and any transport connect with each other usually through external component protocol (XEP-0114 [http://www.xmpp.org/extensions/xep-0114.html]). So all you need to do is to correctly prepare configuration for this protocol on both sides.

*Continue reading to learn how to setup Tigase and PyMSN for working together...*

There are a few basic parameters to set for this protocol:

- **PORT number** - this is standard thing for any TCP/IP connection. Usually the port number should be above 1024 and for PyMSN-t transport it is usually **5347**.
- **IP address** - again, standard thing for any TCP/IP connection. If both applications - Jabber server and transport run on the same machine the IP address should be **127.0.0.1**.
- **SECRET** - this is kind of connection password. Transport connects to the Jabber server and authenticates itself using this password. So no other, unauthorised transport can connect to the Jabber server. For our guide let the password be just **secret**.
- **Transport ID** - is an ID in Jabber network. Let's say we want to setup transport for **MSN** for the server **tigase.org**. Transport ID can be: **msn.tigase.org**. It could be also: **anything.tigase.org** but this name while still valid would be confusing for users and my suggestion is to avoid confusing names. **Note!** Transport ID should resolve to correct IP address. For your tests you can add the ID to `/etc/hosts` file.

Here is side by side configuration for both applications: PyMSN-t and Tigase to make them work together. I have setup both services on my laptop which hostname is **test-d**. To make sure both **test-d** and **msn.test-d** resolve to correct IP address I am adding entry to `/etc/hosts` file:

```
## In your case the IP address should be probably different.
192.168.0.13    test-d                msn.test-d
```

Tigase server connects to MySQL database (or built-in XMLBD for simpler configuration variant).

I am not going to setup **PyMSN-t** to run in background as a system service. This is specific to the system you use and is covered in transport documentation and you operating system. Most of systems have own scripts to start services so I would recommend to use them. Here we just run it in foreground with full logging switched on to the console to make it easier track what happens.

## PyMSN-t - `/etc/jabber/pymsn-t.xml` file

```
<pymsnt>
  <!-- The JabberID of the transport --->
  <jid>msn.test-d</jid>
  <!-- The public IP or DNS name of the machine
       the transport is running on --->
  <host>test-d</host>
```

```
<!-- The location of the PID file, relative
to the PyMSNT directory --->
<pid>/var/run/jabber/pymnsn-t.pid</pid>
<!-- If set, the transport will background
itself when run, we don't want to do this right
now. --->
<!-- <background/> --->
<!-- The IP address of the main Jabber server
to connect to --->
<mainServer>127.0.0.1</mainServer>
<!-- The TCP port to connect to the Jabber
server on (this is the default for Jabberd2) --->
<port>5347</port>
<!-- The authentication token to use when
connecting to the Jabber server --->
<secret>secret</secret>
<lang>en</lang>
<website>http://test-d/</website>
<allowRegister/>
<getAllAvatars/>
<!-- Please give the port to listen for Jabber
socks5 transfers on. Note the standard port number
set here is <strong>8010</strong>. This port
however is in use on my machine so this is why
I had to set it to different value.-->
<ftJabberPort>8014</ftJabberPort>
<admins>
  <jid>tus@test-d</jid>
</admins>
<!-- The logging level
0 --> No logging
1 --> Log tracebacks
2 --> Log tracebacks, warnings and errors
3 --> Log everything --->
<debugLevel>3</debugLevel>
<!-- The file to log to. Leave this disabled
for stdout --->
<!-- <debugFile>debug.log</debugFile> --->
</pymnsnt>
```

## PyMSN-t - run command

```
python -/usr/lib/python2.4/site-packages/pymnsn-t/pymnsn-t.py \
--c -/etc/jabber/pymnsn-t.xml
```

**Note!** the full path to PyMSN-t config file is important.

## PyMSN-t - expected output

```
A few last lines should look like:
[2007-05-07 13:00:39] Starting factory
<twisted.xish.xmlstream.XmlStreamFactory
instance at 0xb7ce80ac>
```

```
[2007-05-07 13:00:39] <twisted.internet.tcp.Connector
instance at 0xb7ceb24c> will retry in 2 seconds
[2007-05-07 13:00:39] Stopping factory
<twisted.xish.xmlstream.XmlStreamFactory
instance at 0xb7ce80ac>
[2007-05-07 13:00:41] Starting factory
<twisted.xish.xmlstream.XmlStreamFactory
instance at 0xb7ce80ac>
[2007-05-07 13:00:41] <twisted.internet.tcp.Connector
instance at 0xb7ceb24c> will retry in 5 seconds
[2007-05-07 13:00:41] Stopping factory
<twisted.xish.xmlstream.XmlStreamFactory
instance at 0xb7ce80ac>
[2007-05-07 13:00:46] Starting factory
<twisted.xish.xmlstream.XmlStreamFactory
instance at 0xb7ce80ac>
[2007-05-07 13:00:46] <twisted.internet.tcp.Connector
instance at 0xb7ceb24c> will retry in 15 seconds
[2007-05-07 13:00:46] Stopping factory
<twisted.xish.xmlstream.XmlStreamFactory
instance at 0xb7ce80ac>
```

And PyMSN should continue to print such lines until it successfully connects to the Tigase server. When it happens following lines should be printed:

```
[2007-05-07 13:29:04] Starting factory
<twisted.xish.xmlstream.XmlStreamFactory instance at 0xb7cf00ac>
[2007-05-07 13:29:04] INFO --: --: --: componentConnected
--: PyTransport --: {'xmlstream': <twisted.xish.xmlstream.XmlStream
instance at 0xb7d0feac>, -'self': -'instance'}
```

## Tigase - etc/tigase.conf file

You may consider to remove 2 last lines from TIGASE\_OPTIONS variable to not use MySQL for now. Tigase will then use internal XMLDB which doesn't need any special setup. (Just remember to leave closing double quotes...)

```
ENC="-Dfile.encoding=UTF-8 --Dsun.jnu.encoding=UTF-8"
DRV="-Djdbc.drivers=com.mysql.jdbc.Driver"
CLASSPATH="${CLASSPATH}:libs/jdbc-mysql.jar"
JAVA_OPTIONS="${ENC} ${DRV} --server --Xms100M --Xmx100M -"
TIGASE_CONFIG="etc/tigase-mysql.xml"
## All TIGASE_OPTIONS settings must be in single line
## They are split to make them more readable
TIGASE_OPTIONS="--gen-config-all ---admins \"tus@test-d\"
---virt-hosts test-d,localhost ---debug server
---ext-comp \"test-d,msn.test-d,5347,secret,plain,accept\"
---user-db mysql ---user-db-uri
\"jdbc:mysql://localhost/tigase?user=tigase&password=myspass\" -"
```

## Tigase - run command

```
./bin/tigase.sh start etc/tigase.conf
```

## Tigase - expected output

To see the log output from Tigase server execute following command:

```
tail --f logs/tigase-console.log
```

After transport connects to Tigase server you should see lines like:

```
2007-05-07 12:29:05
  ComponentConnectionManager.processHandshake() FINE:
  Connected to: msn.test-d
2007-05-07 12:29:05
  ComponentConnectionManager.updateServiceDiscovery()
  FINEST: Modifying service-discovery info:
  <item name="XEP-0114 connected"
  jid="msn.test-d"/>
```

**Note!** There was a bug in **jabber:iq:register** plugin which caused problems with registering account in transport. Please use build 432 or later.

---

# Chapter 27. Two or More SessionManagers

Artur Hefczyc <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2010-04-06 21:18

In the most cases you use just one SessionManager object for your Tigase server installation. A single SM can handle multiple virtual domains with separate SSL certificates for each domain.

Sometimes, however you need very different configuration for each domain. For example you wish to use a separate database for a selected domain or you need a different set of plugins for each domain. For one domain you might want to allow user registration via XMPP and for another you might want to disable this feature. In such a case you need to load more than one session manager.

This is generally not a problem. You just need to add another component in the configuration and adjust default settings.

The question is now how Tigase server knows to which session manager it has to forward packets received from the network. Mind, there is only one component responsible for handling client connections. So it needs to know somehow which session manager is receiver for certain packet. Of course you set domain names in session manager too. But that is not enough. Tigase server supports cluster mode configuration where session manager can be running on a separate machine. So packet routings rules are not so simple to look at the domain name only. Therefore client connection manager (**c2s**) must know where is located the session manager responsible for handling the packet received from the network.

To solve the problem **routings** concept has been introduced. You can define packet routings based on the domain name set during XMPP stream initialization. Each time **c2s** component receives packet from the network it tries to resolve destination component for the packet based on the current routings table. If you look in you server XML configuration file and search for **c2s** configuration section you can find routings node. Default configuration for routings table is quite simple. Just a single regular expression:

```
<node name="routings">
  <map>
    <entry key="." type="String" value="sess-man%40tigase.org"/>
    <entry key="multi-mode" type="Boolean" value="true"/>
  </map>
</node>
```

As you can see this routings table forwards all packets to a single destination - session manager located on the **tigase.org** server.

Let's say we have now two session managers each of them is responsible for a separate domain. **sm1@tigase.org** handles requests for **tigase.org** and **sm2@tigase.net** handles requests for domain **tigase.net**. So let's modify our default configuration to properly spread the traffic between these two sessiona managers:

```
<node name="routings">
  <map>
    <entry key="tigase.org" type="String" value="sm1%40tigase.org"/>
    <entry key="tigase.net" type="String" value="sm2%40tigase.net"/>
    <entry key="multi-mode" type="Boolean" value="true"/>
  </map>
</node>
```

Please remember that a key is a regular expression in Java style: Pattern.html [<http://java.sun.com/j2se/1.4.2/docs/api/java/util/regex/Pattern.html>]. You can match more than a single domain with the key, for example: *tigase.+* to match all domains starting with **tigase**. The expression, however won't match domain: **xmpp.tigase.org**. To match this domain the expression would need to be: *.+tigase.+*.

---

# Chapter 28. Database Preparation

Artur Hefczyc <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2012-07-15 06:42

Tigase uses generally the same DB schema and the same set of stored procedures and functions on every database. However, the schema creation scripts and code for stored procedures is different for each database. Therefore the manual process to prepare database is different for each of them.

Of course the simplest and easiest way to prepare database is to use Tigase installer which automates the whole process. Sometimes it is not possible however. Here we provide a set of guides, describing initialization and preparation process for each supported database.

- [Prepare the MySQL Database for the Tigase Server](#)
- [Hashed User Passwords in Database](#)
- [Prepare the Derby Database for the Tigase Server](#)
- [Prepare the MS SQL Server Database for the Tigase Server](#)
- [Prepare the PostgreSQL Database for the Tigase Server](#)

---

# Chapter 29. Prepare the MySQL Database for the Tigase Server

Artur Hefczyk <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2010-04-06 21:18

This guide describes how to prepare the MySQL database for connecting the Tigase server to it.

## Basic Setup

The MySQL database can be prepared in many ways. Most of Linux distributions contain tools which allow you to go through all steps from the shell command line. To make sure, however it works on all platforms the same way I show first how to do it under MySQL command line client.

## Configuring from MySQL command line tool

Run the MySQL command line client in either Linux or MS Windows environment and enter following instructions:

1. Create the database for the Tigase server:

```
mysql> create database tigasedb;
```

2. Add the `tigase_user` user and grant him access to the `tigasedb` database. Depending on how you plan to connect to the database (locally or over the network) use one of following commands or all if you are not sure:

**Grant access to `tigase_user` connecting from any network address.**

```
mysql> GRANT ALL ON tigasedb.* TO tigase_user@'%'  
IDENTIFIED BY -'tigase_passwd';
```

**Grant access to `tigase_user` connecting from localhost.**

```
mysql> GRANT ALL ON tigasedb.* TO tigase_user@'localhost'  
IDENTIFIED BY -'tigase_passwd';
```

**Grant access to `tigase_user` connecting from local machine only.**

```
mysql> GRANT ALL ON tigasedb.* TO tigase_user  
IDENTIFIED BY -'tigase_passwd';
```

For the Tigase server version 4.x additional permissions must be granted for the database user:

```
mysql> GRANT SELECT, INSERT, UPDATE ON mysql.proc TO -'tigase_user'@'localhost';  
mysql> GRANT SELECT, INSERT, UPDATE ON mysql.proc TO -'tigase_user'@'%';  
mysql> GRANT SELECT, INSERT, UPDATE ON mysql.proc TO -'tigase_user';
```

And now you can update user permission changes in the database:

```
mysql> FLUSH PRIVILEGES;
```

3. Load database schema [<http://server.tigase.org/browser/trunk/database/mysql-schema.sql>] to initialize the Tigase server database space. First, switch to the database you have just created:

```
mysql> use tigasedb;
```

Assuming you run the mysql client in Linux from the Tigase installation directory. If you run the Tigase server all versions below 4.0:

```
mysql> source database/mysql-schema.sql;
```

For the Tigase server version 4.x you have to use proper schema version:

```
mysql> source database/mysql-schema-4.sql;
```

On Windows you have probably to enter the full path:

```
mysql> source c:/Program Files/Tigase/database/mysql-schema.sql;
```

The initialization schema file should be also available locally in database/ directory of your Tigase installation.

## Configuring From the Linux Shell Command Line

Follow steps below to prepare the MySQL database:

1. Create the database space for the Tigase server:

```
mysqladmin --p create tigasedb
```

1. Add the `tigase_user` user and grant him access to the `tigasedb` database. Depending on how you plan to connect to the database (locally or over the network) use one of following commands or all if you are not sure: **Grant access to `tigase_user` connecting from any network address.**

```
echo -"GRANT ALL ON tigasedb.* TO tigase_user@%' \
      IDENTIFIED BY -'tigase_passwd'; \
      FLUSH PRIVILEGES;" -| mysql --u root --pdbpass mysql
```

**Grant access to `tigase_user` connecting from localhost.**

```
echo -"GRANT ALL ON tigasedb.* TO tigase_user@'localhost' \
      IDENTIFIED BY -'tigase_passwd'; \
      FLUSH PRIVILEGES;" -| mysql --u root --pdbpass mysql
```

**Grant access to `tigase_user` connecting from local machine only.**

```
echo -"GRANT ALL ON tigasedb.* TO tigase_user \
      IDENTIFIED BY -'tigase_passwd'; \
      FLUSH PRIVILEGES;" -| mysql --u root --pdbpass mysql
```

1. Load database schema [<http://server.tigase.org/browser/trunk/database/mysql-schema.sql>] to initialize the Tigase server (version below 4.0) database space:

```
mysql --u dbuser --p tigasedb < mysql-schema.sql
```

For the Tigase server version 4.0 and later:

```
mysql --u dbuser --p tigasedb < mysql-schema-4.sql
```

The initialization schema file should be also available locally in database/ directory of your Tigase installation.

## Configuring MySQL for UTF-8 Support

In the my.conf put following lines:

```
[mysql]
default-character-SET=utf8

[client]
default-character-SET=utf8

[mysqld]
init_connect='SET collation_connection = utf8_general_ci; SET NAMES utf8;'
character-set-server=utf8
default-character-SET=utf8
collation-server=utf8_general_ci
skip-character-set-client-handshake
```

Then connect to the database and from the command line shell check settings:

```
SHOW VARIABLES LIKE -'character_set_database';
SHOW VARIABLES LIKE -'character_set_client';
```

If any of these shows something else then 'utf8' then you have to correct it:

```
ALTER DATABASE tigasedb DEFAULT CHARACTER SET utf8;
```

You can now also test your database installation if it accepts UTF-8 data. Best way is just to create an account with UTF-8 characters:

```
call TigAddUserPlainPw('#ó#w@some.domain.com', -'#ó#w');
```

And then check of the account has been created:

```
SELECT * FROM tig_users WHERE user_id = -'#ó#w@some.domain.com';
```

If the last command gives you no results it means there is still something wrong with settings. You might also check you shell settings to make sure your command line shell supports UTF-8 characters and passes them correctly to MySQL:

```
export LANG=en_US.UTF-8
export LOCALE=UTF-8
export LESSCHARSET='utf-8'
```

It seems to me that MySQL 5.0.x also needs an extra parameters in the connection string: '&useUnicode=true&characterEncoding=UTF-8' while MySQL 5.1.x seems to not need it but it doesn't hurt to have it for both versions. You have to edit 'etc/init.properties' file and append this to the database connection string.

For MySQL 5.1.x, however, you need also updated code for all database stored procedures and functions used by the Tigase. They are updated for Tigase version 4.4.x and up, for the time being if you use older version of the Tigase server you can reload stored procedures using the file from SVN.

## Other MySQL Setting Worth Considering

There is a number of other options useful, especially for a performance reasons. Please note, you have to review them as some of them may impact data reliability and are useful for performance or load tests installations only.

```
# InnoDB seems to be a better choice
# so lets make it a default DB engine
default-storage-engine = innodb
```

Some the general MySQL settings which mainly affect performance:

```
key_buffer = 64M
max_allowed_packet = 32M
sort_buffer_size = 64M
net_buffer_length = 64K
read_buffer_size = 16M
read_rnd_buffer_size = 16M
thread_stack = 192K
thread_cache_size = 8
query_cache_limit = 10M
query_cache_size = 64M
```

InnoDB specific settings:

```
# Keep data in a separate file for each table
innodb_file_per_table = 1
# Allocate memory for data buffers
innodb_buffer_pool_size = 1000M
innodb_additional_mem_pool_size = 100M
# A location of the MySQL database
innodb_data_home_dir = -/home/databases/mysql/
innodb_log_group_home_dir = -/home/databases/mysql/
# The main thing here is the -'autoextend' property
# without it your data file may reach maximum size and
# no more records can be added to the table.
innodb_data_file_path = ibdata1:10M:autoextend
innodb_log_file_size = 10M
innodb_log_buffer_size = 32M
# Some other performance affecting settings
innodb_flush_log_at_trx_commit = 2
innodb_lock_wait_timeout = 50
innodb_thread_concurrency = 16
```

I am certainly not a database expert nor MySQL expert and I do not pretend to be one. So any comments or suggestions you may have are very welcome and appreciated.

---

# Chapter 30. Hashed User Passwords in Database

Artur Hefczyc <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2012-10-09 03:13

By default user passwords are stored in plain-text in the Tigase's database. However, there is an easy way to have them encoded in either one of already supported ways or to even add a new encoding algorithm on your own.

The reason to store passwords in plain-text format in the database is to make it possible to avoid plain-text password authentication mechanism. At the moment you cannot have hashed passwords in the database and non-plain-text password authentication. On the other hand, the connection between the server and the client is almost always secured by SSL/TLS so maybe the plain-text password authentication method is less of a problem than storing plain-text passwords in the database.

Nevertheless, it is simple enough to adjust this in Tigase's database and we will add an option in the Tigase installer to allow you to make the decision at installation time.

## Shortcut

Connect to your database from a command line and execute following statement for MySQL database:

```
call TigPutDBProperty('password-encoding', -'encoding-mode');
```

Where encoding mode is one of the following:

- **MD5-PASSWORD** the database stores MD5 hash code from the user's password.
- **MD5-USERID-PASSWORD** the database stores MD5 hash code from concatenated user's bare JID and password.
- **MD5-USERNAME-PASSWORD** the database stores MD5 hash code from concatenated user's name (localpart) and password.

For example:

```
call TigPutDBProperty('password-encoding', -'MD5-PASSWORD');
```

## Full Route

The way passwords are stored in the DB is controlled by Tigase database schema property. Properties in the database schema can be set by a stored procedure called: `TigPutDBProperty(key, value)`. Properties from the DB schema can be retrieved using another stored function called: `TigGetDBProperty(key)`.

The simplest way to call them is via command-line interface to the database.

For the purpose of this guide let's say we have MySQL database and a test account: **test@example.com** with password **test77**.

By default, most of DB actions for Tigase, are performed using stored procedures. This includes user authentication. So, the first thing to do is to make sure the stored procedures are working correctly.

## Create a Test User Account

To add a new user account we use a stored procedure: `TigAddUserPlainPw(bareJid, password)`. As you can see there is this strange appendix to the procedure name: **PlainPw**. It means the procedure accepts plain password regardless how it is stored in the database. So it is safe and easy to use either for plain-text passwords or hashed in the DB. There are also versions of procedures without this appendix but they are sensitive on the data format and always have to pass password in the exact format it is stored in the database.

So, let's add a new user account:

```
call TigAddUserPlainPw('test@example.com', -'test77');
```

If the result was 'Query OK', then it means the user account has been successfully created.

## Test User Authentication

We can now test user authentication:

```
call TigUserLoginPlainPw('test@example.com', -'test77');
```

If authentication was successful the result looks like this:

```
+-----+
| user_id          -|
+-----+
| -'test@example.com' -|
+-----+
1 row in set (0.01 sec)
```

Query OK, 0 rows affected (0.01 sec)

If authentication was unsuccessful the result looks like this:

```
+-----+
| user_id -|
+-----+
|      NULL -|
+-----+
1 row in set (0.01 sec)
```

Query OK, 0 rows affected (0.01 sec)

## Password Encoding Check

`TigGetDBProperty` is a function, not a procedure in MySQL database so we have to use `select` to call it:

```
select TigGetDBProperty('password-encoding');
```

Most likely output is this:

```
+-----+
| TigGetDBProperty('password-encoding') -|
+-----+
```

```
| NULL -|
+-----+
1 row in set, 1 warning (0.00 sec)
```

Which means a default password encoding is used, that is plain-text, thus no encoding. And we can actually check this in the database directly:

```
select uid, user_id, user_pw from tig_users where user_id = -'test@example.com';
```

And expected result with plain-text password format would be:

```
+-----+-----+-----+
| uid -| user_id          -| user_pw -|
+-----+-----+-----+
| 41 -| -'test@example.com' -| test77 -|
+-----+-----+-----+
1 row in set (0.00 sec)
```

## Password Encoding Change

Now let's set password encoding to MD5 hash:

```
call TigPutDBProperty('password-encoding', -'MD5-PASSWORD');
```

'Query OK', means the password encoding has been successfully changed. Of course we changed the property only. All the existing passwords in the database are still in plain-text format. Therefore we expect that attempt to authenticate the user would fail:

```
call TigUserLoginPlainPw('test@example.com', -'test777');
+-----+
| user_id -|
+-----+
| NULL -|
+-----+
1 row in set (0.00 sec)
```

Query OK, 0 rows affected (0.00 sec)

We can fix this updating the user's password in the database:

```
call TigUpdatePasswordPlainPw('test@example.com', -'test777');
Query OK, 1 row affected (0.01 sec)
```

```
mysql> call TigUserLoginPlainPw('test@example.com', -'test777');
+-----+
| user_id          -|
+-----+
| -'test@example.com' -|
+-----+
1 row in set (0.00 sec)
```

Query OK, 0 rows affected (0.00 sec)

---

# Chapter 31. Prepare the Derby Database for the Tigase Server

Wojciech Kapcia <wojciech.kapcia@tigase.org [mailto:wojciech.kapcia@tigase.org]> v2.0, June 2014:  
Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2012-06-21 13:28

This guide describes how to prepare the Derby database for connecting the Tigase server to it.

## Basic Setup

Preparation of the Derby database is quite simple, but the following assumptions are made

- DerbyDB - Derby database name
- database/ directory contains all necessary schema files
- jars/ and libs/ directories contains Tigase and Derby binaries

## General Approach

From the main Tigase directory execute following commands (Linux and Windows accordingly)

```
java --Dij.protocol=jdbc:derby: --Dij.database="DerbyDB;create=true" --cp libs/der
```

```
java --Dij.protocol=jdbc:derby: --Dij.database="DerbyDB;create=true" --cp libs\der
```

This will create Derby database named DerbyDB in the main Tigase directory and load Tigase schema for the version 5.1.

If there is a need to create schema for any other Tigase version then please use schema relevant to the version that you intend to use:

If you run the Tigase server all versions below 4.0:

```
derby-schema-3.sql
```

For the Tigase server version 4.x and later:

```
derby-schema-4.sql
```

For the Tigase server version 5.0 and later:

```
derby-schema-5.sql
```

For the Tigase server version 5.1 and later:

```
derby-schema-5-1.sql
```

---

# Chapter 32. Prepare the MS SQL Server Database for the Tigase Server

Wojciech Kapcia <wojciech.kapcia@tigase.org [mailto:wojciech.kapcia@tigase.org]> v2.0, June 2014:  
Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2013-09-04 14:57

This guide describes how to prepare the MS SQL Server database for connecting the Tigase server to it.

## Basic Setup

It's expected that a working installation of Microsoft SQL Server is present. Following guide will describe necessary configuration option required for using MS SQL Server with Tigase XMPP Server.

## Preparing SQL Server Instance

After installation of SQL Server an instance needs to be configure to handle incoming JDBC connections. For that purpose it's required to open *SQL Server Configuration Manager*. In the left-hand side panel navigate to *SQL Server Configuration Manager*, then *SQL Server Network Configuration # Protocols for \${INSTANCE\_NAME}*. After selecting instance in the right-hand side panel select TCP/IP and open *Properties*, in the Protocol tab in General section select Yes for Enabled property. In the IP Addresses tab select Yes for Active and Enabled properties of all IP Addresses that you want SQL Server to handle. Subsequently set TCP Port property (if missing) to the default value - 1433. A restart of the instance may be required afterwards.

## Configuring using SQL Server Management Studio

In order to prepare database one can use either Wizards or execute queries directly in the Query Editor. First of all you need to establish connection to the SQL Server instance thus from Object Explorer select Connect and in the Connect to Server dialog enter administrator credentials.

## Using Wizards

- Create Login

In the left-hand side panel select Security → Logins and from context menu choose New Login, in the Wizard window enter desired Login name, select SQL Server authentication and enter desired password subsequently confirming action with OK

- Create Database

From the Object Explorer select Databases node and from context menu select New Database; in the Wizard window enter desired Database name and enter previously created Login name into Owner field; subsequently confirming action with OK.

## Using Queries

From the Object Explorer root node's context menu select New Query. In the Query windows execute following statements adjusting details to your liking:

```
USE [master]
GO

CREATE DATABASE [tigasedb];
GO

CREATE LOGIN [tigase] WITH PASSWORD=N'tigase12', DEFAULT_DATABASE=[tigasedb]
GO

ALTER AUTHORIZATION ON DATABASE::tigasedb TO tigase;
GO
```

## Import Schema

From the File menu Select Open → File (or use Ctrl+O) and then open following files:

- sqlserver-schema-5-1-schema.sql
- sqlserver-schema-5-1-sp.sql
- sqlserver-schema-5-1-props.sql

Subsequently select created database from the list of Available Databases (Ctrl+U) available on the toolbar and execute each of the opened files in the order listed above.

## Configuring from command line tool

Creation of the database and import of schema can be done from command line as well. In order to do that one needs to execute following commands (either from directory where Tigase XMPP Server is installed or paths to the schema need to be adjusted accordingly):

```
sqlcmd --S %servername% --U %root_user% --P %root_pass% --Q -"CREATE DATABASE [%da
sqlcmd --S %servername% --U %root_user% --P %root_pass% --Q -"CREATE LOGIN [%user%
sqlcmd --S %servername% --U %root_user% --P %root_pass% --d %database% --Q -"ALTER
sqlcmd --S %servername% --U %root_user% --P %root_pass% --d %database% --i databas
sqlcmd --S %servername% --U %root_user% --P %root_pass% --d %database% --i databas
sqlcmd --S %servername% --U %root_user% --P %root_pass% --d %database% --i databas
```

Above can be automatized with provided script %tigase-server%\scripts\db-create-sqlserver.cmd (note: it needs to be executed from main Tigase XMPP Server directory due to maintain correct paths):

```
$ scripts\db-create-sqlserver.cmd %database_servername% %database_name% %tigase_us
```

If no parameters are provided then defaults are used:

```
%database_servername%=localhost
%database_name%=tigasedb
%tigase_username%=tigase
%tigase_password%=tigase12
%root_username%=root
%root_password%=root
```

## Tigase configuration - init.properties

Configuration of the MS SQL Server follows general database convention. For MS SQL Support --user-db needs to be set to sqlserver:

```
--user-db=sqlserver
```

and the --user-db-uri needs to point to the configured database:

```
--user-db-uri=jdbc:[jtds:]sqlserver://db_hostname:port[;property=val]
```

where any number of additional can (and should) consist of:

- databaseName - name of the database
- user - username configured to access database
- password - password for the above username
- schema - name of the database schema
- lastUpdateCount - 'false' value causes all update counts to be returned, including those returned by server triggers

Example:

```
--user-db-uri=jdbc:sqlserver://hostname:1433;databaseName=tigasedb;user=tigase;pas
```

## JDBC: JTDS vs MS JDBC driver

Tigase XMPP Server supports two JDBC drivers intended to be used with Microsoft SQL Server - one created and provided by Microsoft itself and the alternative implementation - JTDS. Tigase is shipped with the latter in the distribution packages, however we do not recommend using JTDS with Tigase and urge users to use JDBC driver created by Microsoft. It can be downloaded from Microsoft website: JDBC Driver 4.0 for SQL Server [<http://www.microsoft.com/en-us/download/details.aspx?displaylang=en&id=11774>] and unpack archive, then copy sqljdbc\_4.0/enu/sqljdbc4.jar file to \${tigase-server}/jars directory.

Depending on the driver used --user-db-uri needs to be configured accordingly.

- Microsoft driver:

```
--user-db-uri=jdbc:sqlserver://...
```

- jDTS driver

```
--user-db-uri=jdbc:jtds:sqlserver://...
```

---

# Chapter 33. Prepare the PostgreSQL Database for the Tigase Server

Andrzej Wojcik <andrzejw@tigase.org [mailto:andrzejw@tigase.org]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2012-06-16 09:38

This guide describes how to prepare the PostgreSQL database for connecting the Tigase server to it.

## Basic Setup

The PostgreSQL database can be prepared in many ways. Below are presented two of them. Following assumptions apply to both methods:

- admin\_db\_user - database user with admin rights
- tigase\_user - database user for Tigase
- tigasedb - database for Tigase

## Configuring from PostgreSQL Command Line Tool

Run the PostgreSQL command line client and enter following instructions:

1. Add the tigase\_user:

```
psql=# create role tigase_user with login password 'tigase123';
```

2. Create the database for the Tigase server with tigase\_user as owner of database:

```
psql=# create database tigasedb owner tigase_user;
```

3. Load database schema to initialize the Tigase server from the file that corresponds to the Tigase version you want to use. First you need to switch to tigasedb.

```
psql=# \connect tigasedb
```

For the Tigase server version 5.0 and later you have to use proper schema version:

```
psql=# \i database/postgresql-schema-5.sql
```

For the Tigase server version 5.1 and later you have to use proper schema version:

```
psql=# \i database/postgresql-schema-5-1.sql
```

## Configuring From the Linux Shell Command Line

Follow steps below to prepare the PostgreSQL database:

1. Add the tigase\_user:

```
createuser --U admin_db_user --W --D --R --S --P tigase_user
```

You will be asked for credentials for admin\_db\_user and password for new database user.

Prepare the PostgreSQL  
Database for the Tigase Server

---

2. Create the database for the Tigase server with `tigase_user` as owner of database:

```
createdb --U admin_db_user --W --O tigase_user tigasedb
```

3. Load database schema to initialize the Tigase server from the file that corresponds to the Tigase version you want to use. For the **Tigase server version 5.0 and later** you have to use proper schema version:

```
psql --q --U tigase_user --W tigasedb --f database/postgresql-schema-5.sql
```

For the **Tigase server version 5.1 and later** you have to use proper schema version:

```
psql --q --U tigase_user --W tigasedb --f database/postgresql-schema-5-1.sql
```

Above commands should be executed from the main Tigase directory. The initialization schema file should be also available locally in `database/` directory of your Tigase installation.

