

## **Tigase Team**

---

Tigase Team

---

---

# Table of Contents

I. Tigase Administration Guide .....	1
1. Tigase XMPP Server 8.0.0 announcement .....	12
Major Changes .....	12
Kernel and beans configuration .....	12
New Configuration File Format .....	12
Cluster Node Shutdown Changes .....	12
Significant cleanup of code and repositories .....	12
BouncyCastle being used for StartTLS .....	13
default-virtual-host property changes .....	13
All artifacts are signed .....	13
Scaled Down Installation Methods .....	13
Emojis now supported on Tigase XMPP Servers .....	13
XEP-0215 External Service Discovery now supported .....	13
XEP-0313 Message Archive Management now supported .....	13
XEP-0363 HTTP File Upload now supported .....	14
Startup now uses bootstrapping .....	14
CAPTCHA system now available for in-band registration .....	14
Schema changes .....	14
Shrinkable Statistics History .....	14
Statistics now available for all modules .....	14
Spam Protection .....	14
Changes in password storage .....	15
Dynamic TLS Buffer .....	15
XEP-305 Quickstart now supported .....	15
Database Timestamps .....	15
Config-type properties have changed .....	16
Database Watchdog implemented .....	16
Packet statistics expanded .....	16
XEP-0016 Behavior changes .....	16
Access Control List has new ACL modifiers .....	16
Option to ignore schema-version check added .....	16
Protection against brute-force attacks .....	17
New Minor Features & Behavior Changes .....	17
Fixes .....	20
Component Changes .....	23
AMP .....	23
PubSub .....	23
http-api .....	24
message-archive .....	24
MUC .....	25
socks5 Proxy .....	25
stats .....	26
STUN Server .....	26
WebSocket .....	26
2. Tigase User Guide .....	27
Jabber/XMPP introduction .....	27
Jabber/XMPP is Instant Messaging Technology .....	27
How to Use Tigase Service .....	27
This Article Describes How to use <b>tigase.im</b> Service for Instant Communica- tions .....	27
Configuration instructions for Psi .....	29

---

Psi - Initial configuration .....	29
Short Instructions How to Add Your First Contact .....	35
3. About Tigase XMPP Server .....	46
Robust and reliable .....	46
Security .....	46
Flexibility .....	47
Extensibility .....	47
Ease of Use .....	47
4. Licensing and Open Source .....	48
5. Tigase Server Binary Updates .....	49
6. Quick Start Guide .....	50
Minimum Requirements .....	50
Contents .....	50
Installation Using Web Installer .....	50
Download and Extract .....	50
Start the Server .....	51
Verify Tigase is Running .....	51
Connect to the Web Installer .....	51
Step Through the Installation Process .....	51
Restart the Server .....	72
Windows Instructions for using Web Installer .....	72
Manual Installation in Console Mode .....	72
Get the Binary Package .....	73
Unpack the Package .....	73
Prepare Configuration .....	73
Install Database .....	75
Start the Server .....	77
Check if it is Working .....	77
Windows Installation .....	78
Step 1: Initial Setup .....	78
Step 2: Starting Server .....	78
MySQL Database Installation .....	79
Tigase Server Network Instructions .....	85
A Records .....	85
SRV Records .....	86
Hosting VIA Tigase.me .....	86
Checking setup .....	87
Ports description .....	87
Tigase Script Selection .....	88
Configuration: For Linux Distributions using systemd .....	89
Configuration: For All Linux Distributions .....	89
Running Tigase as a system service .....	92
Shutting Down Tigase .....	93
Shutdown statistics .....	93
Shutdown StackTrace Dump .....	93
Shutting Down Cluster Nodes .....	94
Upgrading to v8.0.0 from v7.1.0 .....	94
Backup your data .....	94
Setup Tigase XMPP Server 8.0.0 .....	94
Upgrade configuration file .....	94
Connect new database .....	95
Upgrade Database schema .....	95
Help? .....	96
Upgrade/Restore with a script [experimental!] .....	96

---

---

7. Configuration .....	97
DSL file format .....	97
Why new format? .....	97
What is DSL? .....	98
Why DSL? .....	98
Example configuration file in DSL .....	103
Default configuration .....	104
Startup File for tigase.sh - tigase.conf .....	104
Linux Settings for High Load Systems .....	105
fs.file-max .....	105
net.ipv4.ip_local_port_range .....	106
TCP_keepalive .....	106
/etc/sysctl.conf .....	106
nofile .....	107
su and init script .....	108
JVM settings and recommendations .....	108
Heap Sizing .....	108
GC settings .....	109
What to use with Machine x, y, z? .....	111
Additional resources .....	112
Session Manager .....	112
Mobile Optimizations .....	113
threads-pool .....	114
Thread Pool factor .....	114
Strategy .....	114
Virtual Hosts in Tigase Server .....	115
Default VHost configuration .....	115
Specification for ad-hoc Commands Used to Manage Virtual Domains .....	116
Virtual Components for the Cluster Mode .....	119
Settings for Custom Logging in Tigase .....	119
Tigase Advanced Options .....	121
Using CAPTCHA for in-band registration .....	121
Enabling Empty Nicknames .....	121
Enable Silent Ignore on Packets Delivered to Unavailable Resources .....	121
Mechanism to count errors within Tigase .....	122
8. Security .....	124
XEP-0191: Blocking Command .....	124
Account Registration Limits .....	125
Brute-force attack prevention .....	126
Configuration .....	126
Server Certificates .....	126
Creating and Loading the Server Certificate in pem Files .....	127
Installing LetsEncrypt Certificates in Your Linux System .....	130
Custom Authentication Connectors .....	131
Tigase Auth Connector ( <b>DEPRECATED</b> ) .....	133
Tigase Custom Auth Connector .....	134
Drupal Authentication .....	138
LDAP Authentication Connector .....	139
Configuration of SASL EXTERNAL .....	139
Packet Filtering .....	140
Domain Based Packet Filtering .....	140
Access Control Lists in Tigase .....	142
9. Database Management .....	145
Recommended database versions .....	145

---

Database Watchdog .....	145
Using modified database schema .....	146
Database Preparation .....	146
Schema Utility .....	147
Prepare the MySQL Database for the Tigase Server .....	150
Prepare the Derby Database for the Tigase Server .....	155
Prepare the MS SQL Server Database for the Tigase Server .....	156
Prepare the PostgreSQL Database for the Tigase Server .....	159
Preparing Tigase for MongoDB .....	161
Hashed User Passwords in Database .....	165
Shortcut .....	165
Full Route .....	166
Tigase Server and Multiple Databases .....	168
Importing User Data .....	171
Importing Existing Data .....	173
Connecting the Tigase Server to MySQL Database .....	173
Integrating Tigase Server with Drupal .....	173
PostgreSQL Database Use .....	174
Schema Updates .....	175
Changes to Schema in v8.0.0 .....	175
Tigase Server Schema v7.2 Updates .....	179
10. Components .....	183
Advanced Message Processing - AMP XEP-0079 .....	183
First of all: plugins .....	183
Secondly: component .....	184
Optional parameters .....	184
Server Monitoring .....	185
Setting Up Remote Monitoring in the Server .....	185
Retrieving statistics from the server .....	187
Eventbus .....	194
Configuration of statistics loggers .....	200
Server to Server Protocol Settings .....	201
Number of Concurrent Connections .....	201
Connection Throughput .....	202
Maximum Packet Waiting Time and Connection Inactivity Time .....	202
Custom Plugin: Selecting s2s Connection .....	202
skip-tls-hostnames .....	203
ejabberd-bug-workaround .....	203
Tigase Load Balancing .....	203
Available Implementations .....	203
Configuration Options .....	204
Auxiliary setup options .....	206
External Component Configuration .....	207
External Component Configuration .....	207
Tigase as an External Component .....	209
Load Balancing External Components in Cluster Mode .....	215
Load Balancing External Component .....	216
External Component and Cluster .....	217
Client to Server Communication .....	219
Configuration .....	219
Connections .....	219
Resumption timeout .....	220
Packet Redelivery .....	220
Tigase External Service Discovery .....	221