---

# Chapter 34. Tigase Database Minor but Useful Schema Change in Version 5.1.0

Artur Hefczyc <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2012-06-05 02:38

We have recently made a simple but very useful change to the DB schema in Tigase. It does preserve backward compatibility and it is not required to change your existing schema, even if you upgrade your installation to the most recent 5.1.0 version.

However, the change is very useful to track new accounts creation, therefore this article shows how to make modifications to you schema manually.

The change is related to adding a new field: `acc_create_time` which stores exact time of user account creation. The time is recorded automatically by database when a new record in the user table is created. So there is no extra overhead or resource usage on the Tigase server.

While the new field was added I have also slightly modifies types of 2 other fields: `last_login` and `last_logout`.

Here is how it looks now:

```
-- Time the account has been created
acc_create_time timestamp DEFAULT CURRENT_TIMESTAMP,
-- Time of the last user login
last_login timestamp DEFAULT 0,
-- Time of the last user logout
last_logout timestamp DEFAULT 0,
```

You can easily update your schema with above changes using a simple SQL query. Enter the MySQL shell and copy-paste following query:

```
alter table tig_users
  modify last_login timestamp DEFAULT 0,
  modify last_logout timestamp DEFAULT 0,
  add acc_create_time timestamp DEFAULT CURRENT_TIMESTAMP;
```

Please note, after executing query above the column `acc_create_time` will have meaningless value. To put some meaning for existing records I just copy `last_login` value to `acc_create_time`:

```
update tig_users set acc_create_time = last_login;
```

---

# Chapter 35. Tigase Load Balancing

Wojciech Kapcia <wojciech.kapcia@tigase.org [mailto:wojciech.kapcia@tigase.org]> v2.0, June 2014:  
Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2013-06-10 15:49

Starting with version 5.2.0 Tigase introduces a load balancing functionality allowing users to be redirected to the most suitable cluster node. Functionality relies on a see-other-host XMPP stream error message. Basic principle behind the mechanism is that user will get redirect if the host returned by the implementation differ from the host to which user currently tries to connect. It is required that the user JID to be known for the redirection to work correctly.

## Available Implementations

Tigase implementation is, as usual, extensible and allows for different, pluggable redirection strategies that implement *SeeOtherHostIfc* interface.

Currently there are three strategies available:

- *SeeOtherHost* - most basic implementation returning either single host configured in init.properties file or name of the current host;
- *SeeOtherHostHashed* (default) - default implementation for cluster environment of *SeeOtherHostIfc* returning redirect host based on the hash value of the user's JID; list of the available nodes from which selection would be made is by default composed and reflects all connected nodes, alternatively hosts list can be configured in the init.properties;
- *SeeOtherHostDB* - extended implementation of *SeeOtherHost* using redirect information from database in the form of pairs *user\_id* and *node\_id* to which given user should be redirected.

## Configuration Options

Most basic configuration is related to the choice of actual redirection implementation:

```
--cm-see-other-host=
```

Possible values are:

- `tigase.server.xmppclient.SeeOtherHost`
- `tigase.server.xmppclient.SeeOtherHostHashed`
- `tigase.server.xmppclient.SeeOtherHostDB`
- `none` - disables redirection

All the remaining options are configured on per-connection-manager basis thus all options need to be prefixed with corresponding connection manager ID, i.e. `c2s`, `bosh` or `ws`; we will use `c2s` in the examples:

- `c2s/cm-see-other-host/default-host=host1;host2;host3` - a semicolon separated list of hosts to be used for redirection;
- `c2s/cm-see-other-host/active=OPEN;LOGIN`  
a semicolon separated list of phases in which redirection should be active;

- currently possible values are:
- *OPEN* which enables redirection during opening of the XMPP stream;
- *LOGIN* which enables redirection upon authenticating user session;
- by default redirection is currently enabled only in the *OPEN* phase.

For SeeOtherHostDB implementation there are additional options:

- `c2s/cm-see-other-host/db-url` - a JDBC connection URI which should be used to query redirect information; if not configured `--user-db-uri` will be used;
- `c2s/cm-see-other-host/get-host-query` - a SQL query which should return redirection hostname;
- `c2s/cm-see-other-host/get-all-data-query` - a SQL helper query which should return all redirection data from database;
- `c2s/cm-see-other-host/get-all-query-timeout` - allows to set timeout for executed queries.

---

# Chapter 36. StanzaSender

Artur Hefczyc <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2010-04-06 21:18

This is a component which make it easier to integrate Jabber/XMPP server with other, third-party tools.

It simply allows you to send stanzas from your application without implementing any Jabber/XMPP specific code. The component regularly reads specified data source for XMPP packets to send. The data source can be an SQL database, directory on your filesystem or anything you might want.

If you have Web application for example which you want to send notifications of any event to selected users you can install StanzaSender [<http://server.tigase.org/browser/trunk/src/main/java/tigase/server/ssender/StanzaSender.java>] component on your Tigase server. It will help you to easily distribute your messages to end-users.

## How it Works

The module itself doesn't do anything. It just schedules tasks and sends stanzas which come from... it doesn't know. To do actual work of retrieving stanzas from data source the component uses **tasks**.

In theory the task can retrieve XMPP packets for sending from any location or may just generate stanzas on its own.

In practise there are 2 **tasks** already implemented and ready to use. You can treat them as a sample code for implementation of your own tasks customised for your specific needs or you can just use these tasks as they are.

The tasks which are available are:

- FileTask retrieving stanzas from directory in file system.
- JDBCTask retrieving stanzas from SQL database.

## FileTask

FileTask implements tasks for cyclic retrieving stanzas from a directory and sending them to the StanzaHandler object.

It looks for any new stanza to send. Any single file can contain only single stanza to send and any entry in database table can also contain only single stanza to send. File on hard disk and record in database is deleted after it is read.

Any file in given directory is treated the same way - Tigase assumes it contains valid XML data with XMPP stanza to send. You can however set in configuration, using wildchars which files contain stanzas. All stanzas must contain complete data including correct "from" and "to" attributes.

By default it looks for \*.stanza files in /var/spool/jabber/ folder but you can specify different directory name in initialization string. Sample initialization strings:

- /var/spool/jabber/\*.stanza
- /var/spool/jabber/\*

The last is equal to:

- /var/spool/jabber/

Note the last forward slash '/' is required in such case if the last element of the path is a directory.

**Please note! Tigase must have writing permissions for this directory, otherwise it may not function properly.**

## JDBCTask

JDBCTask implements tasks for cyclic retrieving stanzas from database and sending them to the Stanza-Handler object.

Database table format:

- **id** - numerical unique record identifier.
- **stanza** - text field containing valid XML data with XMPP stanza to send.

Any record in this table is treated the same way - Tigase assumes it contains valid XML data with XMPP stanza to send. No other data are allowed in this table. All stanzas must be complete including correct "from" and "to" attributes.

By default it looks for stanzas in xmpp\_stanza table but you can specify different table name in connection string. Sample connection string:

```
jdbc:mysql://localhost/tigasedb?user=tigase&password=pass&table=xmpp_stanza
```

Please note the last parameter which is specific to JDBCTask. You can specify the table name which stores stanzas for sending. If omitted default value is: xmpp\_stanza.

## Configuration

It is Tigase component so the configuration is similar to configuration of all other components. The simplest way to get the settings for **StanzaSender** is by generating configuration with all possible components. To do this you have to run Tigase server with --gen-config-all parameter set.

By default this component name is **sSend** and here is a content of the configuration file for **StanzaSender**:

It is one of msg-receivers:

```
<entry type="String[]" key="id-names">
  ...
  <item value="sSend"/>
</entry>
```

To activate the component and specify class name for it following entries has been added:

```
<entry value="true" type="Boolean" key="sSend.active"/>
<entry value="tigase.server.sSender.StanzaSender" type="String" key="sSend.class"/>
```

And the main settings section for the component:

```
<component name="sSend">
  <map>
```

```
<entry value="10" type="Long" key="default-interval"/>
<entry value="1000" type="Integer" key="max-queue-size"/>
<entry type="String[]" key="stanza-listeners">
  <item value="jdbc"/>
  <item value="file"/>
</entry>
</map>
<node name="file">
  <map>
    <entry value="true" type="Boolean" key="active"/>
    <entry value="tigase.server.ssender.FileTask" type="String" key="class-name"/>
    <entry value="/var/spool/jabber/*.stanza" type="String" key="init-string"/>
    <entry value="10" type="Long" key="interval"/>
  </map>
</node>
<node name="jdbc">
  <map>
    <entry value="true" type="Boolean" key="active"/>
    <entry value="tigase.server.ssender.JDBCTask" type="String" key="class-name"/>
    <entry value="jdbc:mysql://localhost/tigase?user=tigase&
      password=mypass&table=xmpp_stanza"
      type="String" key="init-string"/>
    <entry value="10" type="Long" key="interval"/>
  </map>
</node>
</component>
```

I think most of parameters should be pretty clear but some may need a little explanation. General Stanza-Sender parameters:

- **default-interval** number which specifies in seconds how often should the task look in data source for new packets to send.
- **max-queue-size** is a number which specifies internal packets queue size. This is used to prevent the component from consume all the memory for data in case the component can not process them.
- **stanza-listeners** is a list of task names to load. Each task can read XMPP packets to send from different data source. You can load as many listeners (tasks) as you need. Each task must read stanzas from different data source.

Each task has own, separate parameters list. For each task from the stanza-listeners list there is a separate section with parameters for each task:

- **active** boolean switch allowing you to turn on/off the task without removing configuration completely.
- **class-name** Java class name which implements the task. This class must extend `tigase.server.ssender.SenderTask` and it is loaded at runtime.
- **init-string** is kind of data source connection string. For database it is just database connection string, for file system this is just a directory name. It may be even different for different tasks. The 2 tasks already implemented have some specific features: `FileTask` allows you to use wild-chars in directory/ file name specification and `JDBCTask` allows you to specify additional parameter at the end of JDBC connection string - database table name. For specific examples look at above config sections.
- **interval** is a number which allows you to specify different interval in seconds for checking data source for each task.

---

# Chapter 37. Debugging Tigase

Artur Hefczyc <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2010-04-06 21:18

If something goes wrong and you can't find out why it is not working as you expect you might want more detailed debugging options switched on.

Tigase is a Java application and it uses Java logging library this gives you flexibility to switch logging on for selected java package or even for a single Java class.

Logs files are stored in logs/ directory. `tigase-console.log` keeps all the data but only basic logs. `tigase.log.N` files keep all the detailed logging entries. So this is the place where you should look in case of problems.

## The easy way - `init.properties` file

The easiest way to change logging for the Tigase package is modifying in `init.properties` following line:

```
--debug=server
```

The line above says: "*Switch on ALL debug messages for packet: `tigase.server`*". The `tigase.server` packet keeps all component's classes. So it allows you to monitor what is going on in each component. What packets it receives and what it is sending out.

Usually people want to see what is going on the network level. That is what has been sent and what has been received by the server - the actual character data. The class which would print all received and sent character data is: `tigase.xmpp.XMPPIOService`. To enable all debugging info for this class you have to modify the debug line:

```
--debug=xmpp.XMPPIOService
```

*Note, you skip the `tigase` part.*

You can also have debugging switched on for many packages/classes at the same time:

```
--debug=server , xmpp.XMPPIOService
```

Other packages you might be interested in are:

- `tigase.io` and `tigase.net` which can print out what is going on a very low level network level including TLS/SSL stuff.
- `tigase.xml` would print the XML parser debugging data.
- `tigase.cluster` would print all the clustering related stuff. So if you have clustered installation you might be interested in debug settings:

```
--debug=server , cluster
```

- `tigase.xmpp.impl` would print logs from all plugins loaded to the Tigase server.

and so on...

This method, however has 2 main disadvantages:

1. You have to remove your XML config file and regenerate it which might be inconvenient. (only applicable to versions before 5.0.0)
2. You can't set logging this way for classes and packages other than `tigase.package`. And this might be a problem if you include your own code in and load it into the server.

To enable logging for your own packages from packages different than `tigase`, you have to use another option which has been made available for this:

```
--debug-packages = your.com.package
```

You can also specify more parameters for the Tigase logging mechanisms like the file size, number of file rotated and location where all Tigase logs are stored. You have to include following lines in the `init.properties` file:

```
basic-conf/logging/java.util.logging.FileHandler.limit=100000000
basic-conf/logging/java.util.logging.FileHandler.count=20
basic-conf/logging/java.util.logging.FileHandler.pattern=/var/log/tigase/tigase.lo
```

## The more difficult but more powerful - tigase.xml file (only applicable to versions be- fore 5.0.0)

If you want to modify debugging settings without regenerating the whole XML config file you can modify it yourself manually and add debug entries. This must be done very carefully or you break the XML file and configuration won't work.

Anyway, Open the XML config file with some good text editor (or XML editor) and find the line:

```
<node name="logging">
```

Below is a long list of all logging settings. One of them is:

```
<entry value="INFO" type="String" key=".level"/>
```

Which says: INFO debug level for all the code. This level gives you very little debugging information. Therefore for example your `init.properties` line `--debug=server` adds one extra line in there:

```
<entry value="ALL" type="String" key="tigase.server.level"/>
```

I think now everything should be clear. You need to add a similar line changing only "key" attribute. If you need to switch logging on for a specific class - `tigase.xmpp.XMPPIOService` for example, add:

```
<entry value="ALL" type="String" key="tigase.xmpp.XMPPIOService.level"/>
```

You can also put there your own package or class name. After you changed the config file you have to restart the server.

*Note, don't overdose the debugging or your logs are full of trash you can read.*

---

# Chapter 38. Importing User Data

Artur Hefczyc <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2010-04-06 21:18

You can easily copy data between Tigase compatible repositories that is repositories for which there is a database connector. It is not that easy however to import data from an external source. Therefore a simple data import functionality has been added to repository utilities package.

You can access repository utilities through command `./bin/repo.sh` or `./scripts/repo.sh` depending on whether you use binary package or source distribution.

`-h` parameter gives you a list of all possible parameters:

```
./scripts/repo.sh --h
```

Parameters:

```
--h                this help message
--sc class         source repository class name
--su uri           source repository init string
--dc class         destination repository class name
--du uri           destination repository init string
--dt string        data content to set/remove in repository
--u user           user ID, if given all operations are only for that ID
                   if you want to add user to AuthRepository parameter must
                   in form: -"user:password"
--st              perform simple test on repository
--at              simple test for adding and removing user
--cp              copy content from source to destination repository
--pr              print content of the repository
--n               data content string is a node string
--kv              data content string is node/key=value string
--add             add data content to repository
--del             delete data content from repository
-----
--roster          check the user roster
--aeg [true|false] Allow empty group list for the contact
--import file     import user data from the file of following format:
                   user_jid, password, roser_jid, roster_nick, subscription, group
```

Note! If you put `UserAuthRepository` implementation as a class name some operation are not allowed and will be silently skipped. Have a look at `UserAuthRepository` to see what operations are possible or what operation does make sense. Alternatively look for admin tools guide on web site.

The most critical parameters are source repository class name and initialization string. Therefore there are a few example, preset parameters which you can use an example and adjust for your system. If you look inside the `repo.sh` script you can find at the end of the script following lines:

```
XML_REP="-sc tigase.db.xml.XMLRepository --su ../testsuite/user-repository.xml_20
MYSQL_REP="-sc tigase.db.jdbc.JDBCRepository --su jdbc:mysql://localhost/tigase?us
```

```
PGSQL_REP="-sc tigase.db.jdbc.JDBCRepository --su jdbc:postgresql://localhost/tigase
java $D --cp $CP tigase.util.RepositoryUtils $MYSQL_REP $*
```

You can see that the source repository has been set to MySQL database with tigase as the database name, root the database user and mypass the user password.

You can adjust these settings for your system.

Now to import data to your repository simply execute the command:

```
./bin/repo.sh --import import-file.txt
```

*Note, the import function is available from **b895***

The format of the import file is very simple. This is a flat file with comma separated values:

```
jid,password,roster_jid,roster_nick,subscription,group
```

To create such a file from MySQL database you have to execute command like this one:

```
SELECT a, b, c, d INTO OUTFILE -'import-file.txt'
FIELDS TERMINATED BY -','
LINES TERMINATED BY -'\n'
FROM test_table;
```

---

# Chapter 39. Drupal Authentication Added

Artur Hefczyc <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2010-04-06 21:18

Well it is authentication against **Drupal** database at the moment. So it is not full integration with **Drupal** yet.

As **Drupal** keeps encrypted passwords in database the only possible authorization protocols are those based on PLAIN passwords.

To protect your passwords **Tigase** server must be used with SSL or TLS encryption.

Implementation of **Drupal** database based authorization is located in `tigase.db.jdbc.DrupalAuth` class. Although this class is capable of adding new user to the repository I recommend to switch in-band registration off due to the caching problems in **Drupal**. Changes in database are not synchronized with **Drupal** yet. Function for adding new users is implemented only to ease user accounts migration from different repository type from earlier **Tigase** server installation.

The idea of that implementation was to allow all accounts administration tasks from **Drupal** like: account creation, all accounts settings, like e-mail, full name, password changes and so on.

**Tigase** server uses following fields from **Drupal** database: name (user account name), pass (user account password), status (status of the account). Server picks up all changes instantly. If user status is not 1 then server won't allow user to login through Jabber/XMPP even if user provides valid password.

There is no *Roster* management in **Drupal** yet. So Roster management have to be done from Jabber/XMPP client.

---

# Chapter 40. Tigase Services

Wojciech Kapcia <wojciech.kapcia@tigase.org [mailto:wojciech.kapcia@tigase.org]> v2.0, June 2014:  
Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2012-08-23 20:49

- `////<<addManageDomain,////Add and Manage Domain`
- `////<<presenceForwarding,////Presence Forwarding`
- `////<<registerXMPP,////Register Own XMPP Domain`

---

# Chapter 41. Add and Manage Domain

Wojciech Kapcia <wojciech.kapcia@tigase.org [mailto:wojciech.kapcia@tigase.org]> v2.0, June 2014:  
Reformatted for AsciiDoc. :toc: :numbered: :website: <http://tigase.net> :Date: 2012-08-23 21:13

For everybody interested in using our service to host own XMPP domain I have good news. You do not have to ask administrator to add your domain or add users for your domain anymore. You can do it on your own.

Please note, this is very new stuff. Something may go wrong or may not be polished. Please report any problems, notices or suggestions.

This is the guide to walk you through new functions and describes how to add a new domain and new users within your domain.

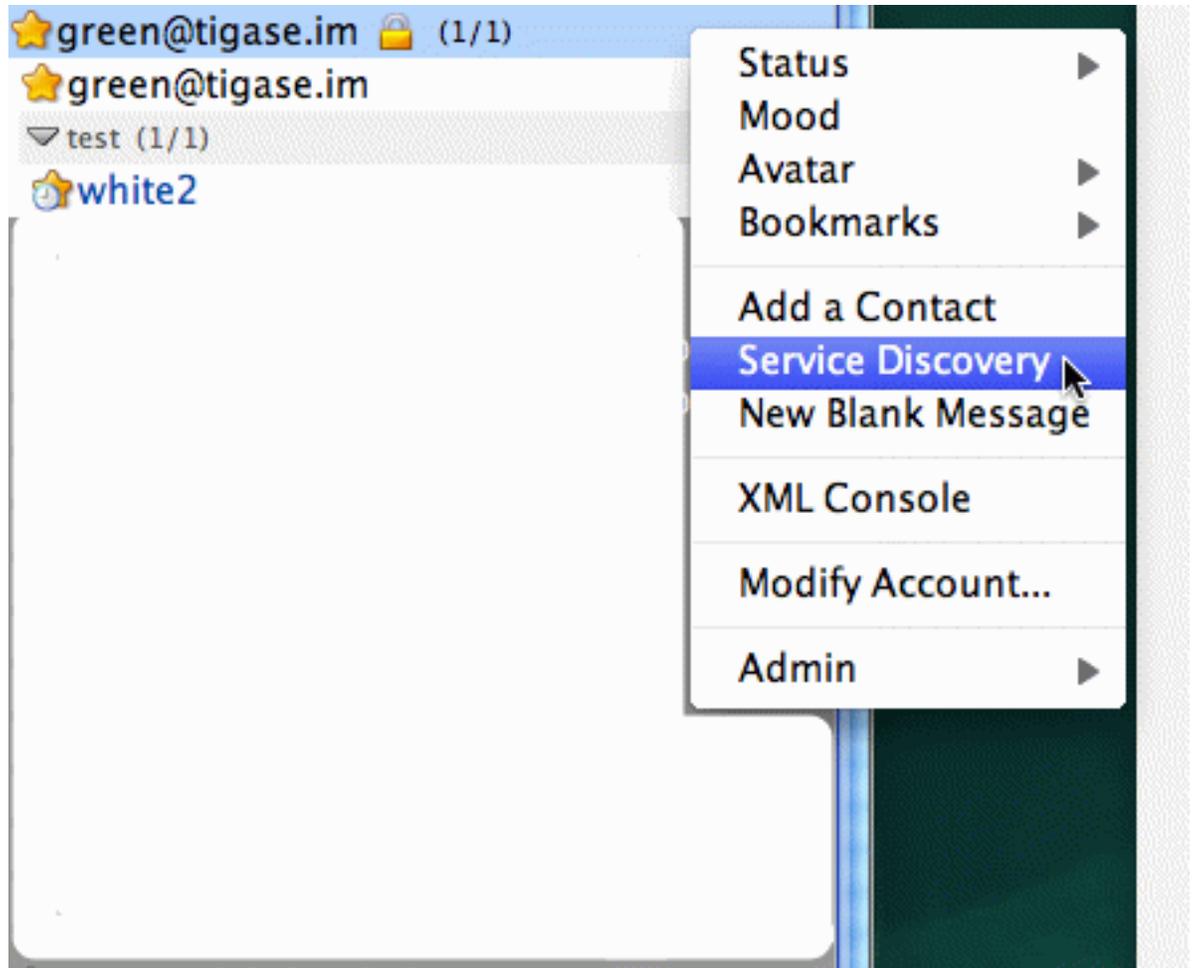
You can do all the stuff from your XMPP client or you can use our WebApplication that allows you to connect to the service and execute admin commands. I recommend Psi [<http://psi-im.org/>] because of its excellent support for parts of the XMPP protocol which are used for domains and users management. You may use any other client as well, there is no limitation but I can only offer support and help if you use Psi client.

Secondly, you need an account on the server. This is because all the commands and features described here are available to local users only. Therefore, if you do not have a registered domain with us yet, please go ahead and register an account on the website either the Jabber.Me [<http://jabber.me/>] or Tigase.IM [<http://www.tigase.im/>].

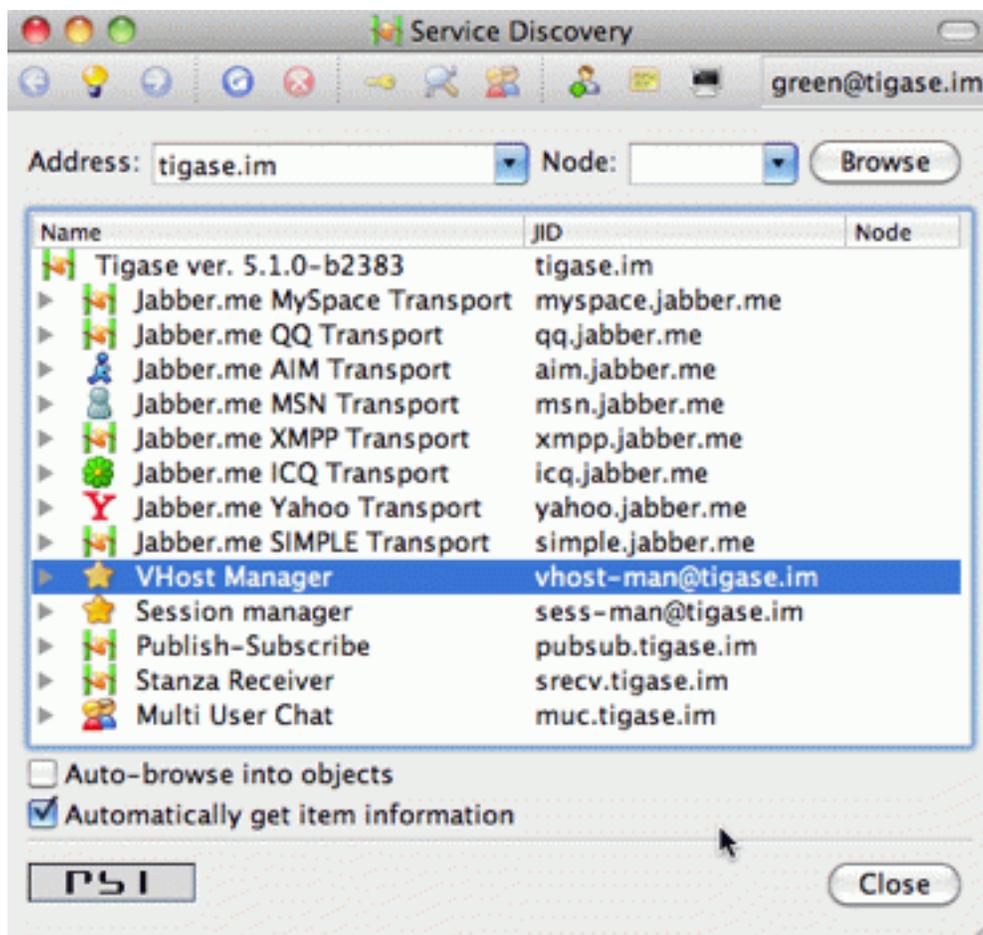
## Adding a New Domain

Once you register an account on one of the websites connect to the XMPP server using the account on the Psi client. I have an example, nice account: `green@tigase.im` [mailto:green@tigase.im] which is used for this guide.

When you are ready right click on the account name in Psi roster window to bring up context menu. Select **Service Discovery** element.



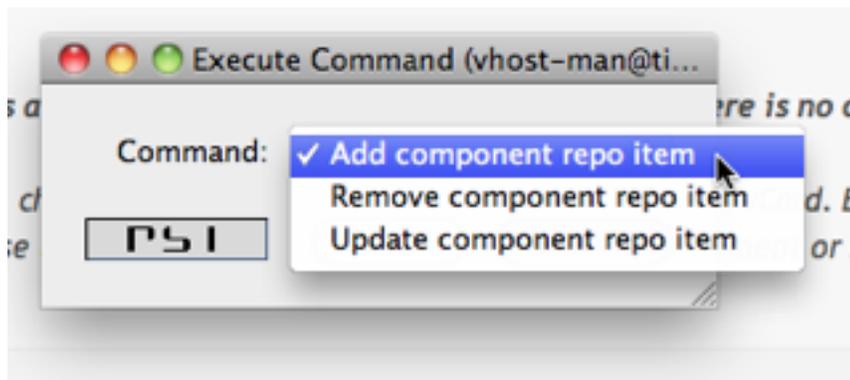
A new window pops up as in the example on the right. Service discovery window is where all the stuff installed on XMPP service should show up. Most of elements on the list are well known transports, MUC and PubSub components. The new stuff on the list, which is interesting for us now, are 2 elements: **VHost Manager** and **Session Manager**.



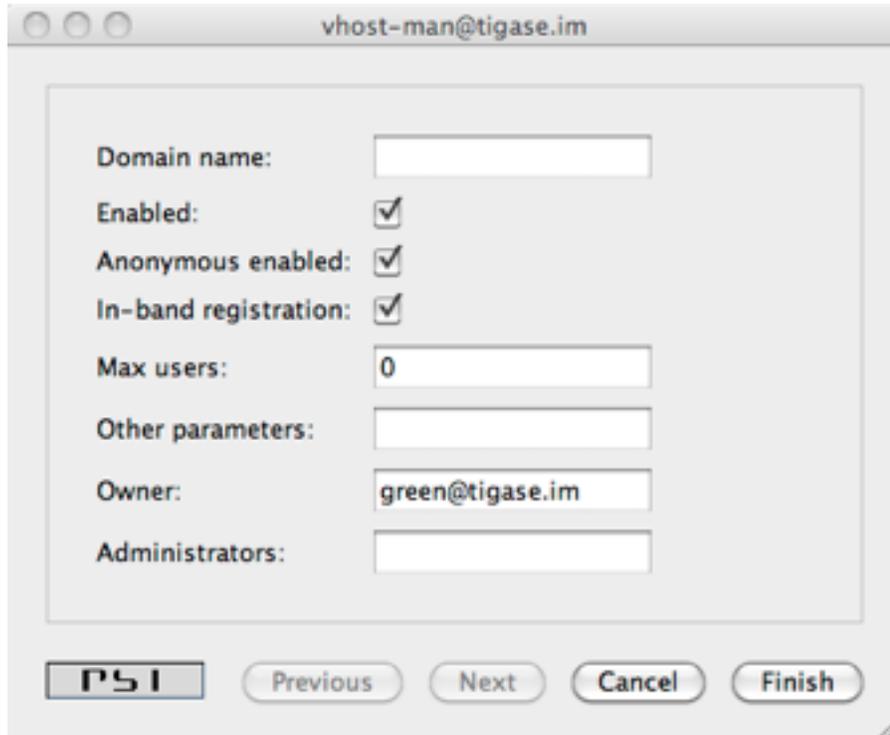
**VHost Manager** component in the Tigase is responsible for managing and controlling virtual hosts on the installation. It provides virtual hosts information to all other parts of the system and also allows you to add new hosts and remove/update existing virtual hosts.

**Session Manager** component in the Tigase is responsible for managing users. In most cases online users but it can also perform some actions on user repository where all user data is stored.

Select **VHost Manager** and double click on it. A new window shows up (might be hidden behind the service discovery window). The window contains another menu with a few items: **Add...**, **Remove...** and **Update...**. These are for adding, removing and updating VHost information. For now, just select the first element **Add...**



Press **Execute** and you get a new window where you can enter all of your VHost configuration details. I think all the fields are pretty much self explanatory. Leave blank field for **Other parameters** for now. **Owner** is you, that is Jabber ID which controls the domain and can change the domain configuration settings or can remove the domain from the service. **Administrators** field can be left blank or can contain comma separated list of Jabber IDs for people who can manage users within the domain. Of course the owner can always manage users for his domain too.



The screenshot shows a window titled "vhost-man@tigase.im" with the following fields and controls:

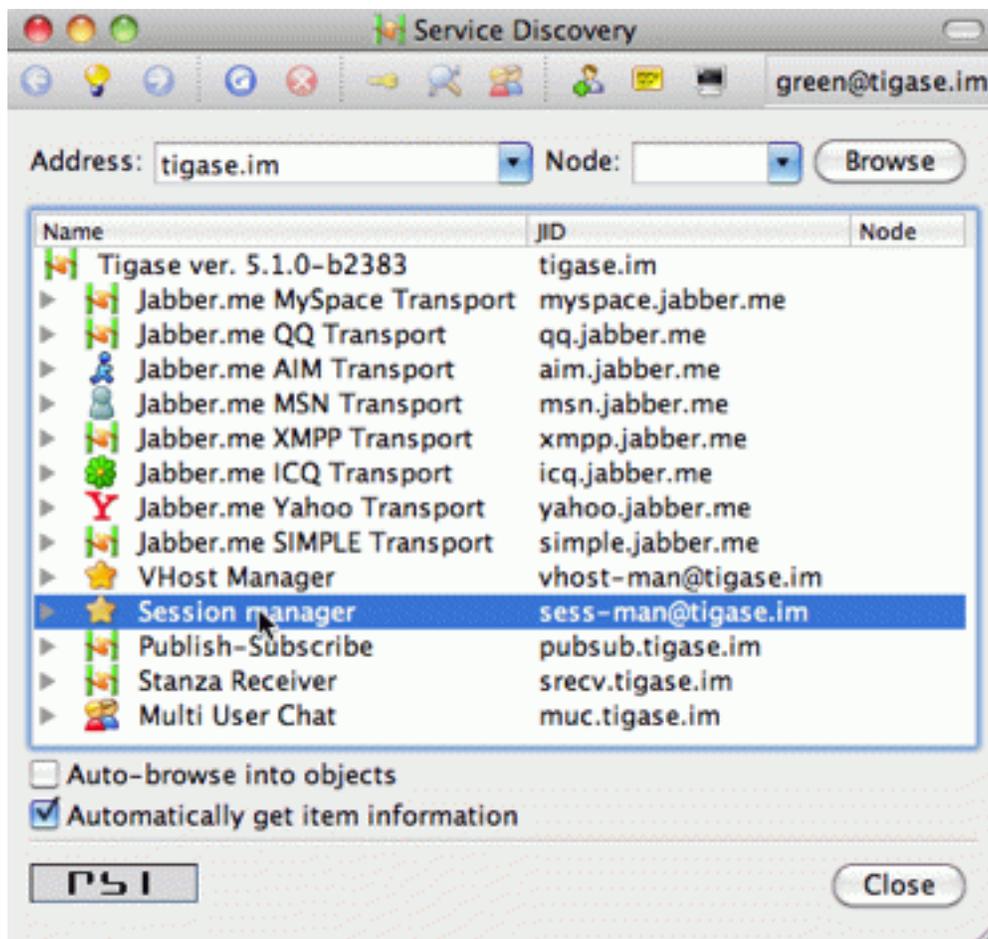
- Domain name:
- Enabled:
- Anonymous enabled:
- In-band registration:
- Max users:
- Other parameters:
- Owner:
- Administrators:

At the bottom, there is a "PSI" button and a set of navigation buttons: "Previous", "Next", "Cancel", and "Finish".

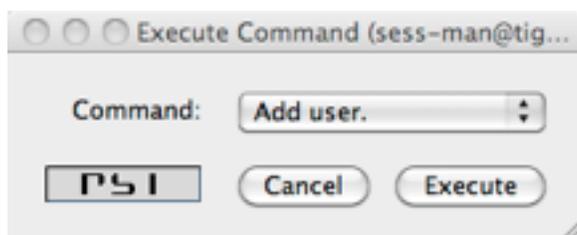
When you are ready press **Finish** button. All done, hopefully. You can get either a window confirming everything went well or a window printing an error message if something went wrong. What can be wrong? There are some restrictions I decided to put on the service to prevent abuse. One of the restrictions is the maximum number of domains a user can register for himself which is **25** right now. Another restriction is that the domain which you add must have a valid DNS entry pointing to our service. The [//////// add XMPP guide](#) describes all the details about DNS settings. Please refer to these instructions if you need more details.

## Adding a New User

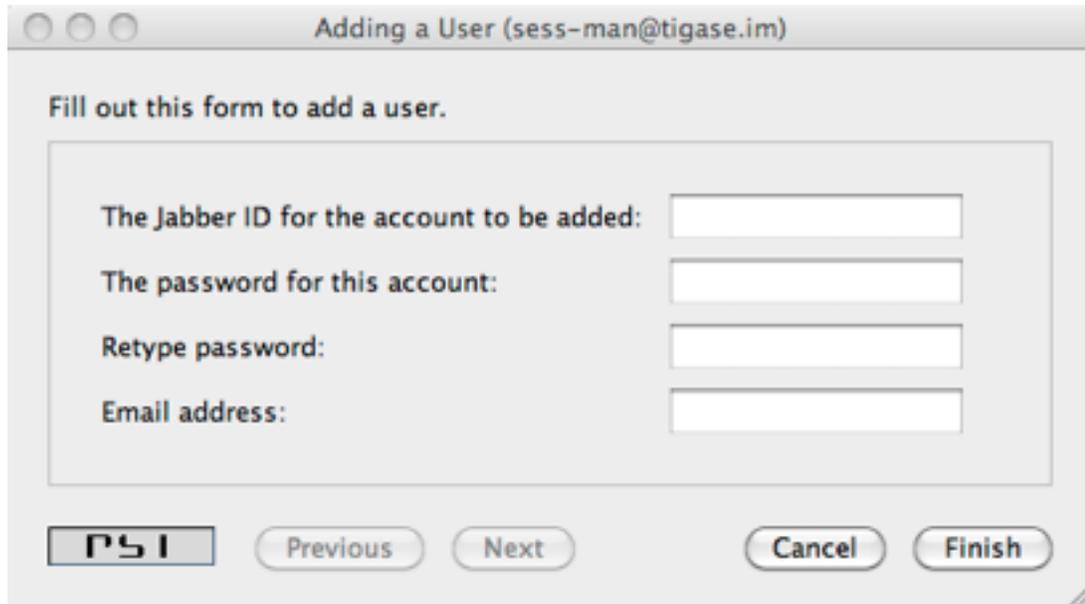
Adding a new user process is quite similar, almost identical to adding a new domain. This time, however we have to select **Session Manager** in the service discovery window.



Double click on the **Session Manager** and a window with SM's commands list shows up. Right now, there is only one command available to domain administrators - **Add user**. I am going to make available more commands in the future and I am waiting for your suggestions.



If you press **Execute** a window presented on the left shows up. Fill all fields accordingly and press **Finish**.



Adding a User (sess-man@tigase.im)

Fill out this form to add a user.

The Jabber ID for the account to be added:

The password for this account:

Retype password:

Email address:

PSI Previous Next Cancel Finish

If everything went well you have just added a new user and you should get a window confirming successful operation. If something was wrong a window with an error message should show up. What can be wrong? You may try to add a user which is already added, you may try to add a user for a domain to which you do not have permission or to non-existent domain. I hope nothing more.

## What Else?

Well, there are plans, very close plans, to add SSL certificates management for your domains. I want to also add more user administration commands but for this I am waiting for your suggestions. You are users, you tell me what you want.

One more thing.

There is quite a few domains already registered by an administrator for people who asked for it using website form. These people cannot control their domains right now. Everybody who is interested in taking control over his domain, please send me an e-mail with his domain details and account for which the ownership should be granted and I will update domain configuration. To make sure the correct person asks for this I require you to send the request as a response to the e-mail I sent you after the domain has been registered in our service.

---

# Chapter 42. Presence Forwarding

Wojciech Kapcia <wojciech.kapcia@tigase.org [mailto:wojciech.kapcia@tigase.org]> v2.0, June 2014:  
Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2012-08-23 21:52

Have you ever thought of displaying your users presence status on the website? Or, maybe, you wanted to integrate XMPP service with your own system and share not only users' accounts but also presence status?

Now it is possible with Tigase. Well, not only possible but also very simple. You have a new option in the domain control form.

Actually there are 2 new options:

1. Presence forward address
2. Message forward address - not fully implemented yet

Presence forward address can be any XMPP address. Usually you want it to be your bot address which can collect your users' presence information. Once this option is set to a valid XMPP address the Tigase forwards user's presence, every time the user changes his status. The presence is processed normally, of course, and distributed to all people from the contact list (roster), plus to this special address. It can be a component or a bot. If this is a bot connecting to a regular XMPP account, **Make sure the presence forward address contains resource part and the bot is connecting with this resource.** Otherwise the presence won't be delivered to the bot.

The screenshot shows a configuration window titled "vhost-man@devel.tigase.org". The window contains the following fields and controls:

- Domain name:** test.tigase.org
- Enabled:**
- Anonymous enabled:**
- In-band registration:**
- Max users:** 0
- Presence forward address:** test4@test.tigase.org/
- Message forward address:** (empty field)
- Other parameters:** (empty field)
- Owner:** admin@devel.tigase.org
- Administrators:** (empty field)

At the bottom of the window, there are five buttons: "PSI", "Previous", "Next", "Cancel", and "Finish".

As the screenshot shows, there are new input lines with option for presence forwarding address and message forwarding address. As you can see this option can be specified separately for each domain, so you can have a different forward address for each domain.

If you have your own Tigase installation, the forwarding address can be also set globally and can be the same for all domains. However, for this website, we offer this feature to all our users who have own domains and this can be set on per-domain basis.

Now, the big question. How this can be used? I am attaching below an example code. With just a few lines of code you can connect a command line bot to the server as a client which would collect all presences from users. Code below is a simple Groovy script which receives presence packet and displays them on the console. However, it should be easy enough to store users' presence information in a database and then load it from a web application.

The bot/client uses our JaXMPP2 [<https://projects.tigase.org/projects/jaxmpp2>] library which is available from our project management system.

You should be able to find a few more code examples on the wiki page.

```
package jaxmppexample
import tigase.jaxmpp.core.client.BareJID
import tigase.jaxmpp.core.client.SessionObject
import tigase.jaxmpp.core.client.exceptions.JaxmppException
import tigase.jaxmpp.core.client.observer.Listener
import tigase.jaxmpp.core.client.xmpp.modules.presence.PresenceModule
import tigase.jaxmpp.core.client.xmpp.modules.presence.PresenceModule.PresenceEvent
import tigase.jaxmpp.j2se.Jaxmpp

final Jaxmpp jaxmpp = new Jaxmpp()
jaxmpp.getProperties().setUserProperty( SessionObject.USER_BARE_JID,
    BareJID.bareJIDInstance( -"test4@test.tigase.org" -) -)
jaxmpp.getProperties().setUserProperty(SessionObject.RESOURCE, -"presence-collector" -)
jaxmpp.getProperties().setUserProperty( SessionObject.PASSWORD, -"pass" -)
jaxmpp.getModulesManager().getModule( PresenceModule.class -).addListener(
    PresenceModule.ContactChangedPresence, new Listener() {
        public void handleEvent( PresenceEvent be -) {
            def msg = (be.getStatus() != null) -? be.getStatus() -: -"none"
            println( -"Presence received:\t" + be.getJid() + -" is now -" + be.getShow()
                -" (" + msg + -)" -)
        }
    }
)
println( -"Logging in..." -)
jaxmpp.login()
println( -"Waiting for the presence for 10 minutes" -)
Thread.sleep( 10 * 60 * 1000 -)
disconnect()
```

---

# Chapter 43. Register Own XMPP Domain

Wojciech Kapcia <wojciech.kapcia@tigase.org [mailto:wojciech.kapcia@tigase.org]> v2.0, June 2014:  
Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2012-08-23 21:05

You can have XMPP service running for your own domain. The only condition right now is that this must be a DNS registered domain and DNS must point to the following DNS address: **tigase.me**. Please note, do not confuse it with `tigase.im` domain name.

We recommend to use SRV records as this is required by XMPP specification but as some DNS services do not allow for SRV records yet we do not require SRV records either. If you want to register: **your-domain.tld** on our XMPP service make sure that either the command:

```
$ host your-domain.tld
your-domain.tld has address 94.23.164.209
```

displays **94.23.164.209** address or commands:

```
$ host --t SRV _xmpp-server._tcp.your-domain.tld
_xmpp-server._tcp.your-domain.tld has SRV record 10 0 5269 tigase.me.
$ host --t SRV _xmpp-client._tcp.your-domain.tld
_xmpp-client._tcp.your-domain.tld has SRV record 10 0 5222 tigase.me.
```

display **tigase.me** DNS name. We strongly recommend not to use the IP address directly however, as if the service grows too much, it will be much easier for us to migrate, expand it using the DNS name rather than IP address.

If you want to have MUC and PubSub available under your domain, you have to setup DNS for **muc.your-domain.tld** and **pubsub.your-domain.tld** domains too.

For MUC:

```
$ host --t SRV _xmpp-server._tcp.muc.your-domain.tld
_xmpp-server._tcp.muc.your-domain.tld has SRV record 10 0 5269 tigase.me.
$ host --t SRV _xmpp-client._tcp.muc.your-domain.tld
_xmpp-client._tcp.muc.your-domain.tld has SRV record 10 0 5222 tigase.me.
```

For PubSub :

```
$ host --t SRV _xmpp-server._tcp.pubsub.your-domain.tld
_xmpp-server._tcp.pubsub.your-domain.tld has SRV record 10 0 5269 tigase.me.
$ host --t SRV _xmpp-client._tcp.pubsub.your-domain.tld
_xmpp-client._tcp.pubsub.your-domain.tld has SRV record 10 0 5222 tigase.me.
```

Now, how do you register your domain with our service?

There are a few ways. I recommend to look at this guide describing how to add and manage your domain on your own. If you cannot or don't want to do it on your own, the way described in the guide please send us a message, either via XMPP to `admin@tigase.im` [mailto:admin@tigase.im] or the contact form requesting new domain. User registration is available via in-band registration protocol. You can also specify whether you want to allow anonymous authentication to be available for your domain and you can specify maximum number of users for your domain.

Any comments or suggestions are very welcomed.

---

# Chapter 44. Server Monitoring

All the documentation and resources related to the Tigase server monitoring.

## Setting Up Remote Monitoring in the Server

The Tigase server can be remotely monitored over following protocols: **JMX/RMI**, **SNMP** and **HTTP**. Even though JMX offers the biggest control and visibility to the server states all of the monitoring services give the same basic set of the server statistics:

- Number of network connections for s2s, c2s and Bosh
- Last second, last minute and last hour load for all main components: SM, MR, c2s, s2s, Bosh, MUC and PubSub
- System statistics - memory usage (heap and non heap) and the server uptime in milliseconds and human readable text.
- Users statistics - number of registered users and number of online user session.

JMX/RMI and SNMP servers offer basic security and can restrict access based and the HTTP server doesn't offer any access restriction mechanisms. Therefore HTTP monitoring is recommended to work behind a firewall.

The monitoring itself causes very low overhead in terms of the resources and CPU consumption on top of the normal Tigase processing requirements so it can be left always on without worrying about performance degradation.

**Note.** *This works with the Tigase server from version 4.2.0 or SVN revision 1418.*

## What You Need.

If you use binary build for version 4.2.0 or later you can skip this section as all required libraries are included in the installation package: Extras pack. Please install the pack and you don't have to worry about libraries any more.

If you run the server from sources or SVN continue reading:

The remote monitoring requires an external library therefore it is implemented in the tigase-extras package to comply with the basic rule - no third-party libraries are needed for the core Tigase server. But still what you need for the remote monitoring activation is just 2 jar files:

1. **tigase-extras** [<http://www.tigase.org/project/extras>] the last **0.3.0-SNAPSHOT** [<http://maven.tigase.org/tigase/tigase-extras/0.3.0-SNAPSHOT/tigase-extras-0.3.0-20090110.223424-2.jar>] or later
2. **jdmkrt.jar** file from OpenDMK [<https://opendmk.dev.java.net/>] project version 5.1 or later. A copy of this jar file is also available in our maven repository: **jdmkrt.jar** [<http://maven.tigase.org/opendmk/jdmkrt/1.0-b02/jdmkrt-1.0-b02.jar>].

Download both libraries and put them in the `libs/` directory.

## Activation

You can either run the Tigase installer and use configuration wizard to activate the monitoring or edit `etc/init.properties` file and add following line:

```
--monitoring=jmx:9050,http:9080,snmp:9060
```

As you see there is only a single line where you put list of monitoring servers you want to activate. Each server is responsible for activation of a different protocol and takes a single parameter - port number. There are following protocols supported right now:

- **jmx** - activating monitoring via JMX/RMI
- **http** - activating monitoring over HTTP protocol
- **snmp** - activating monitoring over SNMP protocol

You can have all protocols active at the same time or any combination of them or none.

## Security

Both JMX and SNMP offer security protection to limit access to monitoring data. The security configuration is a bit different for both.

### JMX

After the server installation or in the SVN repository you can find 2 files in the `etc/` directory: **jmx.access** and **jmx.password**.

- **jmx.access** is a user permission file. You can use it to specify whether the user can access the monitoring data for reading only 'readonly' or with read-write 'readwrite' access. There are example entries in the file already and the content may simply look like:

```
monitor readonly
admin readwrite
```

- **jmx.password** is a user password file. You can set user passwords here and the format again is very simple and the same as for `jmx.access`. There are example entries already provided for you convenience. Content of the file may look like the example below:

```
admin admin_pass
monitor monitor_pass
```

Using above to files you can control who and how can access the JMX monitoring services.

### SNMP

Access to the SNMP monitoring is controlled using ACL (access control lists) which can be configured in the file **snmp.acl** located in `etc/` directory. It contains lots of detailed instructions how to setup ACL and restrict access per user, host and what kind access is allowed. The simplest possible configuration may look like this:

```
acl = {
```

```
{
  communities = public, private
  access = read-only
  managers = public.host.com, private.host.com
-}
{
  communities = admin
  access = read-write
  managers = localhost, admin.host.com
-}
}
```

You might also need Tigase MIB definition: TIGASE-MANAGEMENT-MIB.mib [<https://svn.tigase.org/repos/tigase-extras/trunk/src/main/resources/mib/TIGASE-MANAGEMENT-MIB.mib>] for the server specific statistics. The MIB contains definition for all the server statistics exposed via SNMP.

## Retrieving statistics from the server

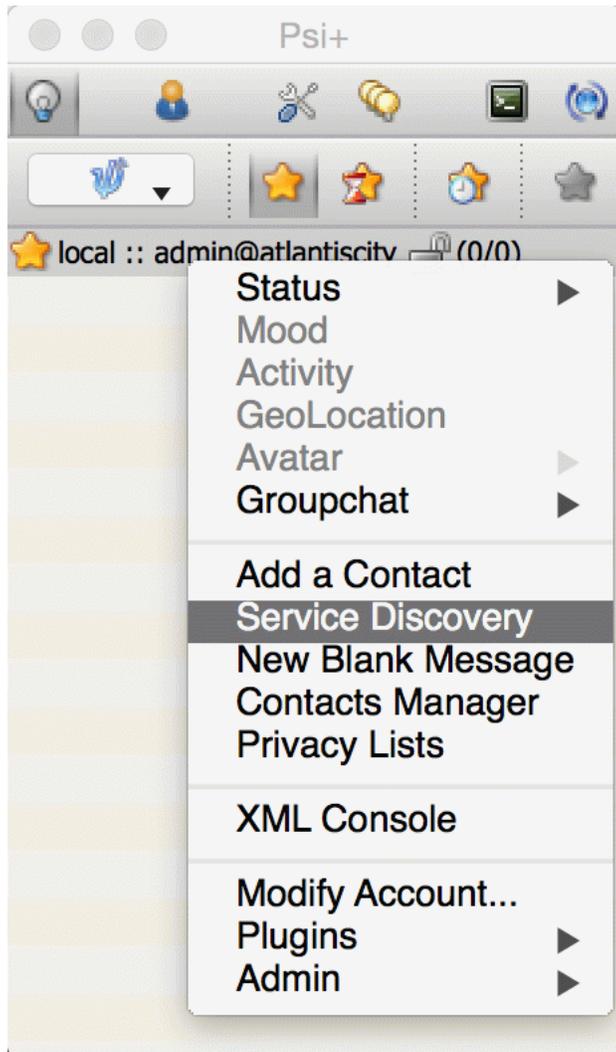
By default we can retrieve server statistics using XMPP. After a successful setup of the section called “Setting Up Remote Monitoring in the Server” we can enable additional@@, configured methods (JMX, HTTP, etc).

## Retrieving statistics using XMPP

Accessing statistics over XMPP protocol requires any XMPP client capable of executing XEP-0050: Ad-Hoc Commands [<http://xmpp.org/extensions/xep-0050.html>]. It’s essential to remember, that only administrator (a user whose JID is configured as administrative) can access the statistics.

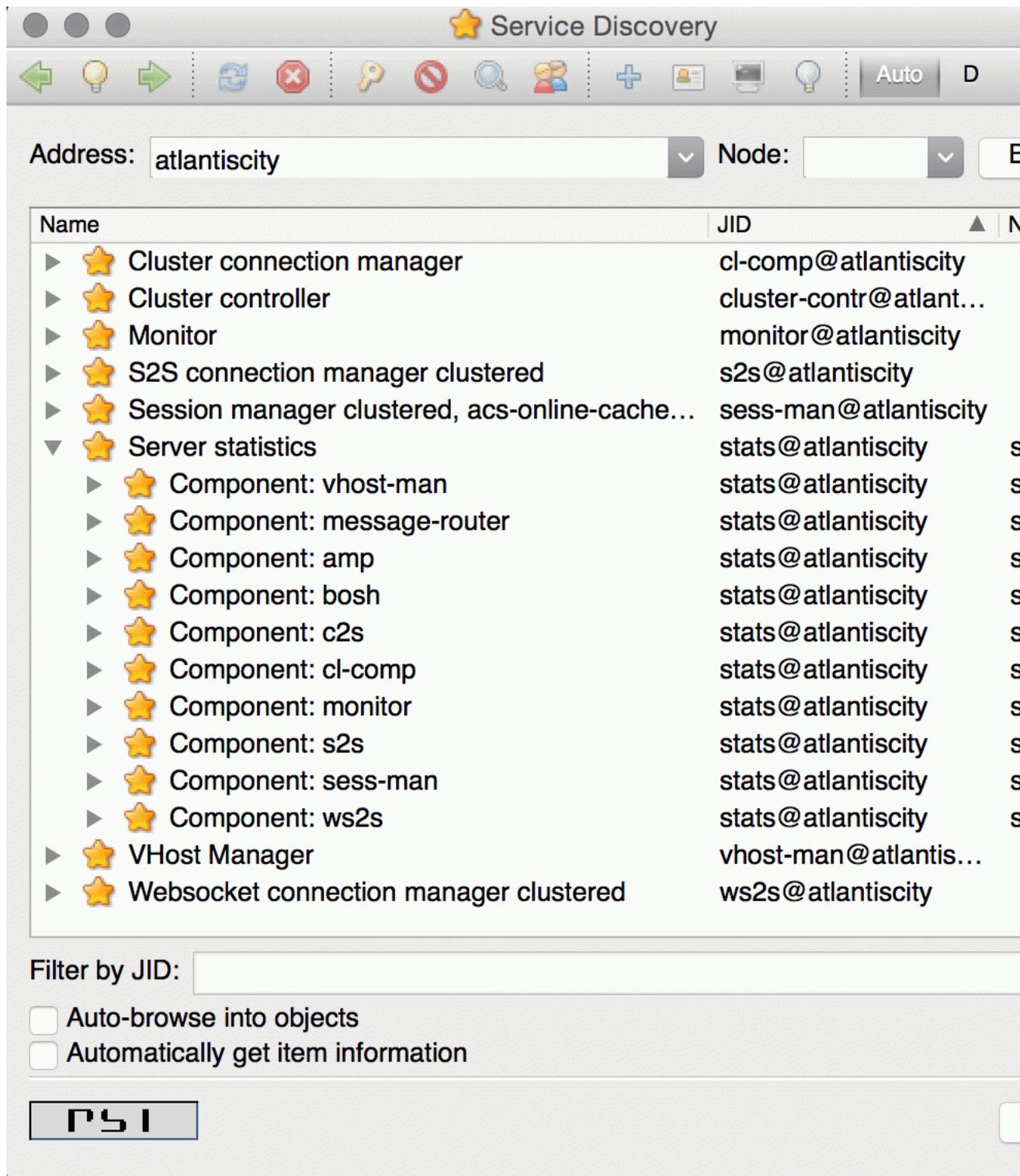
## Psi XMPP Client

For the purpose of this guide a Psi [<http://psi-im.org/>] client will be used. After successfully configuring and connecting to account with administrative privileges we need to access *Service Discovery*, either from application menu or from context menu of the particular account account:

**Figure 44.1. Access service discovery**

In the *Service Discovery* window we need to find *Server Statistics* component:

Figure 44.2. Access statistics component in service discovery



We can either access statistics for all components or select particular component after expanding the tree. To execute ad-hoc command simply double click on the particular node which will open window with statistics:

stats@atlantiscity

message-router/Local hostname:

message-router/Uptime:

message-router/CPU usage:

message-router/Max Heap mem:

message-router/Used Heap:

c2s/Open connections:

sess-man/Open user connections:

sess-man/Maximum user connections:

sess-man/Open user sessions:

presence/Users status changes:

Stats level:

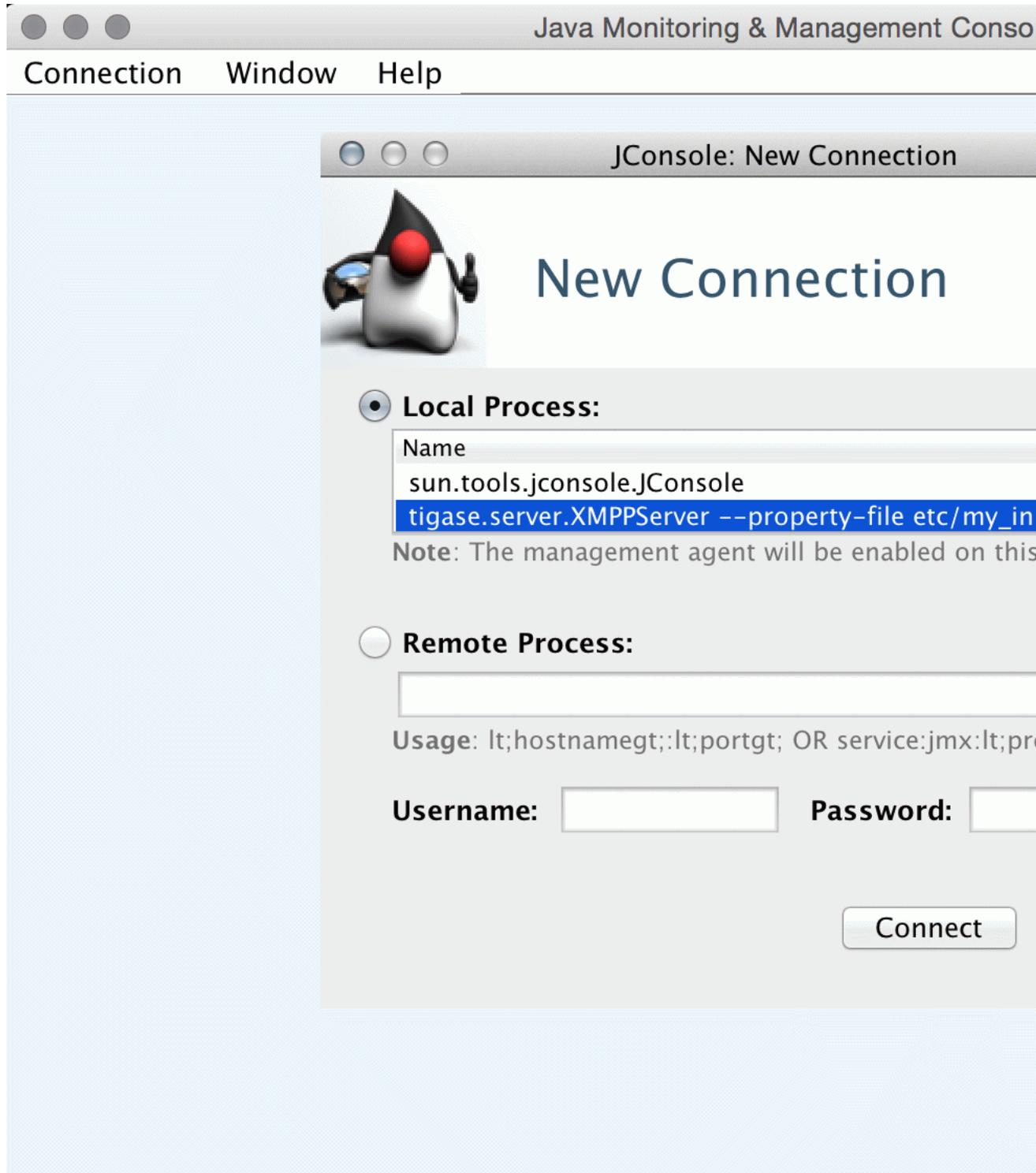
In this window, in addition to see the statistics, we can adjust *Stats level* by selecting desired level from the list and confirming by pressing *Finish*.

## Retrieving statistics using JMX

In order to access statistics over JMX we need to enable support for it in Tigase - the section called "Activation". Afterwards we can use a number of tools, for example following:

### JConsole

After opening JConsole we either select local process or provide details of the remote process, including IP, port and credentials from **etc/jmx.\*** files:



Afterwards we navigate to MBeans tab from where we can access `tigase.stats` MBean. It offers similar options to XMPP - either accessing statistics for all components or only for particular component as well as adjusting level for which we want to obtain statistics:

Java Monitoring & Ma

Connection Window Help

pid: 86933 tigase.server.XMPPServer -

Overview | Memory | Threads | C

▶ JMIImplementation  
▶ com.sun.management  
▶ java.lang  
▶ java.nio  
▶ java.util.logging  
▶ localhost  
▶ org.apache.derby  
▶ system  
▼ tigase.stats  
    ▶  StatisticsProvider  
        ▶ Attributes  
        ▶ **Operations**

Operation invocation

java.util.Map  (level

java.util.Map

java.util.Map

java.util.Map



## StatsDumper.groovy

In order to collect statistics over period of time following groovy script can be used: StatsDumper.groovy [files/StatsDumper.groovy]. It's a Simple JMX client that connects to Tigase and periodically save all statistics to files.

It takes following parameters:

```
$ groovy StatsDumper.groovy [hostname] [username] [password] [dir] [port] [delay(m
```

- hostname - address of the instance
- username - JMX username
- password - JMX username
- dir - directory to which save the files with statistics
- port - port on which to make the connection
- delay(ms) - initial delay in milliseconds after which statistics should be saved
- interval(ms) - interval between each retrieval/saving of statistics
- loadhistory(bool) - indicates whether or not load statistics history from server (if such is enabled in Tigase)

---

# Chapter 45. Tips and Tricks

The section contains some short trick and tips helping in different kind of issues related to the server administration and maintenance.

## Tigase Tip: Checking the Runtime Environment

It has happened recently that we have tried very hard to fix a few annoying problems on one of the Tigase installations. Whatever we did, however the problems still existed after uploading a new version and the server restart. It worked fine in our development environment and it just didn't on the target system.

It turned out that due to a specific environment settings on the target system an old version of the Tigase server was always started regardless updates we were uploading. When I finally started looking at the installation the first indication that something is wrong was lack of any log files in place where I expected them.