---

Setup & Configuration .....	221
11. Using Tigase .....	222
Offline Messages .....	222
Offline Message Limits .....	223
Storing offline messages without body content .....	224
Disabling Offline Messages .....	226
Last Activity .....	226
What updates last activity .....	226
Persist everything to repository .....	226
Tigase Log Guide .....	227
install.log .....	227
derby.log .....	227
etc/config-dump.properties .....	227
logs/tigase.log.# .....	227
logs/statistics.log.# .....	228
logs/tigase.pid .....	228
logs/tigase-console.log .....	228
Log File Location .....	229
Debugging Tigase .....	229
Configuration .....	229
Basic System Checks .....	230
Add and Manage Domains (VHosts) .....	230
Using Admin UI .....	231
Using ad-hoc commands .....	233
SSL Certificate Management .....	238
Presence Forwarding .....	238
Watchdog .....	240
Setup .....	240
Watchdog Configuration .....	241
Logic .....	241
Testing .....	243
Tips and Tricks .....	243
Tigase Tip: Checking the Runtime Environment .....	244
Licensing .....	247
Registering for a License .....	247
What happens if I do not use a license file or it is expired? .....	248
Demo mode .....	248
Unauthorized use .....	252
Manual mode .....	252
Tigase Clustering .....	252
Configuration .....	252
Old configuration method .....	254
Checking Cluster Connections .....	254
Anonymous Users & Authentication .....	256
Anonymous Authentication .....	256
Anonymous User Features .....	256
Scripting support in Tigase .....	257
Scripting Introduction - Hello World! .....	257
Tigase Scripting Version 4.4.x Update for Administrators .....	265
Tigase and Python .....	268
12. Appendix I - Statistics description .....	271
Data source statistics .....	271
User repository statistics of {repo} .....	271
Auth repository statistics of {repo} .....	272

---

---

Statistics common to custom {compname} component repositories .....	273
Statistics common to components .....	274
Component statistics .....	281
AMP .....	281
bosh .....	281
c2s .....	281
cl-comp .....	281
eventbus .....	283
message-archive .....	283
message-router .....	284
monitor .....	285
muc .....	286
proxy .....	286
pubsub .....	287
repo-factory .....	298
rest .....	298
s2s .....	298
sess-man .....	300
vhost-man .....	312
ws2s .....	312
13. Appendix II - Properties Guide .....	313
General .....	313
admins .....	313
Certificate Container .....	313
Component .....	314
config-type .....	315
debug-packages .....	315
debug .....	316
monitoring .....	316
plugins .....	317
priority-queue-implementation .....	318
roster-implementation .....	319
s2s-secret .....	319
scripts-dir .....	320
ssl-container-class .....	320
stats .....	320
stream-error-counter .....	323
stringprep-processor .....	324
test .....	324
tls-jdk-nss-bug-workaround-active .....	325
trusted .....	325
Repository .....	325
authRepository .....	325
authRepository .....	327
Cluster .....	328
cl-comp .....	328
cluster-mode .....	329
cluster-nodes .....	329
User connectivity .....	330
bosh-close-connection .....	330
bosh-extra-headers-file .....	330
client-access-policy-file .....	330
client-port-delay-listening .....	331
cross-domain-policy-file .....	332

---



domain-filter-policy .....	332
see-other-host .....	333
watchdog_timeout .....	333
watchdog_delay .....	334
watchdog_ping_type .....	334
ws-allow-unmasked-frames .....	335
External .....	335
bind-ext-hostnames .....	335
default-virtual-host .....	336
ext .....	336
Performance .....	336
cm-ht-traffic-throttling .....	336
cm-traffic-throttling .....	337
elements-number-limit .....	337
hardened-mode .....	337
max-queue-size .....	338
net-buff-high-throughput .....	338
net-buff-standard .....	338
nonpriority-queue .....	339
VHost / domain .....	339
vhost-anonymous-enabled .....	339
vhost-disable-dns-check .....	339
vhost-max-users .....	340
vhost-message-forward-jid .....	340
vhost-presence-forward-jid .....	341
vhost-register-enabled .....	341
vhost-tls-required .....	342
Tigase Server Extras - mDNS support .....	342
Overview .....	342
Enabling mDNS .....	342
Using different domain name .....	343
Forcing single server for domain .....	343
14. HTTP API component .....	344
Available modules .....	344
Admin UI module .....	344
Index module .....	344
REST module .....	344
Server status module .....	344
Setup module .....	344
Web UI module .....	344
User Status Endpoint module .....	344
Common module configuration .....	345
Enabling/disabling module .....	345
Context path .....	345
List of virtual hosts .....	345
Complex example .....	345
Module specific configuration .....	346
Rest Module .....	346
DNS Web Service module .....	347
Enabling password reset mechanism .....	348
Admin UI Guide .....	348
A Note about REST .....	348
General overview of the UI .....	349
Configuration .....	349

---

Example Scripts .....	349
Notifications .....	350
Other .....	350
Scripts .....	359
Statistics .....	359
Users .....	360
Tigase Web Client .....	361
Chat .....	362
Discovery .....	363
Management .....	363
15. HTTP File Upload component .....	371
Enabling HTTP File Upload Component .....	371
Metadata repository .....	371
DummyFileUploadRepository .....	371
JDBCFileUploadRepository .....	371
Storage .....	371
DirectoryStore .....	372
Logic .....	372
URI template format .....	372
File upload expiration .....	373
Examples .....	373
Complex configuration example .....	373
Example configuration for clustering with HA .....	374
16. HTTP server .....	375
Dependencies .....	375
Configuration Properties .....	375
Additional properties of embedded HTTP server .....	376
Examples .....	376
HTTPS on port 8443 with SSL certificate for example.com .....	376
Changing port from 8080 to 8081 .....	376
Usage of Jetty HTTP server as HTTP server .....	376
17. Tigase Message Archiving Component .....	378
Tigase Message Archiving Component .....	378
Announcement .....	378
Major changes .....	378
New features .....	378
Additional features .....	379
Querying in all messages .....	379
Querying by part of message body .....	379
Querying by tags .....	380
Automatic archiving of MUC messages .....	381
Database .....	382
Preparation of database .....	382
Upgrade of database schema .....	382
Schema description .....	382
Configuration .....	384
Custom Database .....	384
XEP-0136 Support .....	384
Support for MAM .....	384
Setting default value of archiving level for message on a server .....	384
Setting required value of archiving level for messages on a server .....	385
Enabling support for tags .....	385
Configuration of automatic archiving of MUC messages .....	386
Purging Information from Message Archive .....	387

---

Using separate store for archived messages .....	388
Setting Pool Sizes .....	388
Message Tagging Support .....	389
Usage .....	390
XEP-0136 Field Values .....	390
Manual Activation .....	392
Limitations .....	392
18. Tigase PubSub Component .....	393
PubSub Component .....	393
Announcement .....	393
Configuration .....	394
Database .....	398
Features .....	403
AdHoc Commands .....	403
REST API .....	409
Limitations .....	438
Addressing .....	438
19. Tigase Socks5 Proxy .....	440
Overview .....	440
Installation .....	440
Database Preparation .....	440
Configuration .....	441
Enabling proxy .....	441
Using a separate database .....	443
Performance .....	444
20. Tigase Push Component .....	445
Tigase Push Component .....	445
Workflow .....	445
Configuration .....	445
Enabling component .....	445
Enabling push notifications for offline messages .....	446
Enabling push notifications for messages received when all resources are AWAY/XA/DND .....	446
Usage .....	447
Sending notifications .....	447
Registering device .....	447
Unregistering device .....	447
Providers .....	448
Tigase Push Component - FCM provider .....	448
Overview .....	448
Configuration .....	448
Tigase Push Component - APNs provider .....	449
Overview .....	449
Configuration .....	449
21. Tigase STUN Component .....	451
Tigase STUN Component .....	451
What is STUN? .....	451
Requirements .....	451
Configuration .....	451
Setting descriptions .....	451
Logback configuration .....	452
22. Tigase SPAM Filter .....	453
Overview .....	453
Configuration .....	453

---

Changing active SPAM filters .....	453
Sending error when packet is dropped .....	453
Enabling logging of dropped messages .....	454
Filters .....	454
Same long message body .....	454
Error message and missing <error/> child .....	455
Groupchat messages sent to bare JID .....	455
Known spammers .....	455
Presence subscription filter .....	457

---

## List of Tables

6.1. init.d chart .....	88
9.1. tig_users .....	163
9.2. tig_nodes .....	163
9.3. msg_history collection .....	164
19.1. tig_socks5_users .....	443

---

# Part I. Tigase Administration Guide

Tigase Team <team@tigase.com [mailto:team@tigase.com]> v8.0.0-RC1, 2019-01-30 :toc: :toclevels: 2 :numbered: :website: <http://tigase.net>