The best way to check all the environment settings used to start the Tigase server is to use..... **check** command line parameter:

```
./scripts/tigase.sh check etc/tigase.conf
Checking arguments to Tigase
TIGASE_HOME = -.
TIGASE_JAR = jars/tigase-server.jar
TIGASE_PARAMS = etc/tigase.conf
TIGASE_CONFIG = etc/tigase.xml
TIGASE_RUN = tigase.server.XMPPServer --c etc/tigase.xml ---property-file etc/init
TIGASE_PID = -./logs/tigase.pid
TIGASE_OPTIONS = ---property-file etc/init.properties
JAVA_OPTIONS = --Dfile.encoding=UTF-8 --Dsun.jnu.encoding=UTF-8 \
  --Djdbc.drivers=com.mysql.jdbc.Driver:org.postgresql.Driver \
  --server --Xms100M --Xmx200M --XX:PermSize=32m --XX:MaxPermSize=256m
JAVA = -/System/Library/Frameworks/JavaVM.framework/Versions/1.6/Home/bin/java
JAVA_CMD =
CLASSPATH = -./jars/tigase-server.jar:./libs/jdbc-mysql.jar:./libs/jdbc-postgresql.jar:
  ./libs/tigase-extras.jar:./libs/tigase-muc.jar:./libs/tigase-pubsub.jar:\
  ./libs/tigase-utils.jar:./libs/tigase-xmltools.jar
TIGASE_CMD = -/System/Library/Frameworks/JavaVM.framework/Versions/1.6/Home/bin/java
  --Dfile.encoding=UTF-8 --Dsun.jnu.encoding=UTF-8 \
  --Djdbc.drivers=com.mysql.jdbc.Driver:org.postgresql.Driver \
  --server --Xms100M --Xmx200M --XX:PermSize=32m --XX:MaxPermSize=256m \
  --cp -./jars/tigase-server.jar:./libs/jdbc-mysql.jar:./libs/jdbc-postgresql.jar:
  ./libs/tigase-extras.jar:./libs/tigase-muc.jar:./libs/tigase-pubsub.jar:\
  ./libs/tigase-utils.jar:./libs/tigase-xmltools.jar tigase.server.XMPPServer \
  --c etc/tigase.xml ---property-file etc/init.properties
TIGASE_CONSOLE_LOG = -./logs/tigase-console.log
```

In our case **TIGASE\_HOME** was set to a fixed location pointing to an old version of the server files. The quick **check** command may be a real time saver.

## Tigase Tip: Checking Cluster Connections

After setting up clustering one may want to verify that cluster is operational. Right now it can be done in two fold manner - first by checking that there are actual network connections established between cluster nodes, the next step is to check internal status of the server.

### Established connections

There are number of ways to check for opened connections, simplest one use command line. (Tigase uses port 5277 for cluster connections)

- Linux

```
$ lsof --iTCP:5277 --sTCP:ESTABLISHED --P --n
```

- Windows

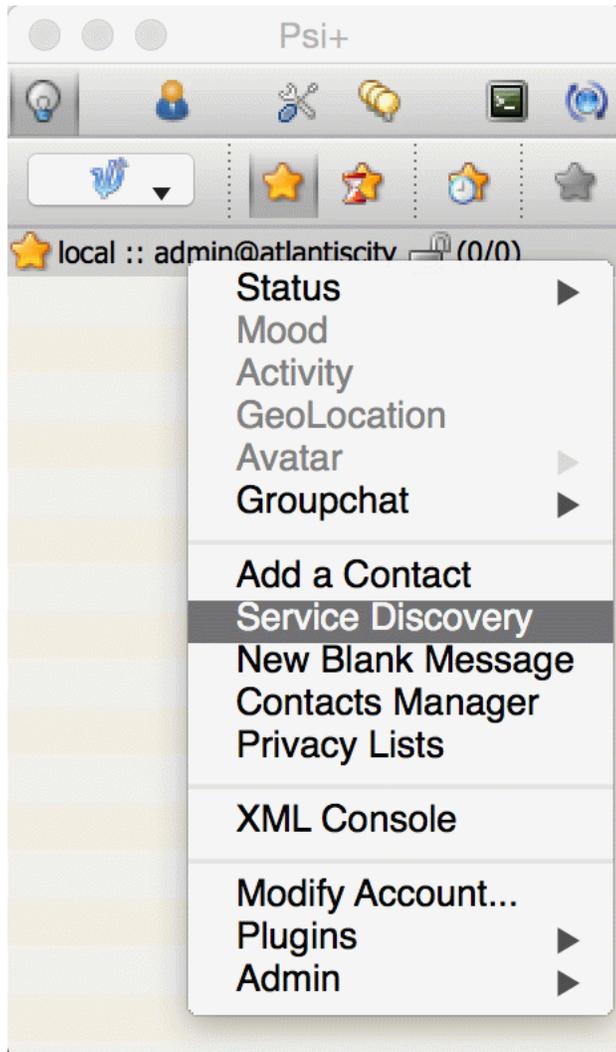
```
C:\WINNT>netstat --anp tcp -| find -":5277 -"
```

### Cluster nodes connected (using XMPP)

Verifying clustering connectivity over XMPP protocol requires any XMPP client capable of XEP-0030: Service Discovery [<http://xmpp.org/extensions/xep-0030.html>]. It's essential to remember, that only administrator (a user whose JID is configured as administrative) has access.

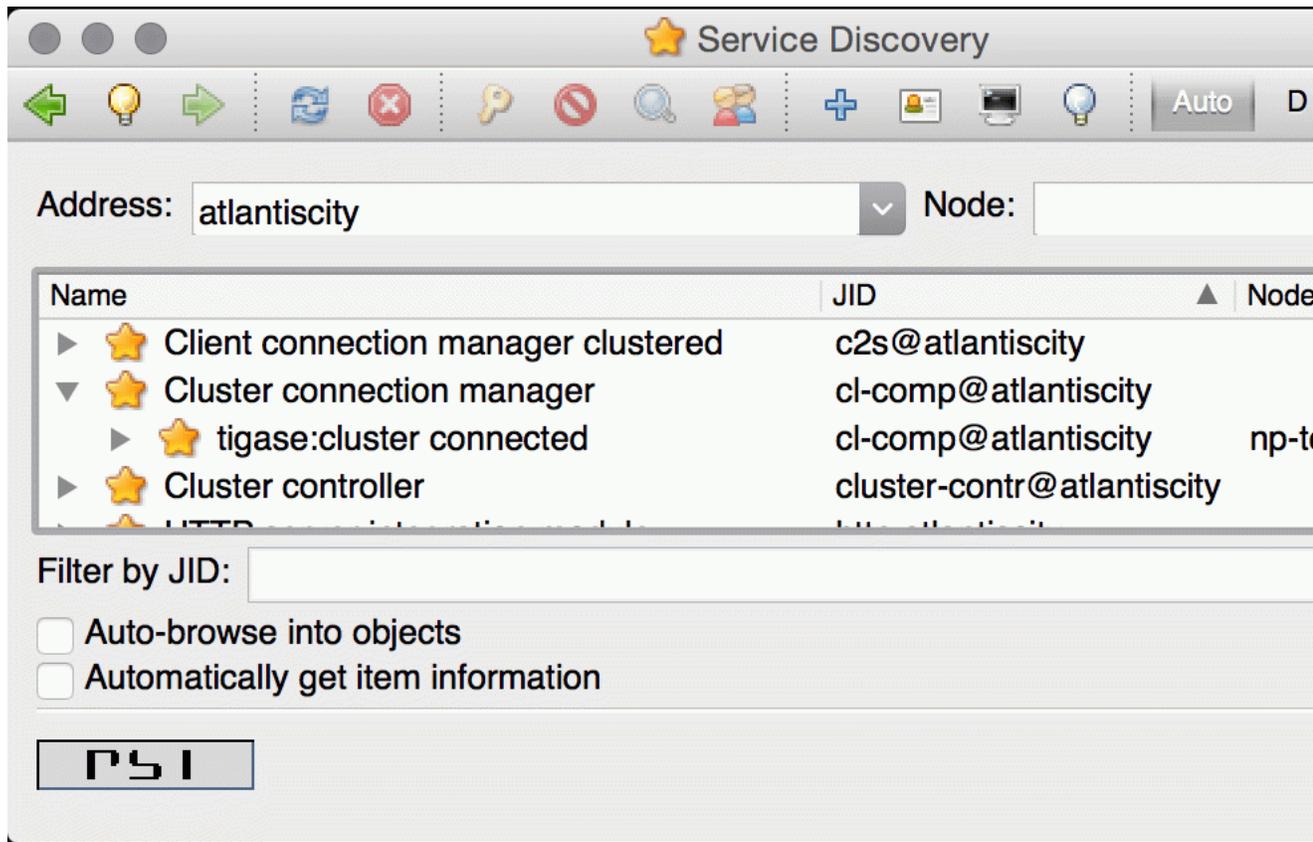
### Psi XMPP Client

For the purpose of this guide a Psi [<http://psi-im.org/>] client will be used. After successfully configuring and connecting to account with administrative privileges we need to access *Service Discovery*, either from application menu or from context menu of the particular account account:

**Figure 45.1. Access service discovery**

In the *Service Discovery* window we need to find *Cluster Connection Manager* component. After expanding the tree node for the component a list of all cluster nodes will be presented with the current status (either *connected* or *disconnected*). Node column will contain actual hostname of the cluster node:

Figure 45.2. List of cluster nodes



---

# Chapter 46. Tigase Server version 5.x

Artur Hefczyc <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2010-01-06 20:22

This section contains documentation for the Tigase server version 5.x or above only. Some documents describe features specific to the new version and some other describe just extensions of older features with references to the complete guides.

- [51schemaUpgeade](#), [Tigase 5.1 Database Schema Upgrade](#)
- [5xServerConfig](#), [Server Configuration 5.x](#)

---

# Chapter 47. Tigase 5.1 Database Schema Upgrade

Wojciech Kapcia <wojciech.kapcia@tigase.org [mailto:wojciech.kapcia@tigase.org]> v2.0, June 2014:  
Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2012-07-16 11:10

Unfortunately we had to make a small change to the database schema for 5.1.0 version of the Tigase server.

It does not affect data or data structure, only the way some data is accessed in database. We added one more stored procedure which has to be installed in database if you upgrade your installation from a previous Tigase version.

Please follow detailed guide for the database applicable to your installation: [Derby Database Schema Upgrade for Tigase 5.1](#), [MySQL Database Schema Upgrade for Tigase 5.1](#), [PostgreSQL Database Schema Upgrade for Tigase 5.1](#)

- [Derby Database Schema Upgrade for Tigase 5.1](#)
- [MySQL Database Schema Upgrade for Tigase 5.1](#)
- [PostgreSQL Database Schema Upgrade for Tigase 5.1](#)

---

# Chapter 48. Derby Database Schema Upgrade for Tigase 5.1

Wojciech Kapcia <wojciech.kapcia@tigase.org [mailto:wojciech.kapcia@tigase.org]> v2.0, June 2014:  
Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2012-06-20 19:12

Unfortunately we had to make a small change to the database schema for 5.1.0 version of the Tigase server.

The good news is that it does not affect data or data structure, only the way some data is accessed in database. We added one more stored procedure which has to be installed in database if you upgrade your installation from a previous Tigase version.

Therefore the schema upgrade is very simple and safe but make sure the current database schema is in version 4.0.

First things first - make a database backup:

```
tar --czf derbyDB.tar.gz -/path/to/derbyDB
```

If you need to restore database for any reason simply extract files from the backup archive:

```
rm --rf -/path/to/derbyDB
tar --xf derbyDB.tar.gz
```

Now we can run schema upgrade script

```
java --Dij.protocol=jdbc:derby: --Dij.database="/path/to/derbyDB" \  
-Dderby.system.home=`pwd` \  
-cp libs/derby.jar:libs/derbytools.jar:jars/tigase-server.jar \  
org.apache.derby.tools.ij database/postgresql-schema-upgrade-to-5-1.sql
```

---

# Chapter 49. MySQL Database Schema Upgrade for Tigase 5.1

Artur Hefczyc <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2012-04-21 03:58

Unfortunately we had to make a small change to the database schema for 5.1.0 version of the Tigase server.

The good news it does not affect data or data structure, only the way some data is accessed in database. We added one more stored procedure which has to be installed in database if you upgrade your installation from a previous Tigase version.

Therefore the schema upgrade is very simple and safe but make sure the current database schema is in version 4.0. If you happen to use ancient version of the Tigase before number 4.0 and you want to upgrade to 5.1 you have to run `mysql -u root -p -e "source database/schemaUpgrade/4.0 upgrade script first."`

Assumptions:

1. **tigasedb** is a database name
2. **tigase\_user** is a database user name
3. **mypass** is database user password

First things first - make a database backup:

```
mysqldump --u tigase_user --pypass tigasedb > tigasedb_dump.sql
```

If you need to restore database for any reason execute following commands:

```
mysqladmin --u tigase_user --pypass drop tigasedb
mysqladmin --u tigase_user --pypass create tigasedb
mysql --u tigase_user --pypass tigasedb < tigasedb_dump.sql
```

*Note! You may be required to use root user and his password to execute mysqladmin commands.*

Now we can run schema upgrade script

```
mysql --u tigase_user --pypass tigasedb < database/mysql-schema-upgrade-to-5-1.sql
```

---

# Chapter 50. PostgreSQL Database Schema Upgrade for Tigase 5.1

Andrzej Wojcik <andrzejw@tigase.org [mailto:andrzejw@tigase.org]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2012-06-16 11:03

Unfortunately we had to make a small change to the database schema for 5.1.0 version of the Tigase server.

The good news it does not affect data or data structure, only the way some data is accessed in database. We added one more stored procedure which has to be installed in database if you upgrade your installation from a previous Tigase version.

Therefore the schema upgrade is very simple and safe but make sure the current database schema is in version 4.0.

Assumptions:

- **tigasedb** is a database name
- **tigase\_user** is a database user name
- **admin\_db\_user** is database admin user name

First things first - make a database backup:

```
pg_dump --U tigase_user --W tigasedb > tigasedb_dump.sql
```

If you need to restore database for any reason execute following commands:

```
dropdb --U admin_db_user --W tigasedb
createdb --U admin_db_user --W --O tigase_user tigasedb
psql --U tigase_user --W tigasedb < tigasedb_dump.sql
```

Now we can run schema upgrade script

```
psql --q --U tigase_user --W tigasedb --f database/postgresql-schema-upgrade-to-5-
```

---

# Chapter 51. Server Configuration 5.x

Artur Hefczyc <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2010-01-06 20:22

Collection of documents and guides for the Tigase server version 5.x and later.

- [Configuration Changes in the Tigase Server 5.x](#)
- [Advanced Message Processing - AMP XEP-0079](#)
- [Configuration SASL EXTERNAL](#)
- [Server to Server Protocol Settings](#)

---

# Chapter 52. Configuration Changes in the Tigase Server 5.x

Artur Hefczyc <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2010-01-06 20:22

The whole configuration framework for the Tigase server has been redesigned and rewritten. This was done to cleanup all the configuration code and logic and also extend the current functionality and allow for configuration storage in different kinds of repositories - memory, file, database, ...

The title says configuration changes but the version 5.x still follows our policy about backward compatibility so the changes should be rather called extensions.

There is however one change which can affect a few users. Those who use the server and worked with it's configuration remember the mess and confusion related to duality in the server configuration - the `init.properties` file and `tigase.xml` file. This is now over.

## Reverting To the Old Behaviour

While using the `tigase.xml` file is still possible and the whole old behaviour can be preserved it is now disabled by default. By default the Tigase server reads only `init.properties` file with initial settings and stores all the complete configuration in memory only.

The `init.properties` works exactly as before and all old parameters are still working exactly as before. The only difference is the lack of the `tigase.xml` which is not created by default and is not read by default if it is present. The main advantage is that you don't have to remove it each time you change something in the `init.properties` to pick up new settings.

I will first present how to revert to old behaviour as this might be critical to some existing systems which want to upgrade. This is actually very simple to accomplish. The Tigase server now, offers pluggable repository support. This means that you can easily extend current functionality with a different configuration storage by writing own class which reads and writes configuration parameters.

By default class `tigase.conf.ConfigurationCache` is loaded which stores configuration in memory only.

Please note, the `init.properties` file is always read if it exists at given location.

To revert to the old behaviour you just need to pass a parameter to the Tigase server with a class name which is responsible for keeping server parameters in the old XML file. You can do it in two ways:

1. Add a parameter to `init.properties` file:

```
--tigase-config-repo-class=tigase.conf.ConfigXMLRepository
```

2. You can pass a system property to the JVM at the startup time:

```
-Dtigase-config-repo-class=tigase.conf.ConfigXMLRepository
```

## Default Behaviour

By default the Tigase server loads `tigase.conf.ConfigurationCache` class which stores the whole configuration in memory. Please note that the `initPropertiesGuide,init.properties` file with initial settings is always loaded if it is available at the given location and all settings in this file work exactly as before. For more details, please refer to the online documentation.

A couple of times I mention about 'initial configuration' and 'whole configuration'. What is this about, what is the difference?

The 'initial configuration' are startup settings provided by the user in the `init.properties` file. Most of the server elements, however use far more configuration parameters which are set to sensible default values if they are not provided by the user. The configuration framework in the Tigase server, however always keeps the complete configuration of all active elements. This is implemented in such a way to make it possible to present currently used settings to the end-users or administrators and allow them to change the server parameters at runtime.

## Storing Configuration in SQL Database

There is one more configuration storage implemented right now. It allows you to store the server settings in the SQL database. In most cases this is not quite useful, just opposite, very inconvenient. There is however at least one case when you really want to keep the server configuration in the SQL database. This is the cluster mode. If you have a Tigase cluster system of 10 or more nodes it is much easier to keep configuration in a single central location and manage it from there, rather than go to every single machine, every time you want to change some settings. You can even change any settings for all cluster nodes with a single database query.

You set the SQL storage the same way as you set it for XML file, there is, however one more parameter as you have to provide also database connection string for the server so it knows where to connect to for the settings:

1. Parameters in `init.properties` file:

```
--tigase-config-repo-class=tigase.conf.ConfigSQLRepository
--tigase-config-repo-uri=connection-uri
```

2. Alternatively you can provide system properties to the JVM:

```
-Dtigase-config-repo-class=tigase.conf.ConfigSQLRepository
-Dtigase-config-repo-uri=connection-uri
```

Please note, the current implementation for the SQL storage automatically creates necessary table if it does not exist. So you don't have to worry about the schema but you should make sure that the database user used by the Tigase has permissions to create a table.

Configuration is stored in table with following schema:

```
create table tigase_configuration (
-- The component name by which the configuration parameter
-- is used.
  component_name varchar(127) NOT NULL,

-- The configuration property key name or identifier.
  key_name varchar(127) NOT NULL,

-- The configuration property value
  value varchar(8191) NOT NULL,

-- The cluster node by which the configuration property is read,
-- if empty it will be read by all cluster nodes.
  cluster_node varchar(255) NOT NULL DEFAULT '',
```

```
-- Additional, secondary identifier for the configuration property.
-- The configuration can be organised in a hierarchical way to allow
-- multiple occurrences of the same property name for a single
-- component, for example you can have the same property for
-- different tcp/ip ports set to a different value:
-- c2s/5222/port_type=plain
-- c2s/5223/port_type=ssl
-- the port number is a secondary identifier.
  key_node varchar(127) NOT NULL DEFAULT '',

-- Not currently used. In future it will be used to distinguish between
-- different kind of properties (initial settings, defaults, updated by
-- user, etc...)
  flag varchar(32) NOT NULL DEFAULT 'DEFAULT',

-- The system detects basic Java types and stores information about
-- the property type, when the property is read the original property
-- type is restored and provided to the component without need for
-- a parsing or conversion.
  value_type varchar(8) NOT NULL DEFAULT 'S',

-- It is not currently used. In the future it will be used to reload
-- settings changed in last, defined period of time. Basicall, the
-- system can automatically check the configuration database to
-- see whether some properties have been updated, then reload
-- them and apply automatically.
  last_update          timestamp,

primary key(cluster_node, component_name, key_node,
            key_node, flag));
```

## Going Further

There is more. As the configuration mechanism in the Tigase server offers pluggable storage engines you can easily write your own engine by implementing the interface: `tigase.conf.ConfigRepositoryIfc` or by extending one of current implementations.

There is even more. You can go even further. The whole configuration framework is pluggable and you can replace it completely if it doesn't suites you well enough. Your implementation has to extend `tigase.conf.ConfiguratorAbstract` class and can be set using JVM system property (as this is configuration framework you can't do this via any configuration system):

```
-Dtigase-configurator=tigase.conf.Configurator
```

The example above shows the parameter set to the default configuration framework.

## Message Router Implementation is Configurable Too

The Message router component was the only component which was fixed to the Tigase instance. In theory it could always be replaced but in practise there was no way of doing it as that was the first element loaded at the server startup time.

Now the Tigase message router implementation can be easily replaced to and it can be made a configurable option if needed.

At the server startup time the code creates configurator and calls method: `getMessageRouterClassName()` which by default returns class: `tigase.server.MessageRouter`. You can extend the configurator and provide any different class name instead which implements required interfaces. You can even make it configurable. It is no longer a fixed thing for the server instance.

---

# Chapter 53. Advanced Message Processing - AMP XEP-0079

Artur Hefczyc <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2010-05-13 16:09

The Tigase server **5.1.0** or later offers support for Advanced Message Processing [<http://xmpp.org/extensions/xep-0079.html>], called AMP or XEP-0079.

It is enabled by default but there are several configuration options that you may tweak.

Configuration is not very complex but as the AMP is implemented as a component in the Tigase server it does needs a few settings to get it right.

Here is a first, brief overview of the AMP configuration and later detailed explanation of each parameter.

```
--sm-plugins=amp, -message, -msgoffline
--amp-repo-uri=jdbc:mysql://localhost/tigasedb?user=db_usr&password=db_pwd
--amp-security-level=STRICT
sess-man/plugins-conf/amp/amp-jid=amp@your-domain.tld
```

First of all: plugins.

Even though the whole functionality is implemented inside the component you need a way to forward messages with AMP payload to that component. This is what the 'amp' plugin does. The 'amp' plugin intercepts all <message/> packets even without AMP payload, redirecting some of the to the AMP component and others processing in a standard way. Therefore you no longer need 'message' plugin or 'msgoffline' plugin. Their all functions are offered by the 'amp' plugin now. Hence you have to switch 'message' and 'msgoffline' plugins off (the 'amp' plugin is loaded by default):

```
--sm-plugins=amp, -message, -msgoffline
```

The 'amp' plugin needs to know where to forward all the AMP packets. By default plugin uses hostname of the given machine as this is true to the most installations, however this is configured by the last line of the example configuration, which forwards all packets to the address 'amp@your-domain.tld [mailto:amp@your-domain.tld]':

```
sess-man/plugins-conf/amp/amp-jid=amp@your-domain.tld
```

Secondly: component.

By default Tigase loads the component with the standard name 'amp'

Optional parameters:

There is also one parameter shared between the component and the plugin. Connection to the database where offline messages are stored. The AMP component has a dedicated schema for storing offline messages designed for a high traffic and high load installations. It does not use UserRepository for storing messages.

By default the same physical database as for UserRepository is used but you can change it and store messages in a completely separate location to reduce performance degradation of rest of the system. You can set a database connection string using following property:

```
--amp-repo-uri=jdbc:mysql://localhost/tigasedb?user=db_usr&password=db_pwd
```

The XEP-0079 [<http://xmpp.org/extensions/xep-0079.html>] specification has a Section 9. - Security Considerations [<http://xmpp.org/extensions/xep-0079.html#security>]. As it describes, in some cases the AMP protocol can be used to reveal user's presence information by other users who are not authorised for presence updates. There are a few possible ways to prevent this.

The Tigase's implementation offers 3 modes to handle AMP requests to prevent revealing user's status to not authorised people:

```
--amp-security-level=STRICT
```

In this mode the server performs strict checking. The AMP specification is fully handled. This however involves roster loading for each offline user, hence it may impact the service performance. It may not be feasible or possible to run in this mode for services under a high load with lots of AMP messages.

In the XEP this mode is described in the following way:

*Accept the relevant condition only if the sender is authorized to receive the receiver's presence, as a result of which the server **MUST** reply with a <not-acceptable/> error condition if the sender is not so authorized; this is the **RECOMMENDED** behavior. This is also the default in Tigase.*

```
--amp-security-level=PERFORMANCE
```

Dummy checking is performed, that is no checking actually. It just returns an error response every time there is a chance that the default action may reveal user status without looking into the user's roster. This does not affect performance but it does impact the AMP compliance.

In the XEP this mode is described in the following way:

*Accept the relevant condition only if the action is "drop", as a result of which the server **MUST** reply with a <not-acceptable/> error condition if the action is "alert", "error", or "notify"; this is slightly less restrictive but still unnecessarily restricts the functionality of the system, so is **NOT RECOMMENDED**.*

It does not do any checking. It acts like all people are authorised to receive notifications, even if it may reveal user status to unauthorised people. It does not impact the server performance and it offers full AMP compliance.

```
--amp-security-level=NONE
```

---

# Chapter 54. Configuration SASL EXTERNAL

Bartosz Malkowski <bmalkowski@tigase.pl [mailto:bmalkowski@tigase.pl]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: <http://tigase.net> :Date: 2013-11-27 13:34

In order to enable SASL External add following line to the `init.properties` file

```
c2s/clientCertCA=/path/to/cacert.pem
```

File `cacert.pem` contains Certificate Authority certificate which is used to sign clients certificate.

Client certificate must include user's Jabber ID as `XmppAddr` in `subjectAltName`:

*As specified in RFC 3920 and updated in RFC 6120, during the stream negotiation process an XMPP client can present a certificate (a "client certificate"). If a JabberID is included in a client certificate, it is encapsulated as an `id-on-xmppAddr` Object Identifier ("`xmppAddr`"), i.e., a `subjectAltName` entry of type `otherName` with an ASN.1 Object Identifier of "`id-on-xmppAddr`" as specified in Section 13.7.1.4 of RFC 6120. [1]*

XEP-0178 [<http://xmpp.org/extensions/xep-0178.html#c2s>]

---

# Chapter 55. Server to Server Protocol Settings

Artur Hefczyc <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2010-06-29 21:27

The Tigase server **5.1.0** or later offers new, rewritten from scratch implementation for s2s communication which allows you to tweak it's configuration to get a better performance in your installation.

S2S (or server to server) protocol is enabled by default with setting which are optimal for the most common cases. There is however a set of configuration parameters you can adjust the server behaviour to get an optimal performance for your installation.

This documents describes following elements of the Tigase server configuration:

1. Number of concurrent connections to external servers
2. The connection throughput parameters
3. Maximum waiting time for packets addressed to external servers and the connection inactivity time
4. Custom plugins selecting connection to the remote server

## Number of Concurrent Connections

Normally only one connection to the remote server is required to send XMPP stanza to that server. In some cases however, under a high load, you can get much better throughput and performance if you open multiple connections to the remote server.

This is especially true when the remote server works in a cluster mode. Ideally you want to open a connection to each of the cluster nodes on the remote server. This way you can spread the traffic evenly among cluster nodes and improve the performance for s2s connections.

The Tigase server offers 2 different parameters to tweak the number of concurrent, s2s connections:

- 'max-out-total-conns' - the property specifies the maximum outgoing connections the Tigase server opens to any remote XMPP server. This is 'per domain' limit, which means that this limit applies to each of the remote domain the Tigase connects to. If it is set to '4' then Tigase opens maximum 4 connections to 'jabber.org' plus maximum 4 connections to 'muc.jabber.org' even if this is the same physical server behind the same IP address.

To adjust the limit you have to add following line to the init.properties file:

```
s2s/max-out-total-conns[I]=2
```

- 'max-out-per-ip-conns' - the property specifies the maximum outgoing connections the Tigase server opens to any remote XMPP server to its single IP address. This is 'per domain' limit, which means that this limit applies to each of the remote domain the Tigase connects to. If it is set to '1', above limit is set to '4' and the remote server is visible behind 1 IP address then Tigase opens maximum 1 connection to 'jabber.org' plus maximum 1 connection to 'muc.jabber.org', etc....

To adjust the limit you have to add following line to the init.properties file:

```
s2s/max-out-per-ip-conns[I]=2
```

## Connection Throughput

Of course everybody wants his server to run with maximum throughput. However this comes with cost, usually increased memory usage. This is especially important if you have huge number of s2s connections on your installations. High throughput means lots of memory for network buffers for every single s2s connection. You may soon run out of all available memory.

There is one configuration property which allows you to adjust the network buffers for s2s connections to lower your memory usage or increase data throughput for s2s communication.

More details about are available in the [initPropertiesGuide](#), [init.properties](#) guide under the link to `--net-buff-high-throughput` property description.

## Maximum Packet Waiting Time and Connection Inactivity Time

There are 2 timeouts you can set for the component controlling s2s communication.

- 'max-packet-waiting-time' - this sets the maximum time for the packets waiting for sending to some remote server. Sometimes, due to networking problems or DNS problems it might be impossible to send message to remote server right away. Establishing a new connection may take time or there might be communication problems between servers or perhaps the remote server is restarted. The Tigase will try a few times to connect to the remote server before giving up. This parameter specifies how long the packet is waiting for sending before it is returned to the sender with an error. The timeout is specified in seconds:

```
s2s/max-packet-waiting-time[L]=420
```

- 'max-inactivity-time' - this parameter specifies the maximum s2s connection inactivity time before it is closed. If the connection is not in use for a long time, perhaps it doesn't make sense to keep it open and it resources up. The Tigase closes s2s connection after specified period of time and reconnects when it is necessary. The timeout is specified in seconds:

```
s2s/max-inactivity-time[L]=900
```

## Custom Plugin Selecting s2s Connection

Sometimes for a very large installations you may want to set larger number of s2s connections to remote servers. Especially if they work in cluster of several nodes. In such a case you can also have a control over XMPP packets distribution among s2s connections to a single remote server.

This piece of code is pluggable and you can write your own connection selector. It is enough to implement 'S2SConnectionSelector' interface and set your class name in the configuration using following parameter in `init.properties` file:

```
s2s/s2s-conn-selector=YourSelectorImplementation
```

The default selector picks connections randomly.

---

# Chapter 56. Tigase Server Version 4.x

Artur Hefczyc <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2010-01-06 20:22

Administration manuals and guides for the Tigase server version 4.x line.

- [MySQL Database Schema Upgrade for Tigase 4.0](#)
- [Virtual Components for the Cluster Mode](#)
- [External Component Configuration](#)
- [Packets filtering](#)
- [Virtual Hosts in the Tigase Server](#)
- [Configuration](#)

---

# Chapter 57. MySQL Database Schema Upgrade for Tigase 4.0

Artur Hefczyc <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2010-01-06 20:18

For number of reasons the database schema had to be changed for Tigase server version 4.0. The most important are:

- Compliance with the XMPP RFC which says that each part of JID may have up to 1023 characters. We store in the database user JIDs without resource name thus the maximum possible size of the user id is 2047. There aren't really JIDs that long yet but we experienced quite long JIDs in a few installations already so we decided to prepare Tigase to accept any JID allowed by RFC.
- Performance and flexibility - the Tigase server now accesses database using stored procedures. This allows for any database storage format and it doesn't really matter for the Tigase server what is the database schema how data is organized inside. What it needs is just bunch of stored procedures to access the data. This allows for much more flexibility in storing user data as well as much easier integration with third-party systems and also organize data in more efficient way.

Therefore when you run the Tigase server now it may (depending on what exact SVN revision you use) refuse to start if it detects that the database schema is not updated. If it happens just follow steps below to update the database schema and start the server again. Updating of the database schema is very easy and almost fully automated process. Just follow the steps below and you should be able to run new version of the Tigase server in a few minutes or even seconds depending on your database size. It takes around 7 minutes to update database with 200k user accounts on an average machine.