---

# Table of Contents

1. Tigase XMPP Server 8.0.0 announcement .....	12
Major Changes .....	12
Kernel and beans configuration .....	12
New Configuration File Format .....	12
Cluster Node Shutdown Changes .....	12
Significant cleanup of code and repositories .....	12
BouncyCastle being used for StartTLS .....	13
default-virtual-host property changes .....	13
All artifacts are signed .....	13
Scaled Down Installation Methods .....	13
Emojis now supported on Tigase XMPP Servers .....	13
XEP-0215 External Service Discovery now supported .....	13
XEP-0313 Message Archive Management now supported .....	13
XEP-0363 HTTP File Upload now supported .....	14
Startup now uses bootstrapping .....	14
CAPTCHA system now available for in-band registration .....	14
Schema changes .....	14
Shrinkable Statistics History .....	14
Statistics now available for all modules .....	14
Spam Protection .....	14
Changes in password storage .....	15
Dynamic TLS Buffer .....	15
XEP-305 Quickstart now supported .....	15
Database Timestamps .....	15
Config-type properties have changed .....	16
Database Watchdog implemented .....	16
Packet statistics expanded .....	16
XEP-0016 Behavior changes .....	16
Access Control List has new ACL modifiers .....	16
Option to ignore schema-version check added .....	16
Protection against brute-force attacks .....	17
New Minor Features & Behavior Changes .....	17
Fixes .....	20
Component Changes .....	23
AMP .....	23
PubSub .....	23
http-api .....	24
message-archive .....	24
MUC .....	25
socks5 Proxy .....	25
stats .....	26
STUN Server .....	26
WebSocket .....	26
2. Tigase User Guide .....	27
Jabber/XMPP introduction .....	27
Jabber/XMPP is Instant Messaging Technology .....	27
How to Use Tigase Service .....	27
This Article Describes How to use <b>tigase.im</b> Service for Instant Communications .....	27
Configuration instructions for Psi .....	29
Psi - Initial configuration .....	29
Short Instructions How to Add Your First Contact .....	35

3. About Tigase XMPP Server .....	46
Robust and reliable .....	46
Security .....	46
Flexibility .....	47
Extensibility .....	47
Ease of Use .....	47
4. Licensing and Open Source .....	48
5. Tigase Server Binary Updates .....	49
6. Quick Start Guide .....	50
Minimum Requirements .....	50
Contents .....	50
Installation Using Web Installer .....	50
Download and Extract .....	50
Start the Server .....	51
Verify Tigase is Running .....	51
Connect to the Web Installer .....	51
Step Through the Installation Process .....	51
Restart the Server .....	72
Windows Instructions for using Web Installer .....	72
Manual Installation in Console Mode .....	72
Get the Binary Package .....	73
Unpack the Package .....	73
Prepare Configuration .....	73
Install Database .....	75
Start the Server .....	77
Check if it is Working .....	77
Windows Installation .....	78
Step 1: Initial Setup .....	78
Step 2: Starting Server .....	78
MySQL Database Installation .....	79
Tigase Server Network Instructions .....	85
A Records .....	85
SRV Records .....	86
Hosting VIA Tigase.me .....	86
Checking setup .....	87
Ports description .....	87
Tigase Script Selection .....	88
Configuration: For Linux Distributions using systemd .....	89
Configuration: For All Linux Distributions .....	89
Running Tigase as a system service .....	92
Shutting Down Tigase .....	93
Shutdown statistics .....	93
Shutdown StackTrace Dump .....	93
Shutting Down Cluster Nodes .....	94
Upgrading to v8.0.0 from v7.1.0 .....	94
Backup your data .....	94
Setup Tigase XMPP Server 8.0.0 .....	94
Upgrade configuration file .....	94
Connect new database .....	95
Upgrade Database schema .....	95
Help? .....	96
Upgrade/Restore with a script [experimental!] .....	96
7. Configuration .....	97
DSL file format .....	97



Why new format? .....	97
What is DSL? .....	98
Why DSL? .....	98
Example configuration file in DSL .....	103
Default configuration .....	104
Startup File for tigase.sh - tigase.conf .....	104
Linux Settings for High Load Systems .....	105
fs.file-max .....	105
net.ipv4.ip_local_port_range .....	106
TCP_keepalive .....	106
/etc/sysctl.conf .....	106
nofile .....	107
su and init script .....	108
JVM settings and recommendations .....	108
Heap Sizing .....	108
GC settings .....	109
What to use with Machine x, y, z? .....	111
Additional resources .....	112
Session Manager .....	112
Mobile Optimizations .....	113
threads-pool .....	114
Thread Pool factor .....	114
Strategy .....	114
Virtual Hosts in Tigase Server .....	115
Default VHost configuration .....	115
Specification for ad-hoc Commands Used to Manage Virtual Domains .....	116
Virtual Components for the Cluster Mode .....	119
Settings for Custom Logging in Tigase .....	119
Tigase Advanced Options .....	121
Using CAPTCHA for in-band registration .....	121
Enabling Empty Nicknames .....	121
Enable Silent Ignore on Packets Delivered to Unavailable Resources .....	121
Mechanism to count errors within Tigase .....	122
8. Security .....	124
XEP-0191: Blocking Command .....	124
Account Registration Limits .....	125
Brute-force attack prevention .....	126
Configuration .....	126
Server Certificates .....	126
Creating and Loading the Server Certificate in pem Files .....	127
Installing LetsEncrypt Certificates in Your Linux System .....	130
Custom Authentication Connectors .....	131
Tigase Auth Connector ( <b>DEPRECATED</b> ) .....	133
Tigase Custom Auth Connector .....	134
Drupal Authentication .....	138
LDAP Authentication Connector .....	139
Configuration of SASL EXTERNAL .....	139
Packet Filtering .....	140
Domain Based Packet Filtering .....	140
Access Control Lists in Tigase .....	142
9. Database Management .....	145
Recommended database versions .....	145
Database Watchdog .....	145
Using modified database schema .....	146

Database Preparation .....	146
Schema Utility .....	147
Prepare the MySQL Database for the Tigase Server .....	150
Prepare the Derby Database for the Tigase Server .....	155
Prepare the MS SQL Server Database for the Tigase Server .....	156
Prepare the PostgreSQL Database for the Tigase Server .....	159
Preparing Tigase for MongoDB .....	161
Hashed User Passwords in Database .....	165
Shortcut .....	165
Full Route .....	166
Tigase Server and Multiple Databases .....	168
Importing User Data .....	171
Importing Existing Data .....	173
Connecting the Tigase Server to MySQL Database .....	173
Integrating Tigase Server with Drupal .....	173
PostgreSQL Database Use .....	174
Schema Updates .....	175
Changes to Schema in v8.0.0 .....	175
Tigase Server Schema v7.2 Updates .....	179
10. Components .....	183
Advanced Message Processing - AMP XEP-0079 .....	183
First of all: plugins .....	183
Secondly: component .....	184
Optional parameters .....	184
Server Monitoring .....	185
Setting Up Remote Monitoring in the Server .....	185
Retrieving statistics from the server .....	187
Eventbus .....	194
Configuration of statistics loggers .....	200
Server to Server Protocol Settings .....	201
Number of Concurrent Connections .....	201
Connection Throughput .....	202
Maximum Packet Waiting Time and Connection Inactivity Time .....	202
Custom Plugin: Selecting s2s Connection .....	202
skip-tls-hostnames .....	203
ejabberd-bug-workaround .....	203
Tigase Load Balancing .....	203
Available Implementations .....	203
Configuration Options .....	204
Auxiliary setup options .....	206
External Component Configuration .....	207
External Component Configuration .....	207
Tigase as an External Component .....	209
Load Balancing External Components in Cluster Mode .....	215
Load Balancing External Component .....	216
External Component and Cluster .....	217
Client to Server Communication .....	219
Configuration .....	219
Connections .....	219
Resumption timeout .....	220
Packet Redelivery .....	220
Tigase External Service Discovery .....	221
Setup & Configuration .....	221
11. Using Tigase .....	222