**Note. Do not update the database schema before the Tigase server tells you to do so. And do a database backup before starting the schema update.**

*Please note. I have done a few schema upgrades already in a different configurations and here are a few tips which might be useful if something goes wrong:*

1. **You really, really have to do the DB backup (database dump) before upgrading.** *If you don't you might not be able to revert database on your own. Contact me in case of problems.*
2. *In case of error: **ERROR 1419 (HY000) at line 31 in file: 'database/mysql-schema-4-sp.schema': You do not have the SUPER privilege and binary logging is enabled (you \*might\* want to use the less safe log\_bin\_trust\_function\_creators variable)\*** Restore the database following description found below and run the update again as MySQL super user.*
3. *The following error may manifest itself in many ways from the `NullPointerException` in the Tigase server log file to message like this: **User does not have access to metadata required to determine stored procedure parameter types. If rights can not be granted, configure connection with "noAccessToProcedureBodies=true" to have driver generate parameters that represent INOUT strings irregardless of actual parameter types.** The best solution to this is to grant proper permissions to this user. Enter the MySQL command line mode as MySQL super user:*

```
$ mysql --u root --proot_passwd mysql
mysql> GRANT SELECT, INSERT, UPDATE ON `mysql`.`proc` TO -'tigase_user'@'loc
mysql> GRANT SELECT, INSERT, UPDATE ON `mysql`.`proc` TO -'tigase_user'@'%';
mysql> GRANT SELECT, INSERT, UPDATE ON `mysql`.`proc` TO -'tigase_user';
mysql> FLUSH PRIVILEGES;
```

\$

Assumptions:

1. **tigasedb** is a database name
2. **tigase\_user** is a database user name
3. **mypass** is database user password

First things first - make a database backup:

```
mysqldump --u tigase_user --pmypass tigasedb > tigasedb_dump.sql
```

If you need to restore database for any reason execute following commands:

```
mysqladmin --u tigase_user --pmypass drop tigasedb  
mysqladmin --u tigase_user --pmypass create tigasedb  
mysql --u tigase_user --pmypass tigasedb < tigasedb_dump.sql
```

*Note! You may be required to use **root** user and his password to execute mysqladmin commands. Ok we have the database backup and we know how to restore it. Now we can run schema upgrade script:*

```
mysql --u tigase_user --pmypass tigasedb < database/mysql-schema-upgrade-to-4.sql
```

*The script should generate output like this:*

```
Dropping index for user_id column  
Resizing user_id column to 2049 characters to comply with RFC  
Creating a new index for user_id column for first 765 bytes of the field  
Adding sha1_user_id column  
Adding user_pw column  
Adding last_login column  
Adding last_logout column  
Adding online_status column  
Adding failed_logins column  
Adding account_status column  
Creating a new index for user_pw column  
Creating a new index for last_login column  
Creating a new index for last_logout column  
Creating a new index for account_status column  
Creating a new index for online_status column  
Resizing node column to 255 characters  
Changing pval column type to mediumtext  
Loading stored procedures definitions  
Setting passwords encoding in the database  
Converting database to a new format  
Creating a new index for sha1_user_id column  
Setting schema version to 4.0  
All done, database ready to use!
```

---

# Chapter 58. Virtual Components for the Cluster Mode

Artur Hefczyk <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2010-04-06 21:18

Let's assume you have a cluster installation and you want to include a component in your installation which doesn't support the cluster mode yet. If you put it on all nodes as a separate instances they will work out of sync and overall functionality might be useless. If you put on one node only it will work correctly but it will be visible to users connected to this one node only.

Ideally you would like to have a mechanism to install it on one node and put some redirections on other nodes to forward all packets for this component to a node where this component is working. Redirection on it's own is not enough however because the component must be visible in service discovery list and must be visible somehow to users connected to all nodes.

This is where the virtual components are handy. They are visible to users as a local normal component, they seem to be a real local component but in fact they just forward all requests/packets to a cluster node where the real component is working.

Virtual component is a very lightweight ServerComponent implementation in the Tigase server. It can pretend to be any kind of component and can redirect all packets to a given address. They can mimic native Tigase components as well as third-party components connected over external component protocol (XEP-0114).

Configuration is very simple and straightforward. In fact it is very similar to configuration of any Tigase component. You set a real component name as a name of the component and a virtual component class name to load. Let's say we want to deploy MUC component this way. The MUC component is visible as muc.domain.our in the installation. Thus the name of the component is: muc

```
--comp-name-1=muc
--comp-class-1=tigase.cluster.VirtualComponent
```

This is pretty much all you need to load a virtual component. A few other options are needed to point to correct destination addresses for packets forwarding and to set correct service discovery parameters:

```
muc/redirect-to=muc@cluster-node-with-real-muc.domain.our
muc/disco-name=Multi User Chat
muc/disco-node=
muc/disco-type=text
muc/disco-category=conference
muc/disco-features=http://jabber.org/protocol/muc
```

That's it.

---

# Chapter 59. External Component Configuration

Artur Hefczyc <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2010-04-06

In the Tigase server version 4.4.x a new implementation for connecting external components has been introduced.

It is much simpler to setup and follows the same configuration standards as all other components. It is also much more powerful as a single instance can control many TCP/IP ports and many external components on each port and even allows for multiple connections for the same component. It supports both XEP-0114 and XEP-0225 with protocol auto-detection mechanisms. Protocols are pluggable so in future more protocols can be supported or custom extensions to existing protocols can be added.

The implementation also supports scripting API and new domains with passwords can be added at runtime using ad-hoc commands. New scripts can be loaded to even further control all connected external components.

Even though it is much simpler to setup and to use it also offers a lot of new functionality and features. Pages in this guide describe in details all the administration aspects of setting up and managing external components.

- [Basic Configuration Options \(External Component\)](#)
- [Tigase as External Component](#)
- [Load Balancing External Components in Cluster Mode](#)

---

# Chapter 60. Basic Configuration Options (External Component)

Artur Hefczyk <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2010-04-06 21:18

As for all Tigase components you can load it and configure it via `initPropertiesGuide` file described in details in `4xconfiguration` another guide. This document describes how to load the component and set the initial configuration to accept or initiate connections for an external component.

First thing to do is to specify the component class and the component name which must be unique within the Tigase installation. The most commonly name used is `ext` and the class is `tigase.server.ext.ComponentProtocol`.

Following 2 lines in the `initPropertiesGuide` will load the component during the server startup time:

```
--comp-name-1 = ext
--comp-class-1 = tigase.server.ext.ComponentProtocol
```

While this would load the component there is no additional configuration provided to the component would be practically useless. It is possible to add necessary parameters (external domains, passwords) during run-time via ad-hoc commands but this is generally a good practise to provide some initial parameters in the configuration file too.

There are two additional properties used for setting initial configuration for external components connections: `initProperties_external` and `initProperties_bind-ext-hostnames`.

These two properties are very well described on page under the given links, therefore I will focus on practical and working examples here.

## Simple Case

The most common scenario is to connect an external component which works for a specific, given domain to the main server. The component authenticates with a defined password and the external component connects to a TCP/IP port the server listens on.

For example let's say our server works for a virtual domain: `devel.tigase.org`. We want it to listen on port 5270 for incoming connections from an external component working for a domain: `muc.devel.tigase.org`. The authentication password for the domain is `muc-secret`.

For such a scenario we need 3 lines in the `init.properties` file:

```
--comp-name-1 = ext
--comp-class-1 = tigase.server.ext.ComponentProtocol
--external = muc.devel.tigase.org:muc-secret:listen:5270
```

## More External Components/Domains

Suppose you want to connect more than one external component. Let's say you want to connect PubSub and MSN components to the Tigase server as well.

In such a case you don't have to open another port on the server. All the components can connect to the same port. Of course each of the components connect for a different domain and probably should use a different password.

Let's say then that we want the Tigase server accept two more domains with corresponding passwords: (pubsub.devel.tigase.org:pubsub\_pass) and (msn.devel.tigase.org:msn\_pass). Your configuration properties should look like this:

```
--comp-name-1 = ext
--comp-class-1 = tigase.server.ext.ComponentProtocol
--external = muc.devel.tigase.org:muc-secret:listen:5270, \
  pubsub.devel.tigase.org:pubsub_pass, \
  msn.devel.tigase.org:msn_pass
```

*Please note, the --external property with value should be written in a single line. Above example has split the line for readability.*

## More TCP/IP Ports

You can make the Tigase to listen on more than one TCP/IP port for incoming external component connections. Please be aware however that there is no way, currently to bind an external component to a particular port. If the Tigase listens on two or more ports it accepts any external component on any of the ports. Therefore there is no practical reason for opening more than one port.

However, if for some reason you need Tigase to listen on more ports then this is an example configuration:

```
--comp-name-1 = ext
--comp-class-1 = tigase.server.ext.ComponentProtocol
--external = muc.devel.tigase.org:muc-secret:listen:5270, \
  pubsub.devel.tigase.org:pubsub_pass:listen:5271, \
  msn.devel.tigase.org:msn_pass:listen:5272
```

*Please note, the --external property with value should be written in a single line. Above example has split the line for readability.*

Above setting sets three TCP/IP ports to listen on: 5270, 5271 and 5272. It also specifies 3 different external domains with passwords which are accepted by the Tigase. Even though each port is specified with conjunction with a domain they are not bound together in any way. Any of specified domains can connect through any of specified ports.

## Outgoing Connections

The Tigase server can, not only, accept connections from external components, it can also open connections to external components.

To make the Tigase connect to external component you have to change 'listen' parameter to 'connect' and of course you have to tell where to connect - the address of the external component:

```
--comp-name-1 = ext
--comp-class-1 = tigase.server.ext.ComponentProtocol
--external = devel.tigase.org:muc-secret:connect:5270:muc.devel.tigase.org
```

Assuming the MUC component listens on the port '5270' at the DNS address: 'muc.devel.tigase.org' the Tigase will connect to the component.

You can of course set as many as you need external components to which you want the Tigase to connect to:

```
--comp-name-1 = ext
--comp-class-1 = tigase.server.ext.ComponentProtocol
--external = devel.tigase.org:mucsecret:connect:5270:muc.devel.tigase.org, \
  devel.tigase.org:pubsub_pass:connect:5271:pubsub.devel.tigase.org, \
  devel.tigase.org:msn_pass:connect:5272:msn.devel.tigase.org
```

If external components run on a separate machines you can of course us the same port number for each of them.

## Specifying Protocol

One of the last parameters you can set for the external component/domain is a protocol which should be used for the connection. At the moment the Tigase server supports two protocols defined in XEP-0114 [<http://xmpp.org/extensions/xep-0114.html>] and XEP-0225 [<http://xmpp.org/extensions/xep-0225.html>] and possibly in the future it may support more.

You don't have to specify a protocol if you setup a connection in 'listen' mode as the server automatically detects a protocol which is used in incoming connections.

You can, however specify the protocol which is used for outgoing connections. You have to add one more parameter to the connection string. There are two possibilities:

1. 'accept' which is an identifier for protocol defined in XEP-0114 [<http://xmpp.org/extensions/xep-0114.html>] (and is a default if you do not specify anything)
2. 'client' which is identifier for protocol defined in XEP-0225 [<http://xmpp.org/extensions/xep-0225.html>] and is based on the client-to-server protocol.

An example configuration with protocol specified:

```
--comp-name-1 = ext
--comp-class-1 = tigase.server.ext.ComponentProtocol
--external = devel.tigase.org:mucsecret:connect:5270:muc.devel.tigase.org:accept,
  devel.tigase.org:pubsub_pass:connect:5270:pubsub.devel.tigase.org:client
```

It defines two outgoing connections to external protocols, the first uses XEP-0114 [<http://xmpp.org/extensions/xep-0114.html>] protocol and the second uses XEP-0225 [<http://xmpp.org/extensions/xep-0225.html>].

## Load Balancer Plugin

The last option you can set for external component connections is load balancer class.

The load balancer plugin is used if you have multiple connections for the same component (external domain name) and you want to spread load over all connections. Perhaps you have an installation with huge number of MUC rooms and you want to spread the load over all MUC instances.

An example configuration with load balancer plugin specified:

```
--comp-name-1 = ext
--comp-class-1 = tigase.server.ext.ComponentProtocol
--external = muc.devel.tigase.org:mucsecret:listen:5270:devel.tigase.org:accept:Re
  pubsub.devel.tigase.org:pubsub_pass:listen:5270:devel.tigase.org:accept:SenderBa
```

It defines two listeners for external component with different load balancer plugins. The first load-balance traffic by a packet destination BareJID, which makes sense for MUC component. This way each MUC instance handles a different set of rooms which allows for a good load distribution.

For PubSub component we use a different load balancer plugin which distributes load by the sender BareJID instead. This is because for PubSub destination BareJID is always the same so we cannot use it to distribute the load.

Either the **ReceiverBareJidLB** or **SenderBareJidLB** are class names from package: **tigase.server.ext.lb** however, you can use any class name as a plugin, you just have to provide a full class name and the class name must implement **LoadBalancerIfc** interface.

---

# Chapter 61. Tigase as External Component

Artur Hefczyc <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2010-04-06 21:18

There are cases when you want to deploy one or more Tigase components separately from the main server or perhaps you want to run some Tigase components connecting to a different XMPP server or perhaps you work on a component and you do not want to restart the main server every time you make a change.

There is a way to run the Tigase server in "external component mode". In fact you can run any of Tigase components as an external component and connect them to the main XMPP server either via XEP-0114 [<http://xmpp.org/extensions/xep-0114.html>] or XEP-0225 [<http://xmpp.org/extensions/xep-0225.html>] connection.

Let's look at the examples...

## A Simple Case - MUC as an External Component

A few assumptions:

1. We want to run a MUC component for a domain: 'muc.devel.tigase.org' and password 'muc-pass'
2. The main server works at an address: devel.tigase.org and for the same virtual domain
3. We want to connect to the server using XEP-0114 [<http://xmpp.org/extensions/xep-0114.html>] protocol and port '5270'.

There is a special configuration type for this case which simplifies setting needed to run the Tigase as an external component:

```
config-type=---gen-config-comp.
```

This generates a configuration for the Tigase with only one component loaded by default - component used for external component connection. If you use this configuration type, your init.properties file may look like this:

```
config-type = ---gen-config-comp
--debug = server
--user-db = derby
--admins = admin@devel.tigase.org
--user-db-uri = jdbc:derby:/tigasedb
--virt-hosts = devel.tigase.org
--comp-name-1 = muc
--comp-class-1 = tigase.muc.MUCComponent
--external = muc.devel.tigase.org:muc-pass:connect:5270:devel.tigase.org:accept
```

Please note, you do not need lines:

```
--comp-name-1 = ext
```

```
--comp-class-1 = tigase.server.ext.ComponentProtocol
```

as the

```
--gen-config-comp
```

automatically includes them.

## More Components

Suppose you want to run more components as external components within one Tigase instance. Let's add another - PubSub component to the configuration above and see how to set it up.

The way which seems to be most straightforward is just to add another component and another connection to the main server for the component domain:

```
config-type = ---gen-config-comp
--debug = server
--user-db = derby
--admins = admin@devel.tigase.org
--user-db-uri = jdbc:derby:/tigasedb
--virt-hosts = devel.tigase.org
--comp-name-1 = muc
--comp-class-1 = tigase.muc.MUCComponent
--comp-name-2 = pubsub
--comp-class-2 = tigase.pubsub.PubSubComponent
--external = muc.devel.tigase.org:muc-pass:connect:5270:devel.tigase.org:accept, \
pubsub.devel.tigase.org:pubsub-pass:connect:5270:devel.tigase.org:accept
```

Please note however that we are opening two connections to the same server. This is kind of wasting resources and overcomplicating system. What if we want to run even more components? Opening a separate connection for each component seems to be an overkill.

In fact there is a way to reuse the connection for all component domains running as an external component. The property '--bind-ext-hostnames' contains a comma separated list of all hostnames (external domains) which should reuse the existing connection.

There is one catch however. Connection reusing (hostname binding is defined in XEP-0225 [<http://xmpp.org/extensions/xep-0225.html>]) only, hence you must use this protocol for the functionality.

Here is an example configuration with a single connection over XEP-0225 [<http://xmpp.org/extensions/xep-0225.html>] protocol used by both external domains:

```
config-type = ---gen-config-comp
--debug = server
--user-db = derby
--admins = admin@devel.tigase.org
--user-db-uri = jdbc:derby:/tigasedb
--virt-hosts = devel.tigase.org
--comp-name-1 = muc
--comp-class-1 = tigase.muc.MUCComponent
--comp-name-2 = pubsub
--comp-class-2 = tigase.pubsub.PubSubComponent
--external = muc.devel.tigase.org:muc-pass:connect:5270:devel.tigase.org:client
--bind-ext-hostnames=pubsub.devel.tigase.org
```

---

# Chapter 62. Load Balancing External Components in Cluster Mode

Artur Hefczyc <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2011-07-09 22:45

This document describes how to load balance any external components using Tigase XMPP Server and how to make Tigase's components work as external components in a cluster mode.

*Please note, all configuration options described here apply to Tigase XMPP Server version 5.1.0 or later.*

These are actually 2 separate topics:

1. One is to distribute load over many instances of a single component to handle larger traffic, or perhaps for high availability.
2. The second is to make Tigase's component work as an external component and make it work in a cluster mode, even if the component itself does not support cluster mode.

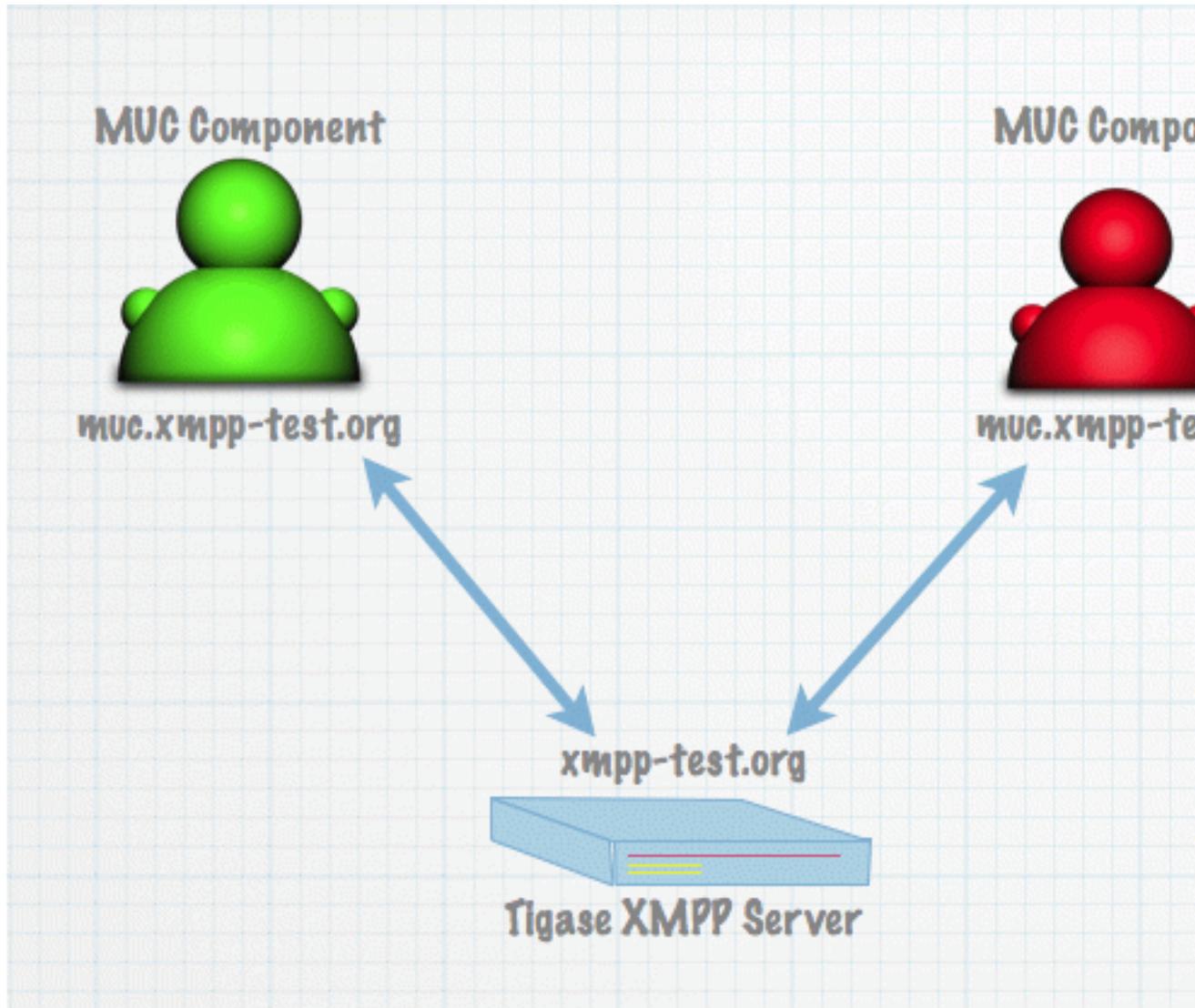
Here are step by step instructions and configuration examples teaching how to achieve both goals

## Load Balancing External Component

The first, simple scenario is to connect multiple instances of an external component to a single Tigase XMPP Server to distribute load.

Why would you like to do that? There are at least 2 reasons, one would be to spread load over more instances/machines and the second is to improve reliability in case one component fails the other one can take over the work.

So here is a simple picture showing the use case.



We have a single machine running Tigase XMPP Server and 2 instances of the MUC component connecting to the Tigase. First example configurations.

The server configuration

```
config-type = ---gen-config-def
--user-db = mysql
--admins = admin@devel.tigase.org
--user-db-uri = jdbc:mysql://localhost/db?user=tigase&password=tigase
--virt-hosts = devel.tigase.org

--comp-name-1 = ext
--comp-class-1 = tigase.server.ext.ComponentProtocol
--external = muc.devel.tigase.org:muc-secret:listen:\
            5270:devel.tigase.org:accept:ReceiverBareJidLB
```

And configuration for both instances of the MUC component (identical for both of them):

```
config-type = ---gen-config-comp
```

```
--user-db = mysql
--admins = admin@devel.tigase.org
--user-db-uri = jdbc:mysql://localhost/db?user=tigase&password=tigase
--virt-hosts = devel.tigase.org

--comp-name-1 = muc
--comp-class-1 = tigase.muc.MUCComponent
--external = muc.devel.tigase.org:muc-secret:connect:\
            5270:devel.tigase.org:accept
```

I guess everything is quite clear and simple to those who are familiar with Tigase's configuration except one small element in the Tigase server configuration. At the end of the component connection we have **ReceiverBareJidLB**.

This is load balancing plugin class. Load balancing plugin decides how the traffic is distributed among different component connections that is different component instances. For the MUC component it does make sense to distribute the traffic based on the receiver bare JID because this is the MUC room address. This way we just distribute MUC rooms and traffic over different MUC component instances.

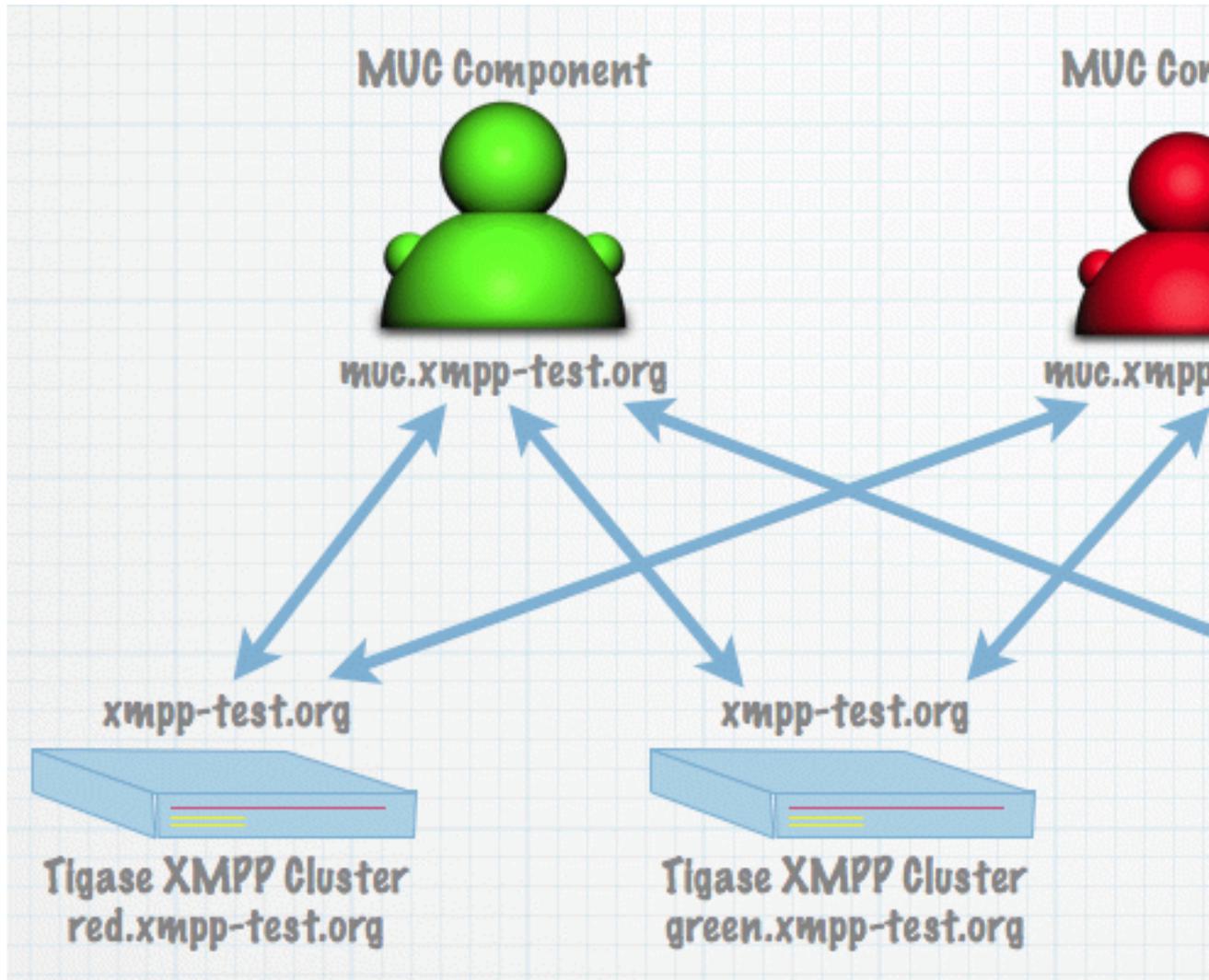
This distribution strategy does not always work for all possible components however. For transports for example this would not work at all. A better way to spread load for transports would be based on the source bare JID. And it is possible of course if you use plugin with class name: **SenderBareJidLB**.

These are two basic load distribution strategies available now. For some use cases none of them is good enough. If you have PubSub, then you probably want to distribute load based on the PubSub node. There is no plugin for that yet but it is easy enough to write one and put the class name in configuration.

## External Component and Cluster

If you want to use Tigase's component in a cluster mode which does not have clustering implemented yet there is a way to make it kind of cluster-able. In the previous section we connected many MUC components to a single Tigase server. Now we want to connect a single MUC component to many Tigase servers (or many Tigase cluster nodes).

Let's say we have Tigase XMPP Server working for domain: **xmpp-test.org** and the server is installed on three cluster nodes: **red.xmpp-test.org**, **green.xmpp-test.org** and **blue.xmpp-test.org**.



We want to make it possible to connect the MUC component to all nodes, so here is configuration for the server (for each node is the same):

```
config-type=--gen-config-def
--admins=admin@xmpp-test.org
--virt-hosts = xmpp-test.org

--cluster-mode = true
--cluster-nodes=red.xmpp-test.org,green.xmpp-test.org,blue.xmpp-test.org

--auth-db=tigase-auth
--user-db=mysql
--user-db-uri=jdbc:mysql://localhost/db?user=tigase&password=tigase

--comp-name-1 = ext
--comp-class-1 = tigase.server.ext.ComponentProtocol
--external = muc.xmpp-test.org:muc-secret:listen:5270:\
            xmpp-test.org:accept:ReceiverBareJidLB
```

As you can see there is nothing special here. The most interesting part comes on the MUC side.

## Load Balancing External Components in Cluster Mode

---

```
config-type = ---gen-config-comp
--user-db = mysql
--admins = admin@xmpp-test.orgg
--user-db-uri = jdbc:mysql://localhost/db?user=tigase&password=tigase

--virt-hosts = xmpp-test.org

--comp-name-1 = muc
--comp-class-1 = tigase.muc.MUCComponent
--external = muc.xmpp-test.org:muc-secret:connect:5270:xmpp-test.org;blue.xmpp-tes
```

As you can see remote host name is not a simple domain but a character string with a few comma separated parts. The first part is our remote domain and the rest are addresses of the host to connect to. This can be a list of domain names or IP addresses.

Of course it is possible to connect multiple external component to all cluster nodes, this way the whole installation would be really working in the cluster and also load balanced.

---

# Chapter 63. Packet Filtering

Tigase offers different ways to filter XMPP packets flying through the server. The most common case to use packet filtering is to restrict users from sending or receiving packets based on the sender or received address.

There are also possible different scenarios: time based filtering, content filtering, volume filtering and so on.

All pages in this section describe different filtering strategies.

## Domain Based Packet Filtering

Domain based packets filter is a simple filter allowing to restrict user communication based on the source/destination domain name. This is especially useful if we want to limit user communication within a single - own domain only or a list of domains.

A company might not wish and allow employers to chat during work hours with anybody in the world. A company may also have a few different domains used by different branches or departments. An administrator may restrict communication to a list of domains.

## Introduction

The restriction is on per-user basis. So the administrator can set a different filtering rules for each user. There is also a per-domain configuration and global-installation setting (applied from most general to most specific, i.e. from installation to user).

Regular user can not change the setting. So this is not like a privacy list where the user control the filter. Domain filter can not be changed or controlled by the user. System administrator can change the settings based on the company policy.

There are predefined rules for packet filtering:

1. ALL - user can send and receive packets from anybody.
2. LOCAL - user can send and receive packets within the server installation only and all it's virtual domains.
3. OWN - user can send and receive packets within his own domains only
4. BLOCK - user can't communicate with anyone. This could be used as a means to temporarily disable account or domain.
5. LIST - user can send and receive packets within listed domains only (i.e. *whitelist*).
6. BLACKLIST - user can communicate with everybody (like ALL), except contacts on listed domains.

Whitelist (LIST) and blacklist (BLACKLIST) settings are mutually exclusive, i.e. at any given point of time only one of them can be used.

Those rules applicable to particular user are stored in the user repository and are loaded for each user session. If there are no rules stored for a particular user server tries to apply rules for a VHost of particular user, and if there is no VHost filtering policy server uses global server configuration. If there is no filtering policy altogether server applies defaults based on following criteria:

1. If this is **Anonymous** user then **LOCAL** rule is applied

2. For all **other** users **ALL** rule is applied.

## Configuration

Filtering is performed by domain filter plugin which must be loaded at startup time. It is loaded by default if plugins list is not set in the configuration file. However if you have a list of loaded plugins in the configuration file make sure **domain-filter** is on the list.

There is no other configuration required for the plugin to work.

## Administration, Rules Management

Although controlling domain filtering rules are possible for each user separately it is normally not practical for large installations. In most cases users are stored in the database and third-party system keeps all the user information.

To change the rule for a single user you can use loadable administration scripts feature and load `UserDomainFilter.groovy` [<https://projects.tigase.org/projects/tigase-server/repository/revisions/master/entry/src/main/groovy/tigase/admin/UserDomainFilter.groovy>] script. It allows to modify rules for a given user JID.

## Implementation

If you have a third party system which keeps and manages all user information than you probably have your own `UserRepository` implementation which allows the Tigase server to access user data. Filtering rules are loaded from user repository using following command:

```
repo.getData(user_id, null, DomainFilter.ALLOWED_DOMAINS_KEY, null)
repo.getData(user_id, null, DomainFilter.ALLOWED_DOMAINS_LIST_KEY, null)
```

Where **user\_id** is user Jabber ID without resource part. **DomainFilter.ALLOWED\_DOMAINS\_KEY** is a property key: "allowed-domains". The user repository **MUST** return one of following only:

1. **ALL** - if the user is allowed to communicate with anybody
2. **LOCAL** - if the user is allowed to communicate with users on the same server installation.
3. **OWN** - if the user is allowed to communicate with users within his own domain only.
4. **LIST** - list of domains within which the user is allowed to communicate with other users. No wild-cards are supported. User's own domain should be included too.
5. **BLACKLIST** - list of domains within which the user is **NOT** allowed to communicate with other users. No wild-cards are supported. User's own domain should **NOT** be included.
6. **null** - a java null if there are no settings for the user.

In case of **LIST** and **BLACKLIST** filtering options, it's essential to provide list of domains for the whitelisting/blacklisting. **DomainFilter.ALLOWED\_DOMAINS\_LIST\_KEY** is a property key: "allowed-domains-list". The user repository **MUST** return semicolon separated list of domains: `domain1.com;domain2.com, domain3.org`

The filtering is performed by `tigase.xmpp.impl.DomainFilter` [<https://projects.tigase.org/projects/tigase-server/repository/revisions/master/entry/src/main/java/tigase/xmpp/impl/DomainFilter.java>] plugin. Please refer to source code for more implementation details.

---

# Chapter 64. Virtual Hosts in the Tigase Server

Artur Hefczyc <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: <http://tigase.net> :Date: 2010-04-06 21:18

The Tigase server supports multiple virtual hosts for a single server installation. This is supported via `VHostManager` [<http://projects.tigase.org/server/trac/browser/trunk/src/main/java/tigase/vhosts/VHostManagerIfc.java>] - the new Tigase server component added recently to the implementation. Virtual hosts can be added or removed, enabled or disabled at the server runtime without restarting the service or disrupting normal operation.

This document describes how virtual hosts work in the Tigase server and how to take the most of this feature in your installation.

The simplest and default way to set virtual hosts is the server configuration. You can either edit manually the `////<<initPropertiesGuide,////init.properties` file or use the graphical installer/configuration program to set the property. If you want to edit it manually search for '--virt-hosts' property for more detailed description.

Alternatively you can use the GUI installer as shown below to set a list of virtual hosts.

IzPack - Installation of Tigase XMPP (Jabber)

## Basic Tigase server configuration

On this panel you can specify basic configuration options. Based on your selection here more configuration options will be shown. After the configuration is complete init.properties will be generated. You can optionally restart the server at the end of the installation.


Configuration type	Default installation
Your XMPP (Jabber) domain	devel.tigase.org
Server administrators	test1@devel.tigase.org
Select database	Derby (built-in database)

Advanced configuration options

(Made with IzPack - <http://izpack.org/>)

Step 13 of 21

This method however has many disadvantages. It requires the server restart after each change, the configuration file is not the best place to store long list of virtual domains and you can not actually set any additional parameters for the domain other than it does exist or not.

There is another way to store and control virtual domains in the Tigase server. They can be put in the database and managed using ad-hoc commands. List of domains can be modified outside the Tigase server through any third-party system or web application and the server reloads the list of when received **VHOSTS\_RELOAD** ad-hoc command.

There are 2 more ad-hoc commands which allow you to add/update and remove virtual hosts via XMPP protocol:

- **VHOSTS\_UPDATE** - for adding new virtual host or changing parameters of the existing domain
- **VHOSTS\_REMOVE** - for removing existing virtual domain from the list of the server domains.

By default, both commands cause vhosts list update in the permanent repository. This is however `VHostRepository` [<http://projects.tigase.org/server/trac/browser/trunk/src/main/java/tigase/vhosts/VHostRepository.java>] implementation dependent feature and can be changed in your repository implementation.

Commands for virtual domains management can be executed using any XMPP client with a good support for service discovery and ad-hoc commands, for example Psi [<http://psi-im.org/>]. Commands are accepted only when they are sent by the service administrator.

Please refer to documents listed below for more detailed information on the following topics:

- [Managing Virtual Domains Using Psi Client](#).
- [Specification for ad-hoc Commands Used to Manage Virtual Domains](#).
- [API Description for Virtual Domains Management in the Tigase Server](#).

---

# Chapter 65. Specification for ad-hoc Commands Used to Manage Virtual Domains

Artur Hefczyc <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2010-04-06 21:18

There are 3 ad-hoc commands for virtual domains management in the Tigase server:

1. **VHOSTS\_RELOAD** used to reload virtual domains list from the repository (database).
2. **VHOSTS\_UPDATE** used to add a new virtual domain or update information for existing one.
3. **VHOSTS\_REMOVE** used to remove an existing virtual host from the running server.

Syntax of the commands follows specification described in the XEP-0050 [<http://xmpp.org/extensions/xep-0050.html>]. Extra information required to complete the command is carried as data forms described in the XEP-0004 [<http://xmpp.org/extensions/xep-0004.html>].

All commands are accepted by the server only when send by the installation administrator. If the command is sent from any other account &lt;not-authorized /&gt; error is returned. To grant administrator rights to an account you have to set **--admins** property in the `////<<initPropertiesGuide,////` configuration file.

Commands are sent to 'vhost-man' server component and the 'to' attribute of the stanza must contain a full JID of the VHostManager on the server. The full **JID** consists of the component name: 'vhost-man' and the local domain, that is domain which is already on the list of virtual domains and is active. Assuming 'existing.domain.com' one of domains already activated for the server installation the **JID** is: 'vhost-man@existing.domain.com [mailto:vhost-man@existing.domain.com]'.

## Reloading the Domains List from the Database

In order to reload virtual domains from the permanent repository other than configuration file you have to send **VHOSTS\_RELOAD** ad-hoc command to the VHostManager on the server.

The reload command request is of the form:

```
<iq type="set"
  to="vhost-man@existing.domain.com"
  id="aac8a">
  <command xmlns="http://jabber.org/protocol/commands"
    node="VHOSTS_RELOAD" -/>
</iq>
```

The server sends a response upon successful completion of the command with current number of virtual domains server by the installation:

```
<iq from="vhost-man@existing.domain.com"
  type="result"
  to="cmd-sender-admin@existing.domain.com"
  id="aac8a">
  <command xmlns="http://jabber.org/protocol/commands"
```

```
        status="completed"
        node="VHOSTS_RELOAD">
<x xmlns="jabber:x:data" type="result">
  <field type="fixed" var="Note">
    <value>Current number of VHosts: 123</value>
  </field>
</x>
</command>
</iq>
```

If the command is sent from other than admin account the server returns an error:

```
<iq from="vhost-man@existing.domain.com"
  type="error"
  to="cmd-sender-admin@existing.domain.com"
  id="aac8a">
<command xmlns="http://jabber.org/protocol/commands"
  node="VHOSTS_RELOAD" -/>
<error type="auth" code="401">
  <not-authorized xmlns="urn:ietf:params:xml:ns:xmpp-stanzas" -/>
  <text xmlns="urn:ietf:params:xml:ns:xmpp-stanzas"
    xml:lang="en">
    You are not authorized for this action.
  </text>
</error>
</iq>
```

The response doesn't have any special meaning other than informative for the end-user. The client may ignore response as it is sent after the command has been executed.

## Adding a New Domain or Updating Existing One

In order to add a new domain or update existing one you have to send an ad-hoc command **VHOSTS\_UPDATE** with at least domain name in the command data form. You can also specify whether the domain is enabled or disabled but this is optional. Future releases may allow for setting additional parameters for the domain: maximum number of user accounts for this domain, anonymous login enabled/disabled for the domain, registration via XMPP enabled/disabled for this domain and some more parameters not specified yet.

The domain add/update command request is of the form:

```
<iq type="set"
  to="vhost-man@existing.domain.com"
  id="aacba">
<command xmlns="http://jabber.org/protocol/commands"
  node="VHOSTS_UPDATE">
<x xmlns="jabber:x:data" type="submit">
  <field type="text-single"
    var="VHost">
    <value>new-virt.domain.com</value>
  </field>
  <field type="list-single"
```

```
        var="Enabled">
        <value>>true</value>
    </field>
</x>
</command>
</iq>
```

Please note! Character case in the command field variable names does matter.

Upon successful completion of the command the server sends a response back to the client with information of the existing number of virtual hosts on the server:

```
<iq from="vhost-man@existing.domain.com"
  type="result"
  to="cmd-sender-admin@existing.domain.com"
  id="aacba">
  <command xmlns="http://jabber.org/protocol/commands"
    status="completed"
    node="VHOSTS_UPDATE">
    <x xmlns="jabber:x:data" type="result">
      <field type="fixed" var="Note">
        <value>Current number of VHosts: 124</value>
      </field>
    </x>
  </command>
</iq>
```

## Removing a Virtual Domain From the Server

In order to remove a virtual domain you have to send **VHOSTS\_REMOVE** command to the server with the domain name.

The domain remove command is sent by the client:

```
<iq type="set"
  to="vhost-man@existing.domain.com"
  id="aacba">
  <command xmlns="http://jabber.org/protocol/commands"
    node="VHOSTS_REMOVE">
    <x xmlns="jabber:x:data" type="submit">
      <field type="text-single"
        var="VHost">
        <value>virt-nn.domain.com</value>
      </field>
    </x>
  </command>
</iq>
```

Upon successful completion of the command the server sends a response back to the client with information of the existing number of virtual hosts on the server:

```
<iq from="vhost-man@existing.domain.com"
  type="result"
  to="cmd-sender-admin@existing.domain.com"
```

```
id="aacba">
<command xmlns="http://jabber.org/protocol/commands"
  status="completed"
  node="VHOSTS_REMOVE">
  <x xmlns="jabber:x:data" type="result">
    <field type="fixed" var="Note">
      <value>Current number of VHosts: 124</value>
    </field>
  </x>
</command>
</iq>
```

---

# Chapter 66. Configuration

Artur Hefczyc <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2010-04-06 21:18

This section contains some specific description for non-standard Tigase installations.

- `////<<loadComponent,////Configuring the Tigase Server to Load a Component`
- `////<<customAuthentication,////Custom Authentication Connectors`
- `////<<custonAuthConnector,////Tigase Custom Auth Connector`
- `////<<LDAPauth,////LDAP Authentication Connector`
- `////<<tigaseAuthConnector,////Tigase Auth Connector`

---

# Chapter 67. Configuring the Tigase Server to Load a Component

Artur Hefczyc <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2010-04-06 21:18

A detailed description of all the configuration options is in the `init.properties` guide where you can also find information described below and much more. Purpose of this document however is to give you a quite and brief information how to load a component into the Tigase server without need to dig through all the details.

I will show how to load 2 components into the Tigase server using configuration in the `init.properties` file: MUC [<https://projects.tigase.org/projects/tigase-muc>] and PubSub [<https://projects.tigase.org/projects/tigase-pubsub>]. Please remember, every time you change something in the **init.properties** file you have to remove the XML configuration file in order to force the server to regenerate the main configuration which is stored in XML file.

The first thing you need is the component implementation. Component implementation is a class or set of classes extending `tigase.server.AbstractMessageReceiver` [<http://projects.tigase.org/server/trac/browser/trunk/src/main/java/tigase/server/AbstractMessageReceiver.java>]. What you need to do is just putting the jar file in the `libs/` directory in the Tigase server installation. Then the Tigase server will find all classes automatically at the startup time.

Next step is to tell the server what components to load, how to name them and optionally give some extra parameters. To do so please open the `init.properties` file you use in your installation. It might be `init-mysql.properties` or `init-pgsql.properties` or even your own properties file.

Let's say you want to add just PubSub for now. All you need to do is adding just 2 lines to the properties file:

```
--comp-name-1=pubsub
--comp-class-1=tigase.pubsub.PubSubComponent
```

They mean: the first component name is 'pubsub' and the main class for this component is: 'tigase.pubsub.PubSubClusterComponent'. It doesn't really matter what the component name is the only requirement is that it must be unique among other components names. It does also help to give it a name which means something thus 'pubsub' is a good name for a 'PubSub' component but it would be a bad name for the 'MUC' component.

We can of course add more components even PubSub components to the same server. Each of them would need to have a different name then. For example:

```
--comp-name-2=pubsub-priv
--comp-class-2=tigase.pubsub.PubSubComponent
```

Which is needed in really rare cases.

Normally, however we want to load few different components like PubSub, MUC, MSN Transport and so on.... Therefore instead of the above second PubSub we can load the MUC component:

```
--comp-name-2=muc
--comp-class-2=tigase.muc.MUCComponent
```

Again! Don't forget to remove your XML config file before restarting the server.

---

# Chapter 68. Custom Authentication Connectors

Artur Hefczyk <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2010-04-06 21:18

Tigase server offers you quite a few authentication connectors which allow you to connect to almost any SQL database for user authentication data and share user accounts between the XMPP server and any different system. This feature makes it possible to integrate the Tigase server with other systems without any development effort and without any coding.

This article presents configuration options available to the administrator and describe how to set the Tigase server up to use user accounts data from a different database.

The first thing to know is that the Tigase server always opens 2 separate connections to the database. One connection is for user login data and another is for all other user data like the user roster, vCard, private data storage, privacy lists and so on...

In this article we still assume that the Tigase server keeps user data in it's own database and only login data are retrieved from the external database.

At the moment the Tigase server offers following authentication connectors:

- 'mysql', 'pgsql', 'derby' - standard authentication connector used to load user login data from the main user database used by the Tigase server. In fact the same physical implementation is used for all JDBC databases.
- 'drupal' - is the authentication connector used to integrate the Tigase server with Drupal CMS [<http://drupal.org/>].
- 'libresource' - is the authentication connector used to integrate the Tigase server with Libresource Collaboration platform [<http://dev.libresource.org/>].
- 'tigase-auth' - is the authentication connector which can be used with any database. It executes stored procedures to perform all actions. Therefore it is a very convenient way to integrate the server with an external database if you don't want to expose the database structure. You just have to provide a set of stored procedures in the database. While implementing all stored procedures expected by the server might be a bit of work it allows you to hide the database structure and change the SP implementation at any time. You can add more actions on user login/logout without restarting or touching the server. And the configuration on the server side is very simple. For detailed description of this implementation please refer to [Tigase Auth documentation](#).
- 'tigase-custom' - is the authentication connector which can be used with any database. Unlike the 'tigase-auth' connector it allows you to define SQL queries in the configuration file. The advantage of this implementation is that you don't have to touch your database. You can use either simple plain SQL queries or stored procedures. The configuration is more difficult as you have to enter carefully all SQL queries in the config file and changing the query usually involves restarting the server. For more details about this implementation and all configuration parameters please refer to [Tigase Custom Auth documentation](#).

As always the simplest way to configure the server is through the [initProperties-Guide](#) file. In the article describing this file you can find long list with all available options and all details how to handle it. For the authentication connector setup however we only need 2 options:

- '--auth-db = connector'
- '--auth-db-uri = database connection url'

If you happen to keep the user data in the same database as user authentication data you can even skip the second parameter as Tigase automatically assumes settings from the '--user-db-uri' if '--auth-db-uri' is missing.

'--auth-db-uri' stored a standard JDBC connection URL and is exactly the same as for all other settings. For example if you store authentication data in the 'drupal' database on 'localhost' the URL might look like:

```
--auth-db-uri = jdbc:mysql://localhost/drupal?user=user&password=passwd
```

'--auth-db' stored just a connector name or connector implementation class. For convenience the Tigase has predefined short names for the most common connectors but you can always use the class name if you know it. And you have to use a class name if you want to attach your own authentication connector. The following 2 settings are equal:

```
--auth-db = tigase-auth  
  
--auth-db = tigase.db.jdbc.TigaseAuth
```

In the same exact way you can setup connector for any different database type:

```
--auth-db = drupal  
  
--auth-db = tigase-custom
```

You can normally skip configuring connectors for the default Tigase database format: 'mysql', 'pgsql' and 'derby' as they are applied automatically if the parameter is missing.

One more important thing to know is that you also have to modify '--user-db-uri' if you use a custom authentication connector. This is because if you retrieve user login data from the external database this external database is usually managed by external system. User accounts are added without notifying the Tigase server. Then, when the user logs in and tries to retrieve the user roster the server can not find such a user in the roster database.

To keep user accounts in sync between authentication database and the main user database you have to add following option to the end of the database connection URL: 'autoCreateUser=true'.

For example:

```
--user-db-uri=jdbc:mysql://localhost/tigasedb?user=nobody&password=pass&autoCreateUser=true
```

If you are interested in even further customize you authentication connector by writing your own queries or stored procedures please have a look at 2 following guides:

- [////<<tigaseAuthConnector,////Tigase Auth guide](#)
- [////<<custonAuthConnector,////Tigase Custom Auth guide](#)

---

# Chapter 69. Tigase Custom Auth Connector

Artur Hefczyc <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2010-04-06 21:18

The Tigase Custom Auth connector with shortcut name: **tigase-custom** is implemented in the class: `tigase.db.jdbc.TigaseCustomAuth` [<http://projects.tigase.org/server/trac/browser/trunk/src/main/java/tigase/db/jdbc/TigaseCustomAuth.java>]. It allows you to connect to any external database to perform user authentication and use a custom queries for all actions..

You can find more details how to setup a custom connector in [////<<customAuthentication,////Custom Authentication Connectors guide](#).

The basic configuration is very simple:

```
--auth-db = tigase-custom
--auth-db-uri = jdbc:mysql://localhost/drupal?user=user&password=passwd
```

That's it.

The connector loads correctly and starts working using predefined, default list of queries. In most cases you also want to define your own queries in the configuration file. The shortest possible description is the following example of the content from [////<<initPropertiesGuide,////init.properties file](#):

```
# This query is used to check connection to the database, whether it is still alive
basic-conf/auth-repo-params/conn-valid-query=select 1

# This is database initialization query, normally we do not use it, especially in
# clustered environment
basic-conf/auth-repo-params/init-db-query=update tig_users set online_status = 0

# Below query performs user authentication on the database level.
# The Tigase server does not need to know authentication algorithm or password
# encoding type, it simply passes user id (BareJID) and password in form
# which was received from the client, to the stored procedure. If the
# authentication was successful the procedure returns user bare JID or null otherwise
# The Tigase checks whether the JID returned from the query matches
# JID passed as a parameter. If they match, the authentication is successful.
basic-conf/auth-repo-params/user-login-query={ call TigUserLoginPlainPw(?, -?) -}

# Below query returns number of user accounts in the database, this is mainly used
# for the server metrics and monitoring components.
basic-conf/auth-repo-params/users-count-query={ call TigAllUsersCount() -}

# Below query is used to add a new user account to the database
basic-conf/auth-repo-params/add-user-query={ call TigAddUserPlainPw(?, -?) -}

# Below query is used to remove existing account with all user's data from the database
basic-conf/auth-repo-params/del-user-query={ call TigRemoveUser(?) -}

# This query is used for the user authentication if "-user-login-query" is not defined
```

```
# that is if there is no database level user authentication algorithm available. I
# a case the Tigase server loads user's password from the database and compares it
# with data received from the client.
basic-conf/auth-repo-params/get-password-query=select user_pw from tig_users where

# Below query is used for user password update in case user decides to change his
basic-conf/auth-repo-params/update-password-query=update tig_users set user_pw = -

# Below query is called on user logout event. Usually we use a stored procedure wh
# records user logout time and marks user as offline in the database
basic-conf/auth-repo-params/user-logout-query=update tig_users, set online_status

# This is configuration setting to specify what non-sasl authentication mechanisms
# expose to the client
basic-conf/auth-repo-params/non-sasl-mechs=password,digest

# This is configuration setting to specify what sasl authentication mechanisms exp
basic-conf/auth-repo-params/sasl-mechs=PLAIN,DIGEST-MD5
```

Queries are defined in the configuration file and they can be either plain SQL queries or stored procedures. If the query starts with characters: '{ call' then the server assumes this is a stored procedure call, otherwise it is executed as a plain SQL query. Each configuration value is stripped from white characters on both ends before processing.

Please don't use semicolon ';' at the end of the query as many JDBC drivers get confused and the query may not work for unknown reason.

Some queries take arguments. Arguments are marked by question marks '?' in the query. Refer to the configuration parameters description for more details about what parameters are expected in each query.

The first example shows how to put a stored procedure as a query with 2 required parameters.

```
add-user-query={ call TigAddUserPlainPw(?, -?) -}
```

The same query with plain SQL parameters instead:

```
add-user-query=insert into users (user_id, password) values (?, -?)
```

The order of the query arguments is important and must be exactly as described in specification for each parameter.

- 'conn-valid-query' - Query executing periodically to ensure active connection with the database.

Takes no arguments.

Example query: 'select 1'

- 'init-db-query' - Database initialization query which is run after the server is started.

Takes no arguments.

Example query: 'update tig\_users set online\_status = 0'

- 'add-user-query' - Query adding a new user to the database.

Takes 2 arguments: (user\_id (JID), password)

Example query: 'insert into tig\_users (user\_id, user\_pw) values (?, ?)'

- 'del-user-query' - Removes a user from the database.

Takes 1 argument: (user\_id (JID))

Example query: 'delete from tig\_users where user\_id = ?'

- 'get-password-query' - Retrieves user password from the database for given user\_id (JID).

Takes 1 argument: (user\_id (JID))

Example query: 'select user\_pw from tig\_users where user\_id = ?'

- 'update-password-query' - Updates (changes) password for a given user\_id (JID).

Takes 2 arguments: (password, user\_id (JID))

Example query: 'update tig\_users set user\_pw = ? where user\_id = ?'

- 'user-login-query' - Performs user login. Normally used when there is a special SP used for this purpose. This is an alternative way to a method requiring retrieving user password. Therefore at least one of those queries must be defined: user-login-query or get-password-query.

If both queries are defined then user-login-query is used. Normally this method should be only used with plain text password authentication or sasl-plain.

The Tigase server expects a result set with user\_id to be returned from the query if login is successful and empty results set if the login is unsuccessful.

Takes 2 arguments: (user\_id (JID), password)

Example query: 'select user\_id from tig\_users where (user\_id = ?) AND (user\_pw = ?)'

- 'user-logout-query' - This query is called when user logs out or disconnects. It can record that event in the database.

Takes 1 argument: (user\_id (JID))

Example query: 'update tig\_users, set online\_status = online\_status - 1 where user\_id = ?'

- 'non-sasl-mechs' - Comma separated list of NON-SASL authentication mechanisms. Possible mechanisms are: password and digest. digest mechanism can work only with get-password-query active and only when password are stored in plain text format in the database.
- 'sasl-mechs' - Comma separated list of SASL authentication mechanisms. Possible mechanisms are all mechanisms supported by Java implementation. The most common are: PLAIN, DIGEST-MD5, CRAM-MD5.

"Non-PLAIN" mechanisms will work only with the get-password-query active and only when passwords are stored in plain text format in the database. Application: Tigase Server

---

# Chapter 70. LDAP Authentication Connector

Artur Hefczyc <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2012-03-30 21:56

From version 5.1.0, rev. (build) 2881 Tigase XMPP Server offers support for authenticating users against LDAP server in **Bind Authentication** mode.

Configuration for the LDAP support is really simple. You have to add a few lines to your init.properties file.

```
# LDAP Authentication connector
--auth-db = tigase.db.ldap.LdapAuthProvider
# LDAP connection URI
--auth-db-uri=ldap://ldap.tigase.com:389
# LDAP access parameters
basic-conf/auth-repo-params/user-dn-pattern=cn=USER_ID,ou=people,dc=tigase,dc=org
```

Please note the **USER\_ID** element, this is a special element of the configuration which is used to authenticate particular user. Tigase LDAP connector replaces it with appropriate data during authentication. You can control what Tigase should put into this part. In your configuration you must replace this string with one of the following:

1. **%1\$s** - use user name only for authentication (JabberID's localpart)
2. **%2\$s** - use domain name only for authentication (JabberID's domain part)
3. **%3\$s** - use the whole Jabber ID (JID) for authentication

---

# Chapter 71. Tigase Auth Connector

Artur Hefczyc <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: <http://tigase.net> :Date: 2010-04-06 21:18

The Tigase Auth connector with shortcut name: **tigase-auth** is implemented in the class: `tigase.db.jdbc.TigaseAuth` [<http://projects.tigase.org/server/trac/browser/trunk/src/main/java/tigase/db/jdbc/TigaseAuth.java>]. It allows you to connect to any external database to perform user authentication. You can find more details how to setup a custom connector in `////<<customAuthentication,////Custom Authentication Connectors` guide.

To make this connector working you have to prepare your database to offer set of stored procedures for the Tigase server to perform all the authentication actions. The best description is the example schema with all the stored procedures defined. Please refer to the Tigase SVN repository [<http://projects.tigase.org/server/trac/browser/trunk/database/>] for the schema definition files.

Files with the stored procedures implementations are located in `mysql-schema-4-sp.schema` [<http://projects.tigase.org/server/trac/browser/trunk/database/mysql-schema-4-sp.schema>] for MySQL database and in `postgresql-schema-4-sp.schema` [<http://projects.tigase.org/server/trac/browser/trunk/database/postgresql-schema-4-sp.schema>] file for PostgreSQL database. You can also refer to the tables definition files to see how database is organized in our implementation: `mysql-schema-4.sql` [<http://projects.tigase.org/server/trac/browser/trunk/database/mysql-schema-4.sql>] file for MySQL database and `postgresql-schema-4.sql` [<http://projects.tigase.org/server/trac/browser/trunk/database/postgresql-schema-4.sql>] file for PostgreSQL database.

The absolute minimum of stored procedures you have to implement is:

- **TigUserLoginPlainPw** - to perform user authentication. The procedure is always called when the user tries to login to the XMPP server. This is the only procedure which must be implemented and actually must work.
- **TigUserLogout** - to perform user logout. The procedure is always called when the user logouts or disconnects from the server. This procedure must be implemented but it can be empty and can do nothing. It just needs to exist because Tigase expect it to exist and attempts to call it.

With these 2 above stored procedures you can only perform user login/logout on the external database. You can't register a user account, change user password or remove the user. In many cases this is fine as all the user management is handled by the external system.

If you however want to allow for account management via XMPP you have to implement also following procedures:

- **TigAddUserPlainPw** - to add a new user account
- **TigRemoveUser** - to remove existing user account
- **TigUpdatePasswordPlainPw** - to change a user password for existing account

---

# Chapter 72. Tigase Server Version 3.x

Artur Hefczyc <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2010-04-06 21:16

Admin manuals and guides for the Tigase server version 3.x line.

- [Connecting the Tigase Server to MySQL Database](#)
- [Integrating Tigase Server with Drupal](#)
- [Integrating Tigase Server with LibreSource](#)
- [Windows Installation](#)
- [Instalacja Pod Windows](#)
- [Configuration](#)

---

# Chapter 73. Connecting the Tigase Server to MySQL Database

Artur Hefczyc <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2010-04-06 21:16

Please before continuing reading of this manual have a look at the [initial MySQL database setup](#). It will help you with database preparation for connecting the Tigase server.

The easiest way to setup the Tigase server for connecting with MySQL database is to use so called [configuration wizards](#) (configuration generators) which release you from manually editing the XML configuration file and allow you quickly regenerate the XML configuration file in case of problems.

The article above describes the older way for using configuration generators which is a bit more difficult and doesn't work on Windows system. The guide below describes a new way to use them which is simpler and can be applied for Windows systems as well. It is using `init.properties` file where you can put all your initial configuration parameters.

This guide describes MySQL database connection parameters.

Well the guide is actually very short as there are example configuration files which can be used and customized for your environment.

Unfortunately these files are not included yet in the server version 3.x binary release and you have to download them from the SVN repository using following links:

1. `tigase-mysql.conf` [<https://svn.tigase.org/repos/tigase-server/trunk/etc/tigase-mysql.conf>] - the Tigase server startup file. The only difference from the default one is that it points to the file described below to load initial parameters.
2. `init-mysql.properties` [<https://svn.tigase.org/repos/tigase-server/trunk/etc/init-mysql.properties>] - the file contains a few initial parameters which can/should be adjusted to your environment. Here is a content of the file with each line described:

```
# Load standard set of the server components.
# Look at the http://www.tigase.org/configuration-wizards
# document for other possible values. Normally you don't
# need to change this line.
config-type=--gen-config-def
# List of administrator accounts, please replace them with
# administrator accounts in your installation
--admins=admin@tigase.org,admin@test-d
# The line says that the database used by the Tigase server is -'mysql'
# Look at the configuration wizards article for different options
# You can also put here a Java class name if you have a custom
# implementation for a database connector.
--user-db=mysql
# The line contains the database connection string. This is database
# specific string and for each kind of database it may look differently.
# Below string is for MySQL database. Please modify it for your system.
# MySQL connector requires connection string in the following format:
# jdbc:mysql://[hostname]/[database name]?user=[user name]&password=[user passwo
```

## Connecting the Tigase Server to MySQL Database

---

```
--user-db-uri=jdbc:mysql://localhost/tigasedb?user=tigase_user&password=mypass
# Virtual domains for your server installation, comma separated list of vhosts
--virt-hosts=tigase.org,test-d,localhost
# Select what packages you want to have logging switched for
# The below setting is recommended for the initail setup and it is required
# when asking for help with setting the server up
--debug=server
```

Download both files and put them to your etc/ directory.

Edit the init-mysql.properties for your environment.

Remove the XML configuration file.

Start the server using following command:

```
./bin/tigase.sh start etc/tigase-mysql.conf
```

*Ask more questions if you got stuck or need any help with this.*

---

# Chapter 74. Integrating Tigase Server with Drupal

Artur Hefczyc <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2010-04-06 21:16

Tigase supports integration with Drupal on many levels. At the moment all stuff described below work with Drupal version 4.x and 5.x. They may also work with Drupal 6.x but this version hasn't been tested.

First of all it can authenticate users against Drupal database which means you have the same user account for both Drupal website and XMPP (Jabber) server. More over in such a configuration all account management is done via Drupal web interface like account creation, password change update user details and so on. Administrator can temporarily disable user account and this is respected by the Tigase server too.

## Connecting to Drupal Database

The best way to setup Tigase with Drupal database is via configuration wizards that is `init.properties` file where you can put initial setting for Tigase configuration.

If you look in `etc/` directory of your Tigase installation you should find a few files there. The one we want to base our configuration on is: `init-mysql.properties`. In some older packages the file might be missing, then you can download it from the SVN repository [<https://svn.tigase.org/repos/tigase-server/trunk/etc/init-mysql.properties>].

If you look inside the file and strip all the comments you would see following lines:

```
config-type=--gen-config-def
--admins=admin@tigase.org,admin@tigase.net
--user-db=mysql
--user-db-uri=jdbc:mysql://localhost/tigasedb?user=tigase_user&password=myspass
--virt-hosts=tigase.org,tigase.net
--debug=server
```

All you need to connect to Drupal database are 2 following lines:

```
--auth-db=drupal
--auth-db-uri=jdbc:mysql://localhost/drupal?user=drupalusr&password=drupalpass
```

Let's combine it in a single file called `etc/init-drupal.properties`:

```
config-type=--gen-config-def
--admins=admin@tigase.org,admin@tigase.net
--auth-db=drupal
--auth-db-uri=jdbc:mysql://localhost/drupal?user=drupalusr&password=drupalpass
--user-db=mysql
--user-db-uri=jdbc:mysql://localhost/tigasedb?user=tigase_user&password=myspass
--virt-hosts=tigase.org,tigase.net
--debug=server
```

In theory you can load Tigase database schema to Drupal database and then both db-uris would have the same database connection string. More details about setting up and connecting to MySQL database can be found in the `////<<prepareMysql,//// MySQL` guide.