Offline Messages .....	222
Offline Message Limits .....	223
Storing offline messages without body content .....	224
Disabling Offline Messages .....	226
Last Activity .....	226
What updates last activity .....	226
Persist everything to repository .....	226
Tigase Log Guide .....	227
install.log .....	227
derby.log .....	227
etc/config-dump.properties .....	227
logs/tigase.log.# .....	227
logs/statistics.log.# .....	228
logs/tigase.pid .....	228
logs/tigase-console.log .....	228
Log File Location .....	229
Debuging Tigase .....	229
Configuration .....	229
Basic System Checks .....	230
Add and Manage Domains (VHosts) .....	230
Using Admin UI .....	231
Using ad-hoc commands .....	233
SSL Certificate Management .....	238
Presence Forwarding .....	238
Watchdog .....	240
Setup .....	240
Watchdog Configuration .....	241
Logic .....	241
Testing .....	243
Tips and Tricks .....	243
Tigase Tip: Checking the Runtime Environment .....	244
Licensing .....	247
Registering for a License .....	247
What happens if I do not use a license file or it is expired? .....	248
Demo mode .....	248
Unauthorized use .....	252
Manual mode .....	252
Tigase Clustering .....	252
Configuration .....	252
Old configuration method .....	254
Checking Cluster Connections .....	254
Anonymous Users & Authentication .....	256
Anonymous Authentication .....	256
Anonymous User Features .....	256
Scripting support in Tigase .....	257
Scripting Introduction - Hello World! .....	257
Tigase Scripting Version 4.4.x Update for Administrators .....	265
Tigase and Python .....	268
12. Appendix I - Statistics description .....	271
Data source statistics .....	271
User repository statistics of {repo} .....	271
Auth repository statistics of {repo} .....	272
Statistics common to custom {compname} component repositories .....	273
Statistics common to components .....	274

Component statistics .....	281
AMP .....	281
bosh .....	281
c2s .....	281
cl-comp .....	281
eventbus .....	283
message-archive .....	283
message-router .....	284
monitor .....	285
muc .....	286
proxy .....	286
pubsub .....	287
repo-factory .....	298
rest .....	298
s2s .....	298
sess-man .....	300
vhost-man .....	312
ws2s .....	312
13. Appendix II - Properties Guide .....	313
General .....	313
admins .....	313
Certificate Container .....	313
Component .....	314
config-type .....	315
debug-packages .....	315
debug .....	316
monitoring .....	316
plugins .....	317
priority-queue-implementation .....	318
roster-implementation .....	319
s2s-secret .....	319
scripts-dir .....	320
ssl-container-class .....	320
stats .....	320
stream-error-counter .....	323
stringprep-processor .....	324
test .....	324
tls-jdk-nss-bug-workaround-active .....	325
trusted .....	325
Repository .....	325
authRepository .....	325
authRepository .....	327
Cluster .....	328
cl-comp .....	328
cluster-mode .....	329
cluster-nodes .....	329
User connectivity .....	330
bosh-close-connection .....	330
bosh-extra-headers-file .....	330
client-access-policy-file .....	330
client-port-delay-listening .....	331
cross-domain-policy-file .....	332
domain-filter-policy .....	332
see-other-host .....	333

watchdog_timeout .....	333
watchdog_delay .....	334
watchdog_ping_type .....	334
ws-allow-unmasked-frames .....	335
External .....	335
bind-ext-hostnames .....	335
default-virtual-host .....	336
ext .....	336
Performance .....	336
cm-ht-traffic-throttling .....	336
cm-traffic-throttling .....	337
elements-number-limit .....	337
hardened-mode .....	337
max-queue-size .....	338
net-buff-high-throughput .....	338
net-buff-standard .....	338
nonpriority-queue .....	339
VHost / domain .....	339
vhost-anonymous-enabled .....	339
vhost-disable-dns-check .....	339
vhost-max-users .....	340
vhost-message-forward-jid .....	340
vhost-presence-forward-jid .....	341
vhost-register-enabled .....	341
vhost-tls-required .....	342
Tigase Server Extras - mDNS support .....	342
Overview .....	342
Enabling mDNS .....	342
Using different domain name .....	343
Forcing single server for domain .....	343
14. HTTP API component .....	344
Available modules .....	344
Admin UI module .....	344
Index module .....	344
REST module .....	344
Server status module .....	344
Setup module .....	344
Web UI module .....	344
User Status Endpoint module .....	344
Common module configuration .....	345
Enabling/disabling module .....	345
Context path .....	345
List of virtual hosts .....	345
Complex example .....	345
Module specific configuration .....	346
Rest Module .....	346
DNS Web Service module .....	347
Enabling password reset mechanism .....	348
Admin UI Guide .....	348
A Note about REST .....	348
General overview of the UI .....	349
Configuration .....	349
Example Scripts .....	349
Notifications .....	350

Other .....	350
Scripts .....	359
Statistics .....	359
Users .....	360
Tigase Web Client .....	361
Chat .....	362
Discovery .....	363
Management .....	363
15. HTTP File Upload component .....	371
Enabling HTTP File Upload Component .....	371
Metadata repository .....	371
DummyFileUploadRepository .....	371
JDBCFileUploadRepository .....	371
Storage .....	371
DirectoryStore .....	372
Logic .....	372
URI template format .....	372
File upload expiration .....	373
Examples .....	373
Complex configuration example .....	373
Example configuration for clustering with HA .....	374
16. HTTP server .....	375
Dependencies .....	375
Configuration Properties .....	375
Additional properties of embedded HTTP server .....	376
Examples .....	376
HTTPS on port 8443 with SSL certificate for example.com .....	376
Changing port from 8080 to 8081 .....	376
Usage of Jetty HTTP server as HTTP server .....	376
17. Tigase Message Archiving Component .....	378
Tigase Message Archiving Component .....	378
Announcement .....	378
Major changes .....	378
New features .....	378
Additional features .....	379
Querying in all messages .....	379
Querying by part of message body .....	379
Querying by tags .....	380
Automatic archiving of MUC messages .....	381
Database .....	382
Preparation of database .....	382
Upgrade of database schema .....	382
Schema description .....	382
Configuration .....	384
Custom Database .....	384
XEP-0136 Support .....	384
Support for MAM .....	384
Setting default value of archiving level for message on a server .....	384
Setting required value of archiving level for messages on a server .....	385
Enabling support for tags .....	385
Configuration of automatic archiving of MUC messages .....	386
Purging Information from Message Archive .....	387
Using separate store for archived messages .....	388
Setting Pool Sizes .....	388

Message Tagging Support .....	389
Usage .....	390
XEP-0136 Field Values .....	390
Manual Activation .....	392
Limitations .....	392
18. Tigase PubSub Component .....	393
PubSub Component .....	393
Announcement .....	393
Configuration .....	394
Database .....	398
Features .....	403
AdHoc Commands .....	403
REST API .....	409
Limitations .....	438
Addressing .....	438
19. Tigase Socks5 Proxy .....	440
Overview .....	440
Installation .....	440
Database Preparation .....	440
Configuration .....	441
Enabling proxy .....	441
Using a separate database .....	443
Performance .....	444
20. Tigase Push Component .....	445
Tigase Push Component .....	445
Workflow .....	445
Configuration .....	445
Enabling component .....	445
Enabling push notifications for offline messages .....	446
Enabling push notifications for messages received when all resources are AWAY/XA/ DND .....	446
Usage .....	447
Sending notifications .....	447
Registering device .....	447
Unregistering device .....	447
Providers .....	448
Tigase Push Component - FCM provider .....	448
Overview .....	448
Configuration .....	448
Tigase Push Component - APNs provider .....	449
Overview .....	449
Configuration .....	449
21. Tigase STUN Component .....	451
Tigase STUN Component .....	451
What is STUN? .....	451
Requirements .....	451
Configuration .....	451
Setting descriptions .....	451
Logback configuration .....	452
22. Tigase SPAM Filter .....	453
Overview .....	453
Configuration .....	453
Changing active SPAM filters .....	453
Sending error when packet is dropped .....	453

Enabling logging of dropped messages .....	454
Filters .....	454
Same long message body .....	454
Error message and missing <error/> child .....	455
Groupchat messages sent to bare JID .....	455
Known spammers .....	455
Presence subscription filter .....	457



---

# Chapter 1. Tigase XMPP Server 8.0.0 announcement

Tigase XMPP Server 8.0.0-RC1 Change notes and announcement

## Major Changes

### Kernel and beans configuration

Tigase now operates using a Kernel and Beans style of programming. What does this mean for Tigase and You? Good news, really. Tigase XMPP Server is now working as a Kernel program, which will operate on it's own and handle all the core functionality of the server. Component, and non-essential functionality will now be loaded as Beans. As a user, your experience will not change all that much. However, beans can be loaded and unloaded without having to restart Tigase, meaning that the program will behave more dynamically. This means a smaller footprint on memory on resources when components are not needed, and longer uptimes without having to rest art the program! This also allows for greater flexibility for Tigase XMPP Server to be better customized for unique solutions.

### New Configuration File Format

With the change of Tigase to a Kernel and Beans style of programming, we have also changed how the configuration file is managed. Although you will still edit the `config.tdsl` file like a plaintext file, a new style of formatting will be used known as DSL. Domain Specific Language may add more lines, but is a cleaner format, and provides a more secure configuration design since validation of the configuration is done at the domain level. For more information on this format and how to configure Tigase, visit DSL Configuration Guide.

### Cluster Node Shutdown Changes

Starting with Tigase XMPP Server 8.0.0, users connected on clustered nodes will be able use a `see-other-host` strategy when a node is being shutdown. **Note: This may not be compatible with all clients.** The Ad-hoc command is designed for a graceful shutdown of cluster nodes as a groovy script `Shutdown.groovy`. This script also allows for the `-timeout` setting which will delay shutdown of the node, and alert all users (via a headline message) that the server will be shutdown after a time. User clients that are compatible with the command will then detect other connected clusters and maintain their connections.

If the command is being sent to shut down the whole cluster, no `see-other-host` implementation will be sent, however timeout settings may still be used.

The script may be activated by an ad-hoc command, or sent using REST from remote or Tigase Admin UI.

### Significant cleanup of code and repositories

Multiple changes have been made to the structure and coding for v8, many related to trimming size of repositories and old calls. Some of these improvements are listed here:

- Empty JavaDocs that do not convey values have been removed.
- All code is reformatted to be compliant with out codestyle guidelines.

- Calls to `System.out.print*()` and `printStackTrace()` have been removed from code.
- Depreciated and unused classes have been removed.

## BouncyCastle being used for StartTLS

BouncyCastle [<https://www.bouncycastle.org/java.html>] Crypto API has now been employed to handle StartTLS negotiation. By doing this, Tigase now supports `tls-unique` within the SCRAM PLUS authentication implementation. This API may be employed by calling the class in your configuration file:

```
c2s () {  
    sslContextContainer(class: tigase.extras.bcstarttls.BCSSLContextContainer) {}  
}
```

The BouncyCastle classes are included in the dist-max archives.

## default-virtual-host property changes

Default virtual hosts property is now able to be configured only as a domain name instead of the list of virtual host domains with options. Additional virtual host domains and their options need to be configured using ad-hoc commands or web AdminUI. Reference Virtual-Hosts Configuration for more details.

## All artifacts are signed

Since work began on v8.0.0 Tigase has required that all changes to Tigase XMPP Server and dependencies be signed with known certificates. This version marks the first to be totally signed.

## Scaled Down Installation Methods

We have cleaned up installation methods for Tigase and now recommend the use of web-installer method. IzPack installer (files `tigase-server-<version>-b<build>.jar` installation methods have been removed and will no longer be produced for v8.0.0 and later. Manual installation is still available for those unable to use HTTP or browser access. Visit our Quick Start guide for instructions on these other methods.

## Emojis now supported on Tigase XMPP Servers

Emojis are now supported on MySQL databases, however some settings may be need to be changed, although they won't affect existing databases. Visit this section for details.

## XEP-0215 External Service Discovery now supported

Tigase now supports XEP-0215 - External Service Discovery [<https://xmpp.org/extensions/xep-0215.html>] allowing Tigase to discover services that are not available VIA the XMPP Protocol. For setup and configuration information visit External Service Discovery Component documentation.

## XEP-0313 Message Archive Management now supported

XEP-0313 - Message Archive Management [<https://xmpp.org/extensions/xep-0313.html>] is now supported by Tigase featuring custom enhancements like full-text search and searching by tags. MAM requires Tigase's message archive to be enabled in the `config.tds1` file, and the schema (XEP-0136 or

XEP-0313) must be configured in session manager settings. To turn on MAM, see configuration guide located here.

## XEP-0363 HTTP File Upload now supported

XEP-0363 - HTTP File Upload [<https://xmpp.org/extensions/xep-0363.html>] is now supported using Tigase HTTP API component now allowing for a more robust one-to-many file uploading option. Configuration details are available at the HTTP File Upload Component section of documentation.

## Startup now uses bootstrapping

Tigase now uses bootstrapping to startup, which will load configuration from `config.tds1` file like before. Then Tigase will begin it's normal operations with the configuration options. All startup functions for Tigase will now run under the `bootstrap` bean.

## CAPTCHA system now available for in-band registration

XEP-0077 In band registration [<https://xmpp.org/extensions/xep-0077.html>] can use Data Forms as an option to process new registrations. Now you can secure these registrations by employing a CAPTCHA solution. By enabling this option you can reduce the number of potential spammers and bots on your server.

## Schema changes

Now each component has it's own schema for databases, they are no longer tied into Tigase XMPP server versions making changes and updates to individual components easier, and may not disrupt all users not using certain components. See the schema update section for more details.

## Shrinkable Statistics History

Statistics history can now be automatically made smaller if a systems memory resources are above a certain amount. By default this is enabled and will trigger when over 95% of memory is in use. Half of all existing entries will be removed at this time. The same pattern will continue to halve the available records every time the threshold is met. A hard-set minimum of 5 entries is set, so you will always have the last 5 entries. This setting may be adjusted by adding the following setting to your `config.tds1` file and adjusting the integer value:

```
stats() {  
    -'stats-high-memory-level' = 95  
}
```

## Statistics now available for all modules

For any bean, you may enable statistics by using the following

```
bean (class) {  
    statistics = true  
}
```

## Spam Protection

Tigase XMPP Server v8.0.0 now includes some efforts to prevent spam bot accounts from running on servers.

## Account Registration Limits Expanded

Account registration limits have been expanded and now you can set separate counters, or configure components individually for their own limits. Visit this section for configuration details.

## Accounts created using in-band registration now will use confirmation E-mail

In an effort to create a more secure method for implementing `JabberIqRegister` Tigase XMPP Server will now require the use of a confirmation E-mail by default in the process. The E-mail must be valid, and accounts will be made into pending status until a user clicks the generated URI in the E-mail and activates the account. This is a plugin and must be enabled in the `config.tds1` file by using the following code:

```
'account-registration-email-validator'() {}
```

## Further Spam prevention

Tigase-spam component is now in `dist-max` distribution package, and has a number of features described here in this section.

## Changes in password storage

Before version 8.0.0, user passwords were stored in plaintext in the `user_pw` database field within `tig_users` table, but in plaintext. It was possible to enable storage of the MD5 hash of the password instead, however this limited authentication mechanism SASL PLAIN only. However an MD5 hash of a password is not really a secure method as it is possible to revert this mechanism using rainbow tables.

Therefore, we decided to change this and store only encrypted versions of a password in PBKDF2 `form which can be easily used for `SCRAM-SHA-1 authentication mechanism or SCRAM-SHA-256. SASL PLAIN mechanism can also use these encrypted passwords.

The storage of encrypted passwords is now enabled **by default** in v8.0.0 of Tigase.

## Dynamic TLS Buffer

Memory Buffer for TLS no longer remains at highest buffer size needed for the server session. Buffer will now free memory during idle connections. Thus drastically improving program footprint.

## XEP-305 Quickstart now supported

It's now possible to establish connection faster due to implementation of XEP-0305: XMPP Quickstart [<https://xmpp.org/extensions/xep-0305.html>] (#1936 [<https://tigase.tech/issues/1936>]). Feature is only available for `c2s` Connection Manager (i.e. connections on port 5222) and needs to be enabled in `config.tds1`

```
c2s () {  
  -'pipelining' = true  
}
```

## Database Timestamps

Timestamps in database will be stored using UTC time.

## Config-type properties have changed

Config-type is now configured using DSL format. Visit this section for more information. The names of different config-type properties have changed: `default` replaces `--gen-config-def`, `--gen=config-all`, and `--gen-config-default` configuration types. `session-manager` replaces `--gen-config-sm`. `connection-managers` replaces `--gen-config-cs`. `component` replaces `--gen-config-comp`. `setup` - is a new type of config created for initial configuration of Tigase XMPP Server.

### Note

Old versions are no longer supported, you HAVE to replace old versions with the new ones manually when upgrading to v8.0.0.

## Database Watchdog implemented

It is now possible to set connection testing to databases when connections are idle and customize the frequency with which this is done. Visit this section for more details.