You need also to edit `etc/tigase.conf` file to point to your newly created properties file. The last line which looks like this:

```
TIGASE_OPTIONS=" ---property-file etc/init.properties -"
```

should be changed to following:

```
TIGASE_OPTIONS=" ---property-file etc/init-drupal.properties -"
```

Make sure you have Java-6 installed and `JAVA_HOME` is set to correct location. If `JAVA_HOME` is not set you can add following line to `etc/tigase.conf` file:

```
JAVA_HOME="/your/java-6-location"
```

The best way to check whether the variable is set correctly is with command:

```
$ ls --l $JAVA_HOME/bin/java
```

should list you the file without error.

Check also if `logs/` directory is created in your Tigase server location as sometimes unpacking application doesn't create empty directories.

Now when you have the configuration file and databases ready to use you can start the Tigase server with the following command:

```
./bin/tigase.sh start etc/tigase.conf
```

If you run the Tigase server installed from sources the command would be:

```
./scripts/tigase.sh start etc/tigase.conf
```

There are system startup scripts for Gentoo and Mandriva Linux on `bin/` or `scripts/` directory which can be used to automatically start the server when system starts.

Now you can register an account on your Drupal website and connect with Jabber client using the account details.

**Note!** *You have to enable plain password authentication in your Jabber client to connect to the Tigase server with Drupal database*

---

# Chapter 75. Integrating Tigase Server With LibreSource

Artur Hefczyc <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2010-04-06 21:16

**This document is still not finished.**

Taken directly from LibreSource [<http://www.libresource.org/>] page:

*LibreSource is a collaborative platform dedicated to both software development and community building. Based on Java/J2EE technology, LibreSource is a modular web server that users can customize online by combining resources and rights: wiki pages, forum, trackers, files, download areas, etc. All the tools are included and integrated.*

## Short Introduction.

Integration between **Tigase** server and **LibreSource** is on database level. It is implemented in the same way as integration with **Drupal** but with slightly more functions available.

Basically LibreSource system maintains own database with all it's data and Tigase connects to LibreSource database to authenticate users. All users data specific to Jabber/XMPP service are kept in separate tables which can be located in the same database as LibreSource data or in different database.

Current list of features included in the integration:

- Jabber users authentication against user data stored in LibreSource database.
- Recording Jabber user on-line status in LibreSource database. This can be displayed on the Web page as additional user onfo.
- Recording user last login time and this information can also be available on Web page.
- Checking user account status. So if the user account is disabled in LibreSource system then this user will not be able to login to Jabber/XMPP service too.
- User account creation. This feature might be useful during testing time or user data transition from old Tigase installation to LibreSource system. *Please note! This feature should be normally disabled on live system. All user account management should be done from LibreSource system because of data caching.*
- User account deletion. *Please note! This feature should be normally disabled on live system. All user account management should be done from LibreSource system because of data caching.*

A few assumptions:

1. LibreSource data are kept in PostgreSQL database - libresource and database user is demo.
2. For use cases where Tigase data are stored in MySQL database name is tigase, database user is dbuser and database user password is dbpass

## How to set Tigase up.

Now we will focus on setting things up to have both services up and running together. Below is example of the most complex environment to run where LibreSource is using PostgreSQL database and Tigase is

using MySQL database. All basic user data needed for authentication are kept by LibreSource so Tigase has to connect to PostgreSQL database too to authenticate users.

1. First you need LibreSource system up and running. Please refer to LS documentation for details.
2. Install Tigase server in the normal way including loading Tigase database schema. Database tables used by Tigase server use different naming convention so you can simply load Tigase DB schema to the same database as LibreSource data. It is also possible and recommended to keep Tigase data in separate database.
3. Using configuration wizards generate configuration for Tigase server to connect to LibreSource database. Here is the sample file with parameters for configuration wizard assuming following setup:

- LibreSource data are kept in PostgreSQL database: libresource, user: demo
- Tigase data are kept in MySQL database: tigase, user: dbuser, password: dbpass
- No external components are connected to Tigase server
- Tigase works for domain: domain.net

```
ENC="-Dfile.encoding=UTF-8 --Dsun.jnu.encoding=UTF-8"
DRV="-Djdbc.drivers=com.mysql.jdbc.Driver:org.postgresql.Driver"

JAVA_OPTIONS="${ENC} ${DRV} --server --Xms100M --Xmx100M -"
TIGASE_CONFIG="etc/tigase-mysql-libresource.xml"
TIGASE_OPTIONS="--gen-config-def \
  ---user-db mysql \
  ---user-db-uri jdbc:mysql://localhost/tigase?user=dbuser&password=dbpass&autoCreateTables=true \
  ---auth-db libresource \
  ---auth-db-uri jdbc:postgresql://localhost/libresource?user=demo \
  ---virt-hosts domain.net,localhost -"
```

4. And simpler example where all data (LibreSource and Tigase) are stored in the same database:

- LibreSource data are kept in PostgreSQL database: libresource, user: demo
- Tigase data are kept also in PostgreSQL database: libresource, user: demo
- No external components are connected to Tigase server
- Tigase works for domain: domain.net

```
ENC="-Dfile.encoding=UTF-8 --Dsun.jnu.encoding=UTF-8"
DRV="-Djdbc.drivers=com.mysql.jdbc.Driver:org.postgresql.Driver"

JAVA_OPTIONS="${ENC} ${DRV} --server --Xms100M --Xmx100M -"
TIGASE_CONFIG="etc/tigase-mysql-libresource.xml"
TIGASE_OPTIONS="--gen-config-def \
  ---user-db pgsq \
  ---user-db-uri jdbc:postgresql://localhost/libresource?user=demo&autoCreateTables=true \
  ---auth-db libresource \
  ---auth-db-uri jdbc:postgresql://localhost/libresource?user=demo \
  ---virt-hosts domain.net,localhost -"
```

Now, you can run Tigase in usual way and all works.

**Note!** In any case you have to load Tigase database schema for user data. Please refer to guide for specific database: `////<<mysql2database,////MySQL` or `////<<postgresDB2,////PostgreSQL`.

## Migration From old Tigase Installation to LibreSource

Tigase package includes additional tools to make it easier to manage and control you installation. One is used to change configuration settings - `config.sh` and another is used to manipulate user data in repository - `repo.sh`.

Depending on whether you use Tigase version built from sources or binary version these scripts might be available in either `scripts/` or `bin/` subdirectory. To make things simpler let's assume they are stored in `scripts/` directory.

Assuming you have old Tigase server installation with number of users in MySQL database and you want to migrate all of them to LibreSource there are 2 steps involved:

1. User data migration
2. Changing your existing configuration

### Data Migration

First we need to migrate user data used for authentication. That data will be used by both services: *LibreSource* and *Tigase* and they normally stored in *LibreSource* database. Therefore we have to use *LibreSource* database connector to handle the data (write or read). *Tigase* server will be using *LibreSource* database for reading only but during migration time we need to write user accounts to LS database. Sample command to migrate user accounts looks like this:

```
./scripts/repo.sh --sc tigase.db.jdbc.JDBCRepository \  
  --su -"jdbc:mysql://localhost/tigase?user=dbuser&password=dbpass" \  
  --dc tigase.db.jdbc.LibreSourceAuth \  
  --du -"jdbc:postgresql://localhost/libresource?user=demo" \  
  --cp
```

Above command will copy all user accounts from MySQL *tigase* database to *libresource* database. Please refer to repository management tool documentation for information how to migrate single or selected user accounts.

If you want to also keep all Tigase server data in the same database you have to copy also all other user data like rosters, vCards and so on.

First thing to do we have to load database schema for Tigase data. You don't have to worry. Tigase tables have distinct names from LibreSource so there is no danger for any conflict. As in example above let's assume LibreSource data are stored in *libresource* database and database user name is *demo*:

```
psql --q --U demo --d libresource --f database/postgresql-schema.sql
```

Now we can load transfer all user data from our MySQL database to LibreSource:

```
./scripts/repo.sh --sc tigase.db.jdbc.JDBCRepository \  
  --su -"jdbc:mysql://localhost/tigase?user=dbuser&password=dbpass" \  
  --dc tigase.db.jdbc.JDBCRepository \  
  --du -"jdbc:postgresql://localhost/libresource?user=demo" \  
  \
```

--cp

This command looks almost the same as a previous one. Just Java class used for handling destination database is different.

---

# Chapter 76. Windows Installation

Artur Hefczyk <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: <http://tigase.net> :Date: 2010-04-06 21:16

Tigase server installation The server installation should be started with downloading the Tigase server from our download area [<http://www.tigase.org/pl/filebrowser/tigase-server>]. This guide describes installation procedure for the branch 3.x of the server so please pick up the latest version of the Tigase 3.x.

The Windows binary package is an executable file containing installer. Run the file and the server will be installed and icons will be added to your Windows start menu. Locate the Tigase group in your menu and execute: "Install Tigase service" which will install the Tigase server as the system service and it will be automatically started whenever your system starts.

If you are going to use the Tigase server with MySQL database have a look now in the directory where the Tigase server is installed. There is a directory etc/. Have a look inside and find file called init.properties. Open the file with a text editor and make sure you added there 2 following lines:

```
--user-db=mysql  
--user-db-uri=jdbc:mysql://localhost/tigasedb?user=tigase_user&password=mypass
```

The content of the file should look like the example screenshot below:

```
config-type=--gen-config-def  
--admins=admin@localhost  
--virt-hosts = localhost  
--debug=server  
--user-db=mysql  
--user-db-uri=jdbc:mysql://localhost/tigase?user=sieniek&password=sieniek35
```

## MySQL Database Installation

The section describes installation and configuration of the MySQL database to work with Tigase server.

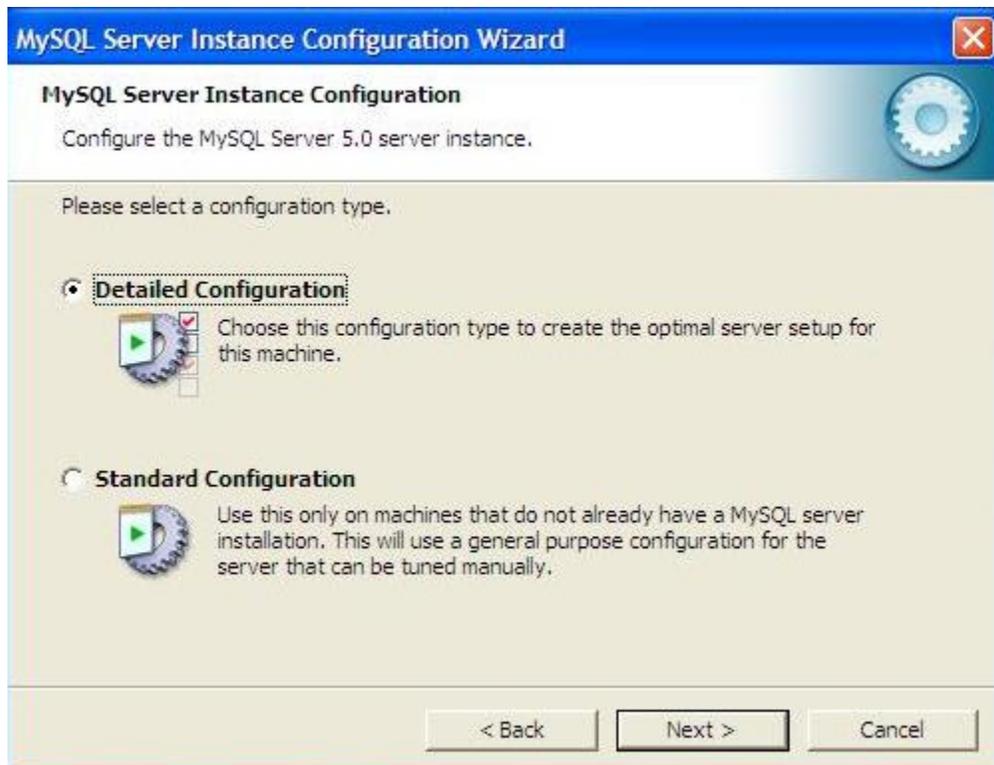
Download the binary package from MySQL download area at [mysql.com](http://dev.mysql.com/downloads/mysql/5.0.html#win32) [<http://dev.mysql.com/downloads/mysql/5.0.html#win32>]. Make sure you select executable proper for your operating system.

Run the installation program and follow default installation steps. When the installation is complete find the MySQL elements in the Windows Start menu and run the MySQL Configuration Wizard. Follow the wizard and make sure to check settings with screenshots in guide below.

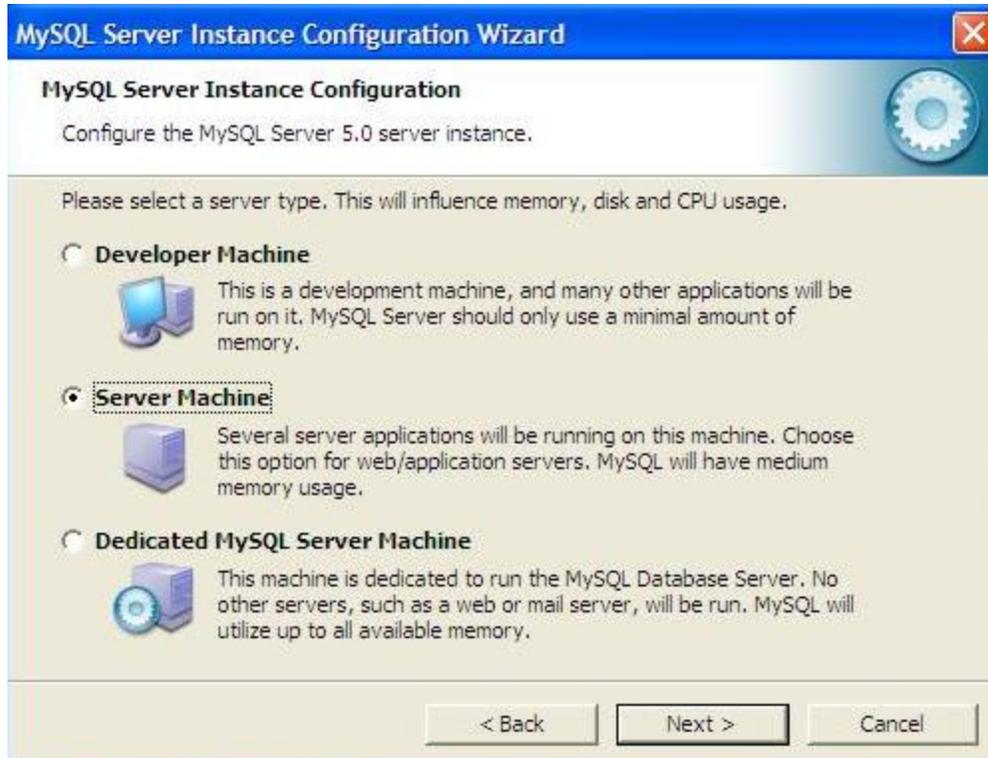
In Welcome window just press 'Next'.(pic.1)



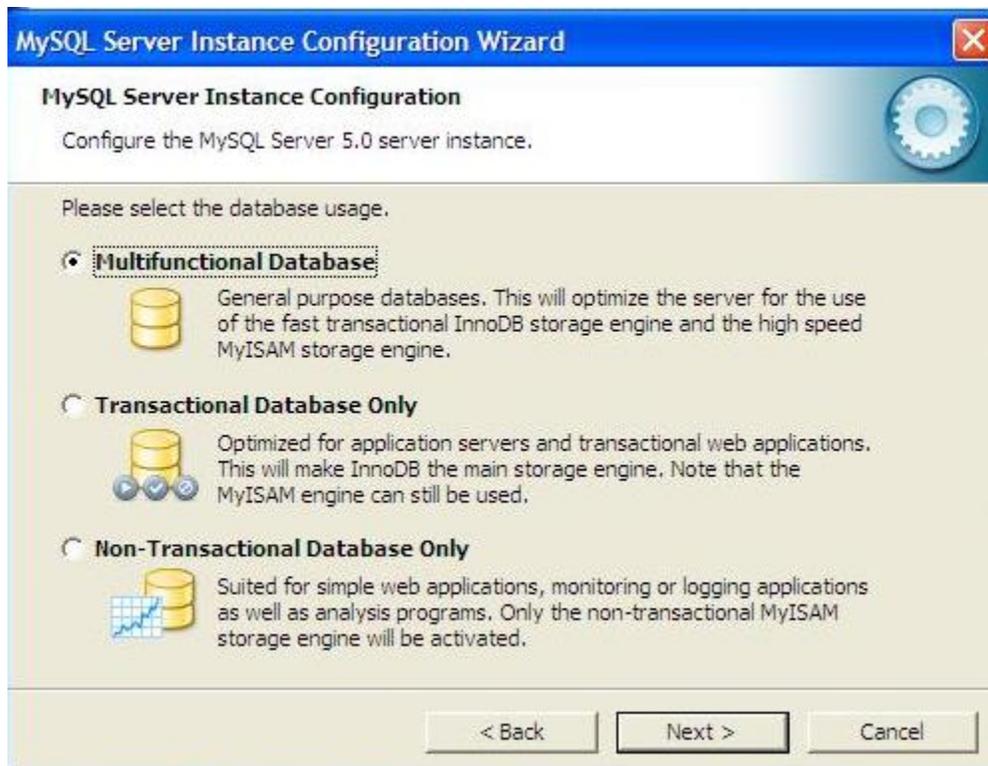
In the next window select option: 'Detailed Configuration' and press 'Next' (pic. 2)



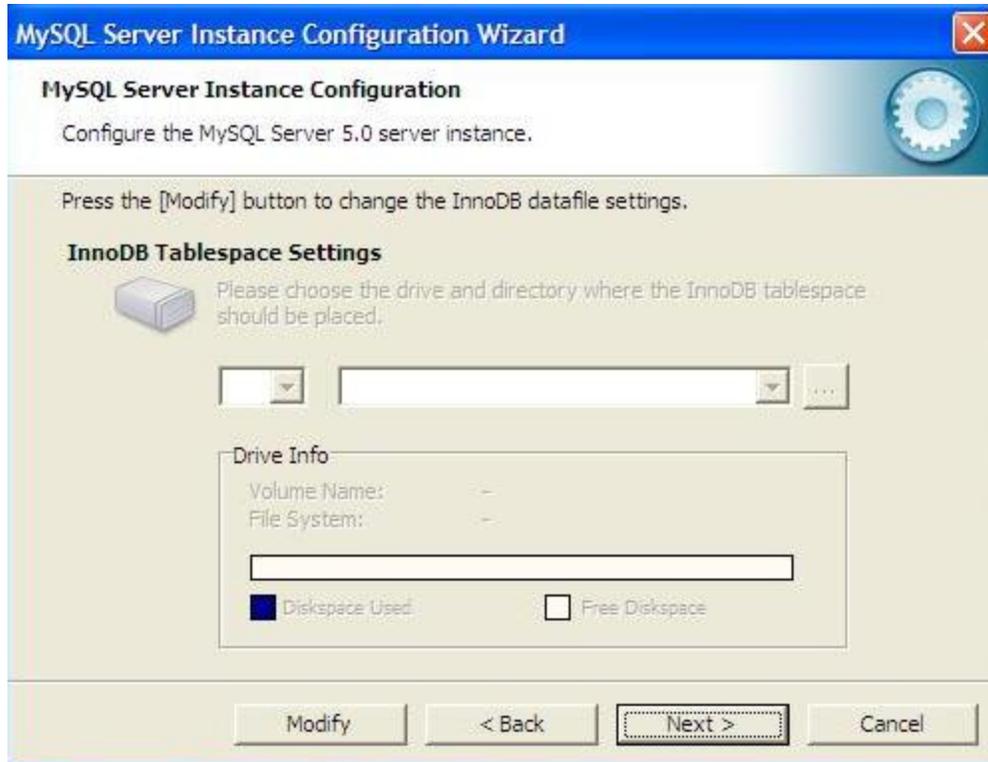
On the next screen select option: 'Server Machine' and press 'Next' (pic. 3)



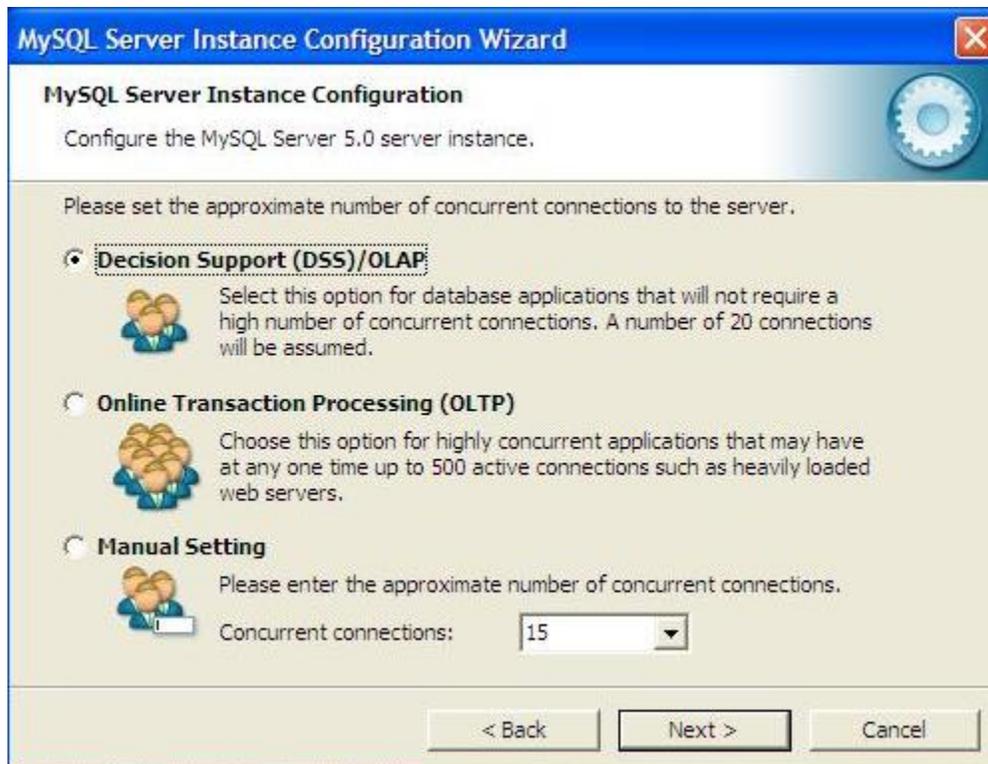
On the fourth window leave the default "Multifunctional Database" and press 'Next' (pic. 4)



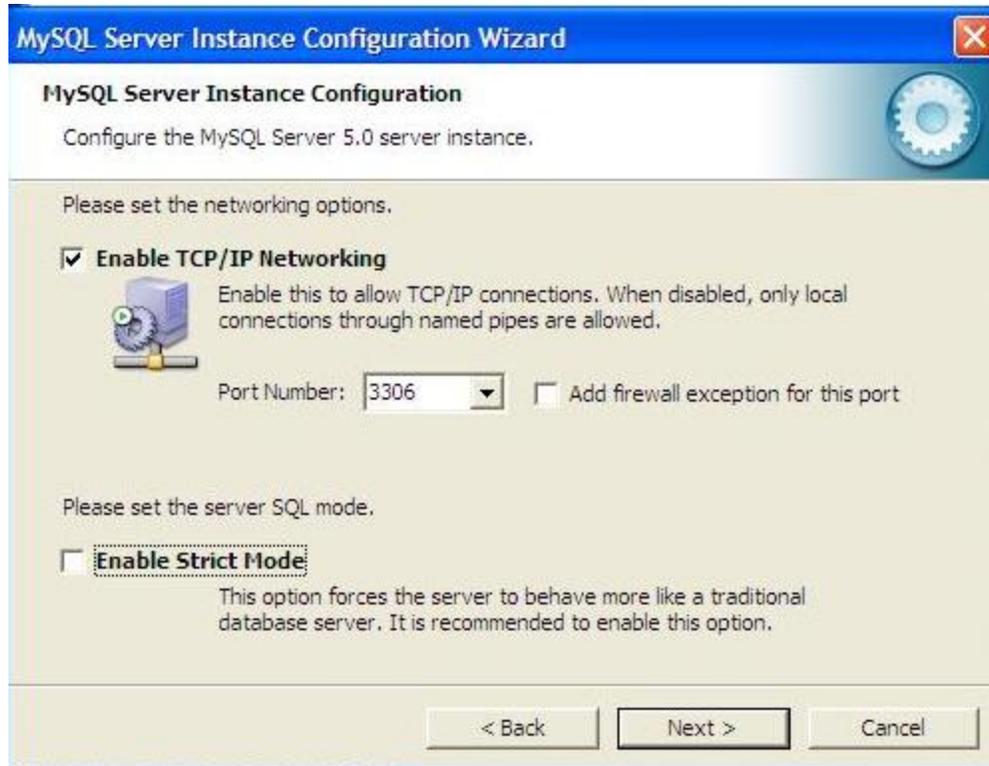
On the step number five just press 'Next' using defaults. (pic. 5)



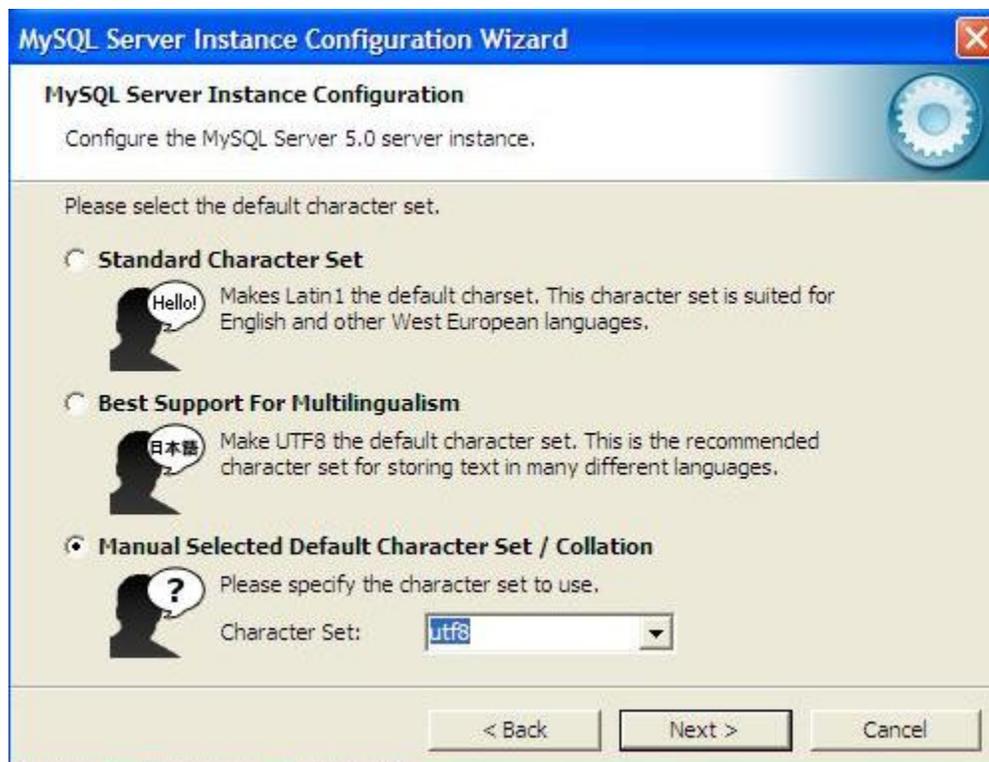
Again, on window 6 select the default - 'Decision Support (DSS)/OLAP' and press 'Next' (pic.6)



Make sure you switch OFF the 'Strict mode' and and press 'Next' (pic. 7)



On the character encoding page select: 'Manual Selected Default Character set/ Collation' and 'utf8', press 'Next' (pic.8)



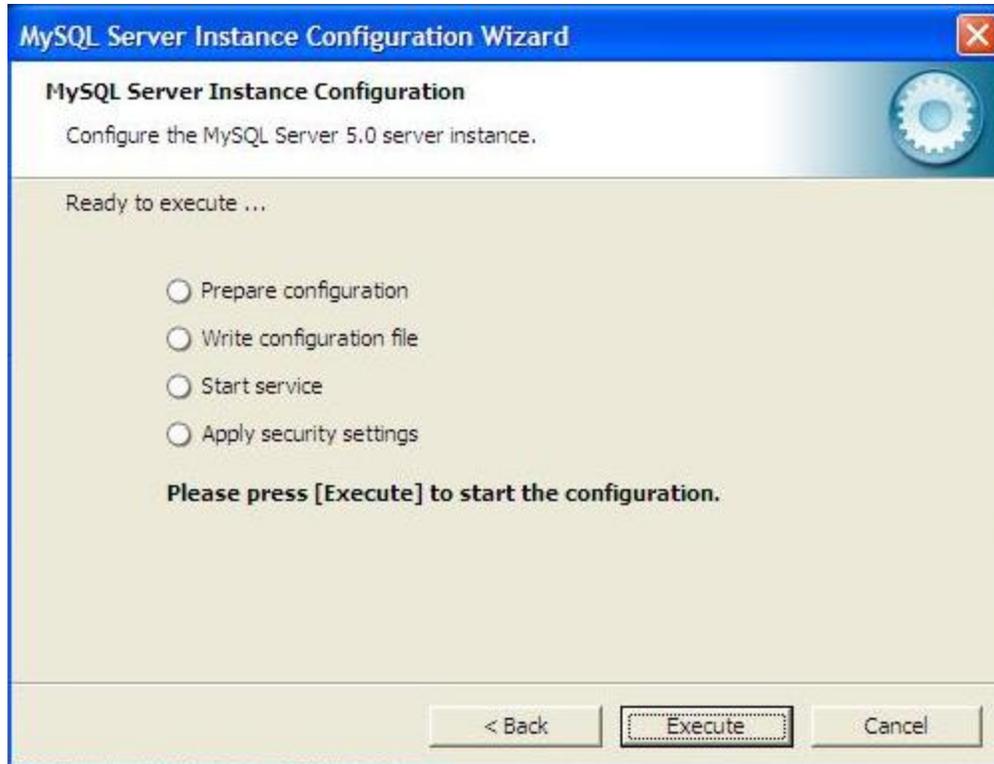
On next window select 'Include Bin Directory in Windows PATH' and press 'Next' (pic.9)



On this window just enter the database super user password and make sure you remember it. When ready press 'Next' (pic. 10)



This is the last screen. Press 'Execute' to save the configuration parameters. (pic. 11)



When the configuration is saved you can repeat all the step and change settings at any time by running:  
**START ⇒ Programs ⇒ MYSQL# MYSQL serwer machine# MySQL Server Instance Config Wizard**

Now we have to set Tigase database up. From the Start menu run the MySQL console and enter all commands below finishing them with <ENTER>:

1. Create the database:

```
mysql>create database tigasedb;
```

2. Add database user:

```
mysql> GRANT ALL ON tigasedb.* TO tigase_user@%' IDENTIFIED BY -'tigase_passwd'  
mysql> GRANT ALL ON tigasedb.* TO tigase_user@'localhost' IDENTIFIED BY -'tigase_passwd'  
mysql> GRANT ALL ON tigasedb.* TO tigase_user IDENTIFIED BY -'tigase_passwd';  
mysql> FLUSH PRIVILEGES;
```

3. Load Tigase database schema:

```
mysql> use tigasedb;  
mysql> source c:/Program Files/Tigase/database/mysql-schema.sql;
```

There is a small configuration bug in the installation of the version 3.x. If you look in the **run.bat** file in the Tigase directory you have to replace string **inital.propereties** with **init.properties**.