## Packet statistics expanded

Packet statistics both retrieved VIA XMPP and during graceful shutdown have now been separated to a per-XMLNS basis. This may be disabled by adding the following line to `config.tdsl` file:

```
'detailed-other-statistics' = false
```

## XEP-0016 Behavior changes

XEP states that Privacy lists should be used when no user session exists in addition to when there is. Previously, Tigase would only filter results when retrieving messages, allowing blocked users to store offline messages. This has now been changed to reflect the XEP properly, and messages will be filtered while there is no user session. If however, you wish to use the previous version, where offline messages are cached first and then filtered, you may use the following configuration:

```
'sess-man' {
  -'jabber:iq:privacy' () {
    privacyListOfflineCache (active: true) {
      size = 20000
    }
  }
}
```

By default, the cache has a limit of 10000 entries, that may be set by using size bean as seen above.

## Access Control List has new ACL modifiers

New permissions have been added to ACL including `DOMAIN_OWNER` and `DOMAIN_ADMIN` to reduce permissions checking, and add another level of fine-grained permissions. For more details, please see Tigase ACL configuration for more details.

## Option to ignore schema-version check added

You can now skip the schema check phase for individual databases. To do this, add the following to the datasource configuration block:

```
DataSource () {  
    default () {  
        - 'schema-management' = false  
    -}  
}
```

This will do the following:

- Print a warning during repository startup.
- Skip schema upgrades for the source.
- Skip schema destruction for the source.

## Protection against brute-force attacks

Version 8.0.0 improves security by preventing brute-force attacks. Feature needs to be explicitly enabled and configured (on per VHost basis). Detailed configuration is described in the section called “Brute-force attack prevention” (#8160 [<https://tigase.tech/issues/8160>])

## New Minor Features & Behavior Changes

- #611 [<https://tigase.tech/issues/611>] Support for Message of the Day is now enabled in Tigase XMPP Server and can be administered using XEP-0133 Service Administration [<http://xmpp.org/extensions/xep-0133.html#set-motd>].
- #1569 [<https://tigase.tech/issues/1569>] Re-implemented XEP-0133 Service Administration Scripts 4.3 Disable User and 4.4 Re-enable User.
- #1449 [<https://tigase.tech/issues/1449>] Monitoring modules now works in OSGi mode.
- #1706 [<https://tigase.tech/issues/1706>] auto-authorize of presence subscriptions can now be set for individual vhosts.
- #1968 [<https://tigase.tech/issues/1968>] Added a Proxy Wrapper to handle reconnections to database connection pool to help prevent deadlocking threads.
- #3511 [<https://tigase.tech/issues/3511>] Mechanism responsible for closing XMPP in SessionManager has been changed to process all packets from TCP connection before closing connection.
- #3802 [<https://tigase.tech/issues/3802>] Implementation and API of LocalEventBus and ClusteredEventBus has been unified and is now available as EventBus.
- #3918 [<https://tigase.tech/issues/3918>] Session Establishment Advertisement is now optional, bringing session establishment in line with RFC 6121 [<https://tools.ietf.org/html/rfc6121>].
- #4111 [<https://tigase.tech/issues/4111>] Changed input buffer sizing to use a ratio of 2 to 1 based on input capacity. No longer using a constant value.
- #4212 [<https://tigase.tech/issues/4212>] Database schema files have been flattened and made for better organization.
- #4501 [<https://tigase.tech/issues/4501>] CounterDataFileLogger now has an upper limit and will by default be shrunk to 75% if available disk space is 5% or less than 100MB.

- #4654 [<https://tigase.tech/issues/4654>] PubSub component has been updated and new schema uses UTF-8 encoding when hashing database lookup.
- #4776 [<https://tigase.tech/issues/4776>] Tigase DbSchemaLoader now prompts for password if one is missing from command line.
- #4788 [<https://tigase.tech/issues/4788>] Push component added to dist-max archive.
- #4814 [<https://tigase.tech/issues/4814>] SASL-SCRAM will now be automatically disabled if auth database uses encoded passwords.
- #4844 [<https://tigase.tech/issues/4844>] External components can now have SSL socket connections assigned to them.
- #4859 [<https://tigase.tech/issues/4859>] Tigase DbSchemaLoader now can support using SSL when connecting to databases.
- #4874 [<https://tigase.tech/issues/4874>] Tigase Test Suite has been updated to correspond to all changes for v8.0.0.
- #4877 [<https://tigase.tech/issues/4877>] In-memory repository implemented for **testing ONLY**.
- #4880 [<https://tigase.tech/issues/4880>] Tigase config-type settings have been reduced and changed. See this section for more details.
- #4908 [<https://tigase.tech/issues/4908>] Limited Ad-hoc execution to admin only within monitor component.
- #5005 [<https://tigase.tech/issues/5005>] Detailed logging configuration is now available in DSL format. See `xref:[customLogging]` for more details.
- #5069 [<https://tigase.tech/issues/5069>] Packet processed statistics now separates results based on XML Namespaces.
- #5079 [<https://tigase.tech/issues/5079>] Tigase DbSchemaLoader can now process multiple .sql files in one command by using a comma separated list when calling.
- #5086 [<https://tigase.tech/issues/5086>] Tigase server monitor is loaded after delay to prevent NPE during startup.
- #5149 [<https://tigase.tech/issues/5149>] StanzaReceiver and StanzaSender Components have been deprecated and are no longer part of Tigase XMPP Server. Related SQL tables `xmpp_stanza` and `short_news` have also been removed from schemas.
- #5150 [<https://tigase.tech/issues/5150>] All TigaseDB tables now use the `tig_` prefix.
- #5214 [<https://tigase.tech/issues/5214>] Check has been added if recipient exists before storing offline messages for local jid.
- #5293 [<https://tigase.tech/issues/5293>] DbSchemaLoader now will fail execution instead of skipping when encountering missing files.
- #5379 [<https://tigase.tech/issues/5379>] Server ready detection has been improved in `testrunner.sh`.
- #5397 [<https://tigase.tech/issues/5397>] Webhelp Documentation will no longer be built.
- #5422 [<https://tigase.tech/issues/5422>] Errors with Beans will now result in compact and more readable StackTrace print in console log.

- #5423 [<https://tigase.tech/issues/5423>] System configuration will now be printed to log file as `ConfigHolder.loadConfiguration` output.
- #5425 [<https://tigase.tech/issues/5425>] `GetAnyFile` and `GetConfigFile` scripts moved to `message-router` instead of `basic-conf`.
- #5429 [<https://tigase.tech/issues/5429>] Adjusted settings for Dynamic Rostering now can use separate beans for multiple implementations.
- #5430 [<https://tigase.tech/issues/5430>] `BindResource` is now set to `FINER` log level to reduce console output verbosity.
- #5475 [<https://tigase.tech/issues/5475>] Setting default environment variables is now possible in `config.tdsl` file using `env( 'env-1', 'def-value' )` lines. Details available in DSL Configuration section.
- #5496 [<https://tigase.tech/issues/5496>] `Destroy Schema` task now added to schema manager.
- #5583 [<https://tigase.tech/issues/5583>] Error messages now properly sent when offline message storage is full.
- #5674 [<https://tigase.tech/issues/5674>] All components now use UTC timestamp when interacting with databases.
- #5800 [<https://tigase.tech/issues/5800>] Better annotation of deprecated code, cleanup and removal code previously marked as deprecated.
- #5964 [<https://tigase.tech/issues/5964>] Server version is now added to JMX statistics.
- #5982 [<https://tigase.tech/issues/5982>] Remote JVM debugging configuration added to `tigase.conf` file, commented by default.
- #6038 [<https://tigase.tech/issues/6038>] Data Source pool connections are now initialized concurrently instead of one at a time, dropping initializing time.
- #6103 [<https://tigase.tech/issues/6103>] `RosterElement` no longer keeps `XMPPRe-sourceConnection` instance as it is cached elsewhere. Removal results in net improvement in memory footprint.
- #6133 [<https://tigase.tech/issues/6133>] Tigase now checks components against server version to ensure compatibility.
- #6163 [<https://tigase.tech/issues/6163>] Groovy plugin updated to v2.4.12.
- #6206 [<https://tigase.tech/issues/6206>] Separated `TigaseXMLTools` and `TigaseUtil` packages for better compatibility with JDK v9.
- #6216 [<https://tigase.tech/issues/6216>] MongoDB Driver now updated to v3.5.0.
- #6560 [<https://tigase.tech/issues/6560>] `tigase` anti-spam component now included in `tigase dist-max` archive.
- #6821 [<https://tigase.tech/issues/6821>] Improved error reporting when errors from `ConfigReader`.
- #6842 [<https://tigase.tech/issues/6842>] `DefaultTypesConverter` no longer requires case sensitive enums.



- #7082 [<https://tigase.tech/issues/7082>] `ClassUtilBean` now handles packet filtering for packets part of Tigase Server but not containing beans, other improvements to mDNS.
- #7433 [<https://tigase.tech/issues/7433>] `SeeOtherHost` no longer uses `PropertiesBeanConfigurator` to parse configuration.
- #7446 [<https://tigase.tech/issues/7446>] User credentials can now be managed with Ad-hoc commands.
- #7743 [<https://tigase.tech/issues/7743>] Improved error message when repository is not found.
- #7773 [<https://tigase.tech/issues/7773>] Ad-hoc commands can now be executed asynchronously.
- #2341 [<https://tigase.tech/issues/2341>] allow specifying `SubscriptionType` when adding buddy to avoid calling separately `.setBuddySubscription()` thus eliminating saving roster twice to database if not needed

## Fixes

- #2750 [<https://tigase.tech/issues/2750>] Multiple artifact and deprecated file cleanup. Massive code cleanup and javadoc cleaning.
- #3582 [<https://tigase.tech/issues/3582>] Schema files streamlined, and no longer embedded in code.
- #3611 [<https://tigase.tech/issues/3611>] Fixed `ThreadExceptionHandler` caused by ACS unable to read PubSub schema changes.
- #3686 [<https://tigase.tech/issues/3686>] Issues with processing XHTML-IM have been fixed, and now render correctly messages with multiple CDATA items.
- #3689 [<https://tigase.tech/issues/3689>] Packets returned from CM no longer bear the original senders' jid.
- #3803 [<https://tigase.tech/issues/3803>] New call `RouteEvent` has been added to check to list and check events and determine which should be forwarded to other nodes.
- #3822 [<https://tigase.tech/issues/3822>] Error is now thrown if listener is registered for an event that is not found in `EventBus`.
- #3910 [<https://tigase.tech/issues/3910>] Fixed NPE in `SessionManager` when session is closed during execution of `everyMinute` method.
- #3911 [<https://tigase.tech/issues/3911>] Fixed issue of dropping connections during thread load distribution.
- #4185 [<https://tigase.tech/issues/4185>] Fixed an error where messages would be duplicated on stream resumption due to a counter being reset upon reconnection.
- #4447 [<https://tigase.tech/issues/4447>] Fixed condition where expired messages in offline store would cause locks.
- #4547 [<https://tigase.tech/issues/4547>] `config.dump` file now is fully compatible with `init.tdsl` file and DSL file formatting.
- #4672 [<https://tigase.tech/issues/4672>] Fixed `UnsupportedOperationException` occurring during configuration of `WebSocketConnectionClustered`.
- #4776 [<https://tigase.tech/issues/4776>] `DBSchemaLoader` now asks for user credentials if parameter is missing. Exceptions are no longer thrown if file specified is not found.

- #4885 [<https://tigase.tech/issues/4885>] `client-port-delay-listening` no longer causes exception when called.
- #4973 [<https://tigase.tech/issues/4973>] Changed Message History query to now include a limit when selecting items, preventing an `SQLException`.
- #5005 [<https://tigase.tech/issues/5005>] Fixed an issue where disabling components would result in server shutdown.
- #5042 [<https://tigase.tech/issues/5042>] Fixed issue when implementing custom SASL providers, mechanisms and callback handler factories.
- #5066 [<https://tigase.tech/issues/5066>] Fixed issue initializing databases using MongoDB.
- #5076 [<https://tigase.tech/issues/5076>] `last_login` and `last_logout` values are now properly updated while using SASL SCRAM authentication.
- #5084 [<https://tigase.tech/issues/5084>] SCRAM now checks to see if account is disabled before retrieving password.
- #5085 [<https://tigase.tech/issues/5085>] Fixed `too many beans implemented` error in Monitor Component.
- #5088 [<https://tigase.tech/issues/5088>] Removed unnecessary SASL request processing after session is closed.
- #5118 [<https://tigase.tech/issues/5118>] Fixed NPE during query of privacy lists then `type` is missing.
- #5303 [<https://tigase.tech/issues/5303>] Fixed beans not being overridden by configuration if they were registered in `RegistrarBean` or `AbstractKernelBasedComponent`.
- #5311 [<https://tigase.tech/issues/5311>] Offline messages are no longer dumped from MongoDB when restarting server.
- #5394 [<https://tigase.tech/issues/5394>] Loading main Derby schema no longer throws exceptions.
- #5428 [<https://tigase.tech/issues/5428>] Fixed parsing of v-host per domain limit property.
- #5450 [<https://tigase.tech/issues/5450>] Server no longer automatically shuts down when default or other db can not be found or accessed.
- #5458 [<https://tigase.tech/issues/5458>] Fixed potential timeout arising from `XMPPIOService::xmppStreamOpened()` method.
- #5480 [<https://tigase.tech/issues/5480>] Fixed issue in Derby DB where obtaining offline messages results in `SQLException`.
- #5525 [<https://tigase.tech/issues/5525>] Fixed S2S `invalid-namespace` error being returned during connection establishment.
- #5587 [<https://tigase.tech/issues/5587>] Fixed unclosed `ResultSet` when storing a message to AMP-offline database in Derby causing deadlock.
- #5645 [<https://tigase.tech/issues/5645>] Added fix for possible NPE when failing to retrieve beans.
- #5670 [<https://tigase.tech/issues/5670>] `config-dump` now prints configuration for inactive components and beans to log.

- #5692 [<https://tigase.tech/issues/5692>] Messages sent with negative priority were being occasionally dropped and not processed to `OfflineMessageHandler`.
- #5727 [<https://tigase.tech/issues/5727>] Fixed potential issue with MySQL procedures not being killed properly.
- #5750 [<https://tigase.tech/issues/5750>] Statistics now filter out zero-value results unless FINEST level is requested.
- #5831 [<https://tigase.tech/issues/5831>] Fixed occurrence of `OutOfMemory` error.
- #5864 [<https://tigase.tech/issues/5864>] Fixed NPE when executing BOSH pre-bind script.
- #5867 [<https://tigase.tech/issues/5867>] Fixed NPE occurring during configuration dump.
- #6000 [<https://tigase.tech/issues/6000>] Fixed a few issues with dynamic rosters properly handling presence subscription requests.
- #6006 [<https://tigase.tech/issues/6006>] Improved configuration file and DB Schema handling.
- #6041 [<https://tigase.tech/issues/6041>] Fixed potential issue where vhosts DB could be overwritten by vhosts configuration in `init.config`.
- #6078 [<https://tigase.tech/issues/6078>] Fixed `ClusterConnectionManager` to use `custom_elements_limit` instead of a fixed value.
- #6080 [<https://tigase.tech/issues/6080>] Fixed Packet Filtering to not filter cluster node information requests.
- #6083 [<https://tigase.tech/issues/6083>] Fixed clustered mode shutting down server when certain components are disabled.
- #6135 [<https://tigase.tech/issues/6135>] Tigase now properly enabled selective TLS if not enabled globally.
- #6140 [<https://tigase.tech/issues/6140>] Fixed issue while sending server welcome message.
- #6141 [<https://tigase.tech/issues/6141>] Fixed NPE at startup.
- #6234 [<https://tigase.tech/issues/6234>] Fixed an error where an error message would repeat unnecessarily.
- #6284 [<https://tigase.tech/issues/6284>] Ad-hoc commands now refresh SSL Certificate, and restart is no longer required.
- #6293 [<https://tigase.tech/issues/6293>] Server no longer sends no response upon setting empty photo in vCard.
- #6263 [<https://tigase.tech/issues/6263>] Fixed missing namespaces in responses from adhoc commands.
- #6400 [<https://tigase.tech/issues/6400>] Added a proper error when max-queue-size is too small and server cannot start.
- #6408 [<https://tigase.tech/issues/6408>] Fixed an issue where single WebSocket frames contained multiple XML stanzas instead of one per frame.
- #6411 [<https://tigase.tech/issues/6411>] Main kernel is now called to smooth shutdown. Further, timeout periods are opened up for large instances.