You can now restart your machine and all services including the MySQL database and the Tigase server should be running. Alternatively if you don't want to restart your computer you can start both services manually if you know how to do it.

When the system is up and running you can connect with any Jabber client (Psi for example) to your server to see if it is working.

Now, you can tweak the server configuration further. Use the `initPropertiesGuide` guide describing `init.properties` file for configuration details.

---

# Chapter 77. Configuration

Artur Hefczyc <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: <http://tigase.net> :Date: 2010-04-06 21:16

This section contains the Tigase server configuration manuals and guides.

*Please note, this guides are for the Tigase server 2.x line which is no longer supported. Please upgrade to the last stable version and contact us if you have any problems.*

---

# Chapter 78. Configuration Wizards

Artur Hefczyc <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2010-04-06 21:16

From the build #247 you can use configuration generators to easily and quickly create configuration file for even complex case.

**Tigase** configuration is not too easy to understand and maintain. Even with current command line tools you still have to know what the all options are for.

To make it easier for average administrators or people who run the server for the first time or even for those who want to quickly test **Tigase** server in different scenarios configuration generators have been created. For each generator you can have also a few extra options which allows you to create configuration which you don't need to change for some time.

A few definitions first to make it easier to read the rest:

1. **sm** - session manager component.
2. **c2s** - client connection manager component
3. **s2s** - server connection manager component
4. **ext2s** - external component connection manager
5. **ssender** - `////<<genericStanzaSender,////StanzaSender` component

The are 4 generators currently available:

1. `--gen-config-all` - creating configuration file with all available components. That is: sm, c2s, s2s, ext2s, ssender.
2. `--gen-config-default` - creating default configuration file. That is configuration which is most likely needed for basic installation. Components included in configuration are: sm, c2s, s2s.
3. `--gen-config-sm` - creating configuration for instance with session manager and external component only. This is useful for distributed installation where you want to have session manager installed on separate machine and components managing network connections on different machines (one or more). Components included in configuration are: sm and ext2s.
4. `--gen-config-cs` - creating configuration for instance with components managing network connections. This is useful for distributed installation where you want to have session manager installed on separate machine and components managing network connections on different machines (one or more). Components included in configuration are: c2s, s2s, ext2s.

For each of above generator you can use additional parameters specifying other configuration details like database type you want to connect to, virtual hosts you want to support, administrator accounts and details for external component connection:

- `--user-db db-type` - where 'db-type' can be one of possible values: **mysql**, **pgsql**, **xml**
- `--user-db-uri connection-uri` - where 'connection-uri' is a full resource uri for user repository data source. If you skip this parameter default value is used depending on database type you selected:

```
-- jdbc:mysql://localhost/tigase?user=root&password=mypass
```

```
-- jdbc:postgresql://localhost/tigase?user=tigase
-- user-repository.xml
```

- --auth-db db-type - where 'db-type' can be one of possible values: **mysql**, **pgsql**, **xml**, **drupal**, **libre-source** (If omitted 'user-db' settings are used.)
- --auth-db-uri connection-uri - where 'connection-uri' is a full resource uri for user repository data source. (If omitted 'user-db-uri' settings are used.)
- --ext-comp connection-string - possible values: connection string 'localdomain,remotedomain,port,passwd,plain/ssl,accept/connect,routing'

**Note:** It is also possible to generate configuration for many external components. To do so place --ext-comp\_1 'parameters' --ext-comp\_2 'parameters' and so on...

- --virt-hosts virtual-hosts-list - possible values: list of virtual domains to support 'domain1,domain2'. This option causes to use virtual hosts given here instead of default/automatically detected host names.
- --admins admin-accounts-list - possible values: list of admin accounts: 'user1@domain,user2@domain2'
- --test - this parameter informs that config is generated for test instance, which means that all loggings are turned off
- --debug tigase-package - you can turn on debugs log for selected tigase package. For example if you want to turn debug logs on for package: **tigase.server** then you have to put parameter: --debug server. If you have any problems with your server the best way to get help from me is to generate configuration with --debug server and run the server. Then from the logs/tigase-console.log log file I can get all information I need to give you a help.

**Note!** If configuration file already exists none of existing settings are overwritten. Configuration generator is activated only if config file does not exist at program startup or for config entries which are missing at startup time. So you can as well leave these settings in the file.

**Note!** tigase.conf property file is *NOT* read by the tigase server. These properties are read by the bash shell to create proper tigase server startup command. It will not work on MS Windows unless you run it in bash (using CygWin for example). On windows however you can use configuration wizards too by preparing proper server startup command manually. For example command for the first below presented config file would look like (all in single line):

```
java -Djdbc.drivers=org.postgresql.Driver
--Dfile.encoding=UTF-8 --Dsun.jnu.encoding=UTF-8
--server --Xms100M --Xmx100M
--cp -"libs/pg73jdbc3.jar;jars/tigase-server.jar;libs/tigase-xmltools.jar;libs/ti
tigase.server.XMPPServer
--c -"etc/tigase.xml"
---gen-config-def ---user-db pgsql
---user-db-uri -"jdbc:postgresql://localhost/tigase?user=tigase"
```

So for example to take advantage of these options you can create tigase.conf and start Tigase server with usual command to generate "tigase-config.xml" configuration file:

```
./bin/tigase.sh run tigase.conf
```

A few sample files are included below for your convenience:

- tigase-def-pgsql.conf - default installation with PostgreSQL database support:

```
ENC="-Dfile.encoding=UTF-8 --Dsun.jnu.encoding=UTF-8"  
DRV="-Djdbc.drivers=org.postgresql.Driver"  
JAVA_OPTIONS="${ENC} ${DRV} --server --Xms100M --Xmx100M -"
```

```
TIGASE_CONFIG="tigase-pgsql.xml"  
TIGASE_OPTIONS="--gen-config-def ---user-db postgresql ---user-db-uri jdbc:postgresql
```

```
ENC="-Dfile.encoding=UTF-8 --Dsun.jnu.encoding=UTF-8"  
DRV="-Djdbc.drivers=org.postgresql.Driver"  
JAVA_OPTIONS="${ENC} ${DRV} --server --Xms100M --Xmx100M -"
```

```
TIGASE_CONFIG="tigase-pgsql.xml"  
TIGASE_OPTIONS="--gen-config-def ---user-db postgresql -"
```

- `tigase-cs.conf` - installation of network connections management components (no DB is used by this instance):

```
ENC="-Dfile.encoding=UTF-8 --Dsun.jnu.encoding=UTF-8"  
JAVA_OPTIONS="${ENC} --server --Xms100M --Xmx100M -"
```

```
TIGASE_CONFIG="etc/tigase-cs.xml"  
TIGASE_OPTIONS="--gen-config-cs ---virt-hosts cs.tigase.org,tigase.org,sm.tigase  
---ext-comp cs.tigase.org,sm.tigase.org,5678,very-secret,plain,connect"
```

- `tigase-sm-mysql.conf` - installation of session manager instance and resource connection string is the same as default so we can skip `--user-db-uri` parameter:

```
ENC="-Dfile.encoding=UTF-8 --Dsun.jnu.encoding=UTF-8"  
DRV="-Djdbc.drivers=com.mysql.jdbc.Driver"
```

```
JAVA_OPTIONS="${ENC} ${DRV} --server --Xms100M --Xmx100M -"  
TIGASE_CONFIG="etc/tigase-sm-mysql.xml"  
TIGASE_OPTIONS="--gen-config-sm ---user-db mysql ---auth-db mysql ---virt-hosts t  
---ext-comp sm.tigase.org,cs.tigase.org,5678,very-secret,plain,accept"
```

---

# Chapter 79. Tigase Server version 2.x

Artur Hefczyc <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2010-04-06 21:16

Administration manuals and guides for the Tigase server 2.x line.

*Please note, this guides are for the Tigase server 2.x line which is no longer supported. Please upgrade to the last stable version and contact us if you have any problems.*

- [Installation](#)
- [MySQL Database Use](#)
- [PostgreSQL Database Use](#)
- [Configuration](#)
- [Command Line Admin Tools](#)

---

# Chapter 80. Installation

Artur Hefczyc <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2010-04-06 21:16

To get server up and running you have to download from our site the most recent version of the binary package.

Unpack it with the following command:

On Unix like system:

```
tar --xzvf tigase-server-x.x.x-bx.tar.gz
```

On MS Windows system use any application which can handle either zip files or tar.gz files and unpack server package to selected directory.

Sometimes after unpacking package on unix system startup script doesn't have execution permissions. To fix the problem you have to run following command:

```
chmod u+x ./bin/tigase.sh
```

As there are also other useful scripts you could just set executable bit for all of them at the same time:

```
chmod u+x ./bin/*
```

Now all you need is *Java 6 (1.6 beta2 at the moment)* compliant virtual machine. You can run server from command line with simple command:

```
./bin/tigase.sh run etc/tigase.conf
```

You can see now a few messages with warning about missing configuration file and missin user repository file. These 2 files will be automaticaly created. Config file will be created just during first execution of the server and user repository file will be created when the first user is added to the system.

You should be able now to connect to the server with Jabber/XMPP client of your choice.

First parameter is a command, second parameter is a config file for startup script. Possible commands are:

- **start** - Starts server in background redirecting all console messages to separate log file.
- **stop** - Stops the last started server Be carefull with this command if the server has been stopped in different way but the script still thinks the server is running. The script detects that the server has been started by looking for `tigase.pid` file. Then it reads server PID from the file and sends kill signal to the process. It may happen that the server is not running but there is another process with this PID. If you know that you server is not running but the script claims it is, it means that the script finds old `tigase.pid` file. In such case to make it possible to start server again you should either remove the file manually or run script with **zap** command.
- **run** - Starts the server as foreground process with all console messages printed onto console. To stop the server simply press Ctrl-C. It does not create `tigase.pid` file.
- **restart** - Restarts currently running server. It simply calls the script first with stop command and then with start command.
- **check** - This commands simply prints all the settings which would be used to start or stop the server. It is strongly recommended to use this command before the first server run.

Config file for startup script simply sets number of environment variables with location of required components. Possible variables to set in this file are:

- **JAVA\_HOME** - location of Java installation home directory. **Must be set.**
- **TIGASE\_HOME** - location of Tigase installation home directory. *By default script try to find this location by searching directories from the location where the script has been run.*
- **TIGASE\_CONSOLE\_LOG** - file to which all console messages will be redirected if server is run in background. By default it will be: TIGASE\_HOME/logs/tigase-console.log. **If this file/directory is not writable by Tigase process all console messages will be redirected to /dev/null**
- **TIGASE\_PID** location of the file with server PID number. By default it will be TIGASE\_HOME/logs/tigase.pid.
- **TIGASE\_CONFIG** - location of the Tigase server config file. This is main config XML file. Not to be confused with startup script parameters file. *If not set script trys to find it in following locations in given order: /etc/conf.d/tigase-server.xml, /etc/tigase-server.xml, /etc/tigase/tigase-server.xml or finally in TIGASE\_HOME/etc/tigase-server.xml*
- **JAVA\_OPTIONS** - options for JVM like size of RAM allocated for the JVM, properties and so on.

You can now proceed to configuration section. Although in simple case installations most of default options should be correct sometimes you need to change server domain name if automatic detection didn't work.

---

# Chapter 81. MySQL Database Use

Artur Hefczyc <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2010-04-06 21:16

This guide describes how to configure Tigase server to use MySQL [<http://www.mysql.com/>] database as user repository.

If you used XML based user repository before you can copy all user data to MySQL database using repository management tool. All steps are described below.

## MySQL Database Preparation

To load db schema to your MySQL instance first create database:

```
mysqladmin --p create tigase
```

And then you can load database schema:

```
mysql --u dbuser --p tigase < mysql-schema.sql
```

## Server Configuration

Now you have to change configuration to load jdbc module instead of XML based repository. Using configuration management script, first change class name handling repository.

To see current settings run command:

```
$ ./scripts/config.sh --c tigase-config.xml --print --key session_1/user-repo-cla
```

As a result you should see something like:

```
session_1/user-repo-class = tigase.db.xml.XMLRepository
```

You can see that current setting points to XML repository implementation. To use jdbc module for connection to MySQL database you have to set `tigase.db.jdbc.JDBCRepository` class (enter text below in one line):

```
$ ./scripts/config.sh --c tigase-config.xml --print
  --key session_1/user-repo-class --value tigase.db.jdbc.JDBCRepository --set
```

As a result you will see new value set for the parameter:

```
session_1/user-repo-class = tigase.db.jdbc.JDBCRepository
```

You have also to set the same value as authorization repository unless you want to use different authorization data source:

```
$ ./scripts/config.sh --c tigase-config.xml --print
  --key session_1/auth-repo-class --value tigase.db.jdbc.JDBCRepository --set
```

And again as a result we can see:

```
session_1/auth-repo-class = tigase.db.jdbc.JDBCRepository
```

Next step is to set database connection string. Assuming you have database: **tigase** on **localhost** with database user: **dbuser** and password **dbpass** your connection string will look like this:

```
jdbc:mysql://localhost/tigase?user=dbuser&password=dbpass
```

To set this in your configuration file you have to call again configuration management script 2 times. First for user data repository and second for authorization data repository:

```
$ ./scripts/config.sh --c tigase-config.xml --print --key session_1/user-repo-url
    --value -"jdbc:mysql://localhost/tigase?user=dbuser&password=dbpass" --set
$ ./scripts/config.sh --c tigase-config.xml --print --key session_1/auth-repo-url
    --value -"jdbc:mysql://localhost/tigase?user=dbuser&password=dbpass" --set
```

Note quotes around connection string. They are needed to make sure shell won't interpret special characters.

Now your configuration is ready to load jdbc module and connect to your database.

One more thing you need to do is to tell JVM which jdbc driver to use to connect to database. Depending on your MySQL and jdbc installation it might be: `com.mysql.jdbc.Driver`. To set is as database driver you have to set is a `jdbc.drivers` property value. Usually you do this by adding `-D` parameter to Java call:

```
$ java -Djdbc.drivers=com.mysql.jdbc.Driver tigase.server.XMPPServer
```

If you use `tigase.sh` script to run server you have to add `-Djdbc.drivers=com.mysql.jdbc.Driver` to startup script `<initProperties,property file>>` to `JAVA_OPTIONS` values.

## User Data Import

If you previously used XML based user repository you can import all data into MySQL database using repository management tool. This is quite long command so let me list all required parameters first with brief explanation:

1. **-cp** copy content of the source repository to destination repository.
2. **-sc `tigase.db.xml.XMLRepository`** source repository class.
3. **-su `user-repository.xml`** source repository connection string - assuming your user repository is in `user-repository.xml` file.
4. **-dc `tigase.db.jdbc.JDBCRepository`** destination repository class.
5. **-du `"jdbc:mysql://localhost/tigase?user=dbuser&password=dbpass"`** destination repository connection string.

And now whole command. Enter all in one line:

```
$ ./scripts/repo.sh --cp --sc tigase.db.xml.XMLRepository --su user-repository.xml
    --dc tigase.db.jdbc.JDBCRepository
    --du -"jdbc:mysql://localhost/tigase?user=dbuser&password=dbpass"
```

For more information how to use command line administration tools refer to [////<<commandLine-Tools2,////command line tools guide](#).

---

# Chapter 82. PostgreSQL Database Use

Artur Hefczyc <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: <http://tigase.net> :Date: 2010-04-06 21:16

This guide describes how to configure Tigase server to use PostgreSQL [<http://www.postgresql.org/>] database as user repository.

If you used XML based user repository before you can copy all user data to PostgreSQL database using repository management tool. All steps are described below.

## PostgreSQL Database Preparation

Create new database user account which will be used to connect to your database:

```
# createuser
Enter name of user to add: tigase
Shall the new user be allowed to create databases? (y/n) y
Shall the new user be allowed to create more new users? (y/n) y
```

Now using new database user account create database for your service:

```
# createdb --U tigase tigasedb
CREATE DATABASE
```

Now you can load database schema [<http://server.tigase.org/browser/trunk/database/postgresql-schema.sql>]:

```
# psql --U tigase --d tigasedb --f postgresql-schema.sql
```

Now database is ready for Tigase server to use it.

## Server Configuration

Server configuration is identical as for MySQL database setup. The same jdbc module is used to connect to PostgreSQL database as for MySQL. The only difference is connection string which usually looks like:

```
jdbc:postgresql://localhost/tigasedb?user=tigase
```

So for more detailed guide how to change configuration refer to [////<<mysql2database,////MySQL database use guide](#) or if you look for more automatic config file generation refer to [////<<3xconfiguration,////configuration wizards page](#).

---

# Chapter 83. Configuration

Artur Hefczyc <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2010-04-06 21:16

This section contains documentation about the Tigase server configuration.

*Please note, this guides are for the Tigase server 2.x line which is no longer supported. Please upgrade to the last stable version and contact us if you have any problems.*

- [Short Configuration Guide](#)

---

# Chapter 84. Short Configuration Guide

Artur Hefczyc <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2010-04-06 21:16

Options you most likely have to change at deployment time are:

1. **Admin accounts** - account names where all admin messages are sent.
2. **Hostnames** - real and virtual hostnames your server has to serve for.
3. **Logs** - setting related to log file location and how much information should be logged there.

Please let me know if you think more options should be described here.

## Changing configuration using command line tool.

Detailed description of configuration management tool in on project site on the: <http://www.tigase.net/documentation/ConfigurationManagement2/> Configuration management tool page.

## Admin Accounts

This is the most likely thing to change after you install server and generate default configuration. Actually it is also the easiest option to customize.

And yes, you can just remove entry with admin account: "admin@localhost" unless you really want to keep it. Be aware though all system messages will be sent to **ALL** admin accounts.

Well, if the account does not exist the message is discarded and a warning is printed in log file. Again, read it again, the previous sentence...

It means that the admin account has to be also created in normal way on the Jabber server. Just register it using your Jabber client. The admin accounts setting works just as a forward instruction. So as a result all system and admin messages are forwarded to all admin accounts if they exist.

Obviously you can have as many admin accounts as you like:

```
session_1/admins = frank@jabber.example.com, lucy@jabber.example.com,  
mark@jabber.example.com, brenda@jabber.example.com, luck@jabber.example.com
```

Now, how to do it. First things is to find current settings. To see what are current admin accounts execute the command:

```
./bin/config.sh --c etc/tigase-server.xml --print --key session_1/admins
```

As a result you can see something like this:

```
session_1/admins = admin@yourhost, admin@localhost
```

As you could read in configuration management too guide '-add' command adds new entries to the list, '-set' command removes old values and sets new one.

As we want to create new admins list for the installation first command would be '-set' command. Let's say we want set jack@yourhost as your first admin account:

```
./bin/config.sh --c etc/tigase-server.xml --print --key session_1/admins --set --v
```

As a result program prints new setting:

```
session_1/admins = jack@yourhost
```

All old accounts have been replaced with new account. Now if you wish to add next admin account john@yourhost to the list execute slightly modified command (Note - '-add' instead of '-set'):

```
./bin/config.sh --c etc/tigase-server.xml --print --key session_1/admins --add --v
```

And result:

```
session_1/admins = jack@yourhost, john@yourhost
```

And... this is it. You can add more admin accounts the same way.

## Hostnames

This one might be a little bit more tricky than previous as **hostnames** setting has to be changed in a few places. Don't ask why now as this is the "*Short configuration guide*", you remember. Here we focus on **how** not on **why**.

The best way to find all places where hostnames should be changed execute following command:

```
./bin/config.sh --c etc/tigase-server.xml --print -| grep hostnames
```

As a result you can see something like this:

```
tigase-xmpp-server/hostnames = yourhost, localhost
client_1/hostnames = yourhost, localhost
server_1/hostnames = yourhost, localhost
session_1/hostnames = yourhost, localhost
```

It may also look:

```
tigase-xmpp-server/hostnames = localhost
client_1/hostnames = localhost
server_1/hostnames = localhost
session_1/hostnames = localhost
```

Depending how successful was mechanism for automatic hostname detection. It of course does not depend on your luck. It depends on network configuration on your server.

The first form is more useful as it includes also hostname recognized in network environment. If it is correct then you can just leave it as it is. If it is incorrect you have to change it. Please remember, if you want your server to be able to communicate with other *Jabber/XMPP* servers the hostname you put there must resolve in DNS to your Jabber server machine IP address. In other words. If you try to connect from the Internet to machine with this hostname the connection should reach your Jabber server.

And remember your Jabber server users' JIDs (*Jabber IDs*) can include only those hostnames which are included in the configuration. So for our case you can use only *JIDs*: "user2@your.hostname.com", "user1@your.hostname.com" and so on.

If your server have more Internet addresses (virtual domains) assigned to it your Jabber server can use them all. So your configuration may look like:

```
tigase-xmpp-server/hostnames = yourhost, localhost, your.hostname.com, clien1.host
```

```
client_1/hostnames = yourhost, localhost, your.hostname.com, clien1.hostname.com
server_1/hostnames = yourhost, localhost, your.hostname.com, clien1.hostname.com
session_1/hostnames = yourhost, localhost, your.hostname.com, clien1.hostname.com
```

In such case users' *JIDs* on your Jabber server may include any of defined above domains like: "user1@your.hostname.com [mailto:user1@your.hostname.com]", "user1@clien1.hostname.com [mailto:user1@clien1.hostname.com]", "user1@yourhost". Each of these 3 sample *JIDs* refer to different user account.

Your server will accept connections only for domains defined in configuration file.

In majority cases it does not matter whether you leave "**localhost**" or remove it. It is sometimes better to leave it though. So if you are not sure if you can remove it in your environment just leave it as is.

Now it is a time to change hostnames. Assuming default configuration is not what you need than first you have to set all these settings to accept connections from localhost:

```
./bin/config.sh --c etc/tigase-server.xml --print --key tigase-xmpp-server/hostnames
./bin/config.sh --c etc/tigase-server.xml --print --key client_1/hostnames --value
./bin/config.sh --c etc/tigase-server.xml --print --key server_1/hostnames --value
./bin/config.sh --c etc/tigase-server.xml --print --key session_1/hostnames --value
```

Now we can add you "real" host name, let's say it is yourhost:

```
./bin/config.sh --c etc/tigase-server.xml --print --key tigase-xmpp-server/hostnames
./bin/config.sh --c etc/tigase-server.xml --print --key client_1/hostnames --value
./bin/config.sh --c etc/tigase-server.xml --print --key server_1/hostnames --value
./bin/config.sh --c etc/tigase-server.xml --print --key session_1/hostnames --value
```

The same way you can add more virtual hosts to you configuration.

---

# Chapter 85. Command Line Admin Tools

Artur Hefczyc <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: <http://tigase.net> :Date: 2010-04-06 21:16

Two command line tools have been created to make it easier to manage server configuration and user repository.

Configuration tool allows to look at configuration settings and modify parameters. It takes care about proper parameters types and encoding.

Repository management tool allows to print repository content for all or for selected users. Modify repository data, add, delete users and copy data from one repository to another.

This guide describe how to efficiently use command line tools which are available for user repository and configuration management.

These 2 command line tools for managing configuration and repository are:

1. `config.sh` [<http://server.tigase.org/browser/trunk/scripts/config.sh>] for configuration management.
2. `repo.sh` [<http://server.tigase.org/browser/trunk/scripts/repo.sh>] for repository management.

Both scripts call class from Tigase package. If you run any of those script with `-h` parameter you will get help screen describing all available parameters.

I will not concentrate on that help information which is easily accessible anyway. This guide will focus on particular use cases. So it will answer to questions: "How to do it with the tool?".

---

# Chapter 86. Configuration Management Tool

Artur Hefczyk <artur.hefczyc@tigase.net [mailto:artur.hefczyc@tigase.net]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: http://tigase.net :Date: 2010-04-06 21:16

Configuration tool allows to look at configuration settings and modify parameters. It takes care about proper parameters types and encoding.

First answer to the question: "Why to use configuration tool instead of directly, manually edit config file?"

There are a couple of reason why you should NOT manually edit configuration file and use the tool to modify settings:

1. Configuration is kept in XML file which can be easily broken if not edited carefully. The tool takes care of creating valid XML configuration file for you. So you can focus on your task - setting proper parameters for your server.
2. Configuration values are kept **UUEncoded** in config file. If you edit file manually you have to take care fo proper encoding special characters. The tool presents parameters to you in decoded form which is easy to read and accept all settings also in decoded form which is easier for you to write. Then when writing your parameters to configuration file settings are automatically encoded to correct form.
3. Data in configuration file have **TYPES**. That is some parameters are expected to be **Strings** other are expected to be **Integers**, **Booleans** or arrays keeping data in any of that type. If data type is set incorrectly then server may have problems with reading configuration data. Configuration tool takes care of propret data type in configuration file.

Configuration management tool is a Java class - `tigase.conf.Configurator` [<http://server.tigase.org/browser/trunk/src/tigase/conf/Configurator.java>]. To make it easier to use this class there is also shell script available - `config.sh` [<http://server.tigase.org/browser/trunk/scripts/config.sh>].

First thing you can do is running the script with `-h` parameter:

```
./scripts/config.sh --h
```

In response you get description of all available parameters:

Parameters:

```
--h          this help message
--c file     configuration file
--key key    node/key for the value to set
--value value value to set in configuration file
--set       set given value for given key
--add       add given value to the values list for given key
--print     print content of all configuration settings or of given node/key
--f        force creation of the new property -- dangerous option...
```

Samples:

```
Setting admin account -- overwriting any previous value(s)
```

```
$ ./scripts/config.sh --c tigase-config.xml --print --set --key session_1/admins
```

```
Adding next admin account leaving old value(s)
```

```
$ ./scripts/config.sh --c tigase-config.xml --print --add --key session_1/admins
```

Note: adding `--print` option is useful always, even with `--set` or `--add` option as it prints set value afterwards.

Let's assume configuration for your server is located in **tigase-config.xml** file. So the first thing you need to set when calling the tool is location of the configuration file.

---

# Chapter 87. Best Practices for Connecting From Web Browser to Tigase XMPP Server

Andrzej Wojcik <andrzejw@tigase.org [mailto:andrzejw@tigase.org]> v2.0, June 2014: Reformatted for AsciiDoc. :toc: :numbered: :website: <http://tigase.net> :Date: 2013-12-15 19:27

Currently we have 2 ways to connect to Tigase XMPP Server from web browsers:

1. BOSH (Bidirectional-streams Over Synchronous HTTP)
2. WebSocket (XMPP over WebSocket)

You will find more informations about these ways for connecting to Tigase XMPP Server with some useful tips below.

## BOSH

BOSH protocol specified in XEP-0124 [<http://xmpp.org/extensions/xep-0124.html>] is one of first protocols defined to allow to establish XMPP connection to XMPP servers from web browsers. Due to that this protocol is widely supported and used by many deployments. It is also easy to use in single server mode. It's enabled by default in Tigase XMPP Server and available at port 5280.

In clustered mode we can deploy it with load balancer deployed with guarantees that each BOSH connection from web browser will be forwarded to same Tigase XMPP Server instance. So in clustered mode if we have two XMPP server t1 and t2 which are hosting domain example.com we would need to have load balancer which will respond for HTTP request to domain example.com and forward all requests from same IP address to same node of a cluster (i.e. all request from 192.168.122.32 should be forwarded always to node t1.

## Tip #1 - BOSH in Cluster Mode Without Load Balancer

There is also a way to use BOSH without load balancer. In this case XMPP client needs to have more logic and knowledge about all available cluster nodes (with names of nodes which will identify particular cluster nodes from internet). Using this knowledge XMPP client should select one random node from list of available nodes and always establish BOSH connections to this particular node. In case if BOSH connection fails due to network connection issues XMPP client should randomly pick other node from list of rest of available nodes.

*Example:*

We have servers t1.example.com and t2.example.com which are nodes of a cluster hosting domain example.com. Web client retrieves list of cluster nodes from web server and then when it needs to connect to XMPP server it picks random host from list of retrieved cluster nodes (i.e. t2.example.com) and tries to connect using BOSH protocol to host t2.example.com but it should send example.com as name of server to which it tries to connect (example.com should be value of to attribute of XMPP stream.

## WebSocket

WebSocket protocol is newly standardized protocol which is supported now by many of current versions of browsers. Currently there is a draft of protocol draft-ietf-xmpp-websocket-00 [<https://datatracker.ietf.org/>]

doc/draft-ietf-xmpp-websocket/] which describes usage of WebSocket to connect to XMPP server. Tigase XMPP Server implementation of WebSocket protocol to connect to XMPP server is very close to this draft of this specification. By default Tigase XMPP Server has XMPP-over-WebSocket protocol enabled without encryption on port 5290. To use this protocol you need to use library which supports XMPP-over-WebSocket protocol.

## Tip #1 - Encrypted WebSocket Connection

It is possible to enable encrypted WebSocket connection in Tigase XMPP Server. To do this you need to add following lines to etc/init.properties config file:

```
ws2s/connections/ports[i]=5290,5291
```

```
ws2s/connections/5291/socket=ssl
```

```
ws2s/connections/5291/type=accept
```

In this example we enabled WebSocket endpoint on port 5290 and encrypted WebSocket endpoint on port 5291. Connections on port 5291 are SSL connections which are encapsulating not encrypted WebSocket connections. As this is TLS/SSL connection (no STARTTLS) it uses default certificate installed in Tigase XMPP Server instance. This certificate is located in certs/default.pem.

## Tip #2 - Encrypted WebSocket Connection - Dealing With Multiple VHosts

As mentioned in Tip #1 WebSocket endpoint is plain TLS/SSL port, so it always serves default certificate for Tigase XMPP Server instance. It is ok, if we are hosting single domain and if default certificate matches our domain. But If we host multiple domain we cannot use wss://example1.com:5291/ connection URI, if our default certificate is for domain example2.com. In this situation it is recommended use default certificate for domain under which server is accessible from internet. This domain should identify this server, so this domain would not point i.e. on two nodes of a cluster. After we deploy separate certificate for each of cluster nodes, we should follow same tip as Tip #1 for BOSH. Our web-based XMPP client should have knowledge about each node of a cluster and when it needs to connect it should randomly select one node from list of available cluster nodes and try to connect to it using connection URL that would contain name of server under which it can be identified from internet.

*Example:*

We have servers t1.example1.com and t2.example1.com which are nodes of a cluster hosting domain example2.com. Each of our nodes contains default SSL certificate which domain matches name of cluster node. Web client retrieves list of cluster nodes from web server and then when it needs to connect to XMPP server it picks random host from list of retrieved cluster nodes (i.e. t2.example1.com) and tries to connect using WebSocket encrypted protocol to host t2.example1.com using following connections URL wss://t2.example1.com:5291/. During connection client should still send example2.com as name of server to which it tries to connect (example2.com should be value of to attribute of XMPP stream. This will allow browser to validate certificate as it will be for the same domain to which browser connects, and it will allow XMPP client to connect to domain example2.com, which is one of hosted vhosts.