- #6574 [<https://tigase.tech/issues/6574>] SSL certificate upload handling is now fixed within cluster mode.
- #6598 [<https://tigase.tech/issues/6598>] Fixed EventBus Registration connection issues between cluster nodes.
- #6658 [<https://tigase.tech/issues/6658>] Cluster connections no longer potentially keep open connection after cluster is no longer connected or available.
- #6749 [<https://tigase.tech/issues/6749>] Fixed schema parsing for DerbyDB.
- #6776 [<https://tigase.tech/issues/6776>] Fixed failing Websocket connections if header contains more than one value.
- #6875 [<https://tigase.tech/issues/6875>] Fixed an issue where C2S connections could be accepted before SessionManager was initialized.
- #7037 [<https://tigase.tech/issues/7037>] Fixed error while parsing negative values from `config.tdsl` file.
- #7055 [<https://tigase.tech/issues/7055>] Improvements to metaspace use and other memory use tweaks.
- #7304 [<https://tigase.tech/issues/7304>] Virtual host logs now properly follow log size limits.
- #7431 [<https://tigase.tech/issues/7431>] AdHoc requests between the same user with different resources are no longer dropped with `NoConnectionIdException` error.
- #7434 [<https://tigase.tech/issues/7434>] Adjusted `SeeOtherHostDualIP` to use new table name in cluster nodes database.
- #7491 [<https://tigase.tech/issues/7491>] Stacktraces from `CertificateContainer` are no longer printed to `tigase-console.log`, but will be printed to `tigase.log`.
- #7687 [<https://tigase.tech/issues/7687>] Fixed an error where connections failed after authentication timeout were marked as active after cleanup.
- #7747 [<https://tigase.tech/issues/7747>] Fixed `ClusterRepoItemEvent` serialization issues causing unsupported conversion error in cluster mode.
- #7495 [<https://tigase.tech/issues/7495>] fix issue with not all logs being obfuscated, added testcase, documentation
- #8305 [<https://tigase.tech/issues/8305>] fix issue with `SeeOtherHostDualIP` when using MongoDB

## Component Changes

### AMP

- #7301 [<https://tigase.tech/issues/7301>] Tigase AMP component now uses multiple processing threads.

### PubSub

- #5033 [<https://tigase.tech/issues/5033>] PubSub now compatible with using emojis in pubsub items.
- #5693 [<https://tigase.tech/issues/5693>] Fixed parsing configuration of SessionManager processors.

- #5766 [<https://tigase.tech/issues/5766>] PubSub now writes to all databases with UTC timestamp.
- #5953 [<https://tigase.tech/issues/5953>] Fixed presences not being removed from `presenceByService` collection if client disconnects without `<unavailable/>` presence being sent.
- #6176 [<https://tigase.tech/issues/6176>] version changed to PubSub v4.0.0.
- #7707 [<https://tigase.tech/issues/7707>] Fixed potential NPE in PubSub.

## http-api

- #4873 [<https://tigase.tech/issues/4873>] Support added to display timestamp fields as data, time, and timezone fields.
- #4876 [<https://tigase.tech/issues/4876>] Implemented using XML repository for new setups, and updated default config to use this.
- #4888 [<https://tigase.tech/issues/4888>] `http-api` now is enabled by default.
- #5209 [<https://tigase.tech/issues/5209>] Updated visual styling of pages hosted by component.
- #5290 [<https://tigase.tech/issues/5290>] Fixed invalid property name.
- #5316 [<https://tigase.tech/issues/5316>] Account Registration now can now require and send confirmation E-mails.
- #5415 [<https://tigase.tech/issues/5415>] Web Setup now checks configuration for message archive conflicts.
- #5460 [<https://tigase.tech/issues/5460>] MongoDB now supported through web-setup.
- #5717 [<https://tigase.tech/issues/5717>] Fixed default values of check-boxes in admin UI not being shown.
- #5950 [<https://tigase.tech/issues/5950>] Supported added for XEP-0363: HTTP File Upload [<https://xmpp.org/extensions/xep-0363.html>].
- #6159 [<https://tigase.tech/issues/6159>] Fixed NPE thrown if scripts directory is not present.
- #6176 [<https://tigase.tech/issues/6176>] version changed to `tigase-http-api v2.0.0`.
- #6212 [<https://tigase.tech/issues/6212>] Added mechanism for password changing through HTTP API.
- #7307 [<https://tigase.tech/issues/7307>] Fixed scripts returning 404 while handling `rest/user/` requests even though user exists.
- #7178 [<https://tigase.tech/issues/7178>] Ad-hoc commands are now categorized in groups for better organization.
- #7568 [<https://tigase.tech/issues/7568>] Added timeout reading for HTTP request headers, added configurable `accept-timeout`.

## message-archive

- #4867 [<https://tigase.tech/issues/4867>] fixed issue when changing MA jid.
- #4888 [<https://tigase.tech/issues/4888>] `message-archive` is enabled by default.

- #5033 [<https://tigase.tech/issues/5033>] Update message archive to be compatible with emojis.
- #5391 [<https://tigase.tech/issues/5391>] Added missing query statement block starts and ends to be compatible with SQL Server.
- #5604 [<https://tigase.tech/issues/5604>] Modified access to static fields and functions.
- #5681 [<https://tigase.tech/issues/5681>] Fixed duplication of groupchat messages with different ids by modifying hash algorithm.
- #6176 [<https://tigase.tech/issues/6176>] version changed to message-archive v2.0.0.
- #7615 [<https://tigase.tech/issues/7615>] `feature-not-implemented` response no longer occurs when removing stored messages.

## MUC

- #4888 [<https://tigase.tech/issues/4888>] muc now is enabled by default.
- #5033 [<https://tigase.tech/issues/5033>] MUC component is now compatible with emojis.
- #5066 [<https://tigase.tech/issues/5066>] Fixed issues working with MongoDB repository.
- #5085 [<https://tigase.tech/issues/5085>] Removed invalid annotation parameter values.
- #5559 [<https://tigase.tech/issues/5559>] Fixed NPE while changing default room configuration.
- #5666 [<https://tigase.tech/issues/5666>] User may add more than one `<item/>` elements to query when querying room members.
- #5715 [<https://tigase.tech/issues/5715>] Welcome messages may now be disabled globally, or in individual room configurations.
- #5736 [<https://tigase.tech/issues/5736>] Rooms with no subject now return empty `<subject/>` element, as per XEP-0048 7.2.16 [<https://xmpp.org/extensions/xep-0045.html#enter-subject>].
- #5813 [<https://tigase.tech/issues/5813>] Fixed NPE during room creation.
- #6176 [<https://tigase.tech/issues/6176>] version changed to tigase-muc v3.0.0.
- #6395 [<https://tigase.tech/issues/6395>] Fixed `tigase.db.UserNotFoundException` during retrieval of MUC user.
- #6734 [<https://tigase.tech/issues/6734>] Introduced `muc#roomconfig_maxresources` to allow configuration of max number of resources for a single occupant.
- #7443 [<https://tigase.tech/issues/7443>] Disabled XEP-0091 by default, added history attribute validation.

## socks5 Proxy

- #2750 [<https://tigase.tech/issues/2750>] Cleanup of code and removal of empty javadocs.
- #5867 [<https://tigase.tech/issues/5867>] Fixed NPE during configuration dump when component is disabled.
- #6176 [<https://tigase.tech/issues/6176>] version changed to tigase-socks5 v2.0.0.

## stats

- #5206 [<https://tigase.tech/issues/5206>] Fixed exception causing duplicate error entry.
- #5728 [<https://tigase.tech/issues/5728>] Fixed `MySQLIntegrityConstraintViolationException` in upload handler.
- #6161 [<https://tigase.tech/issues/6161>] Removed usage of classes from `javax.xml.ws` package for JDKv9 compatibility.

## STUN Server

- #6176 [<https://tigase.tech/issues/6176>] version changed to `tigase-stun v2.0.0`.

## WebSocket

- #6481 [<https://tigase.tech/issues/6481>] Websocket component has been improved to be more compliant with `rfc6455` [<https://tools.ietf.org/html/rfc6455>]

---

# Chapter 2. Tigase User Guide

Tigase Team <team@tigase.com [mailto:team@tigase.com]> v8.0.0-RC1, 2019-01-30

## Jabber/XMPP introduction

### Jabber/XMPP is Instant Messaging Technology

All federated **XMPP** servers are connected in one global communications network allowing you to send messages to friends who have accounts on other Jabber servers.

This is very much like sending e-mail but the difference between Jabber and e-mail is the same as the difference between sending a traditional mail and talking on the phone.

All messages sent through Jabber are sent instantly and you also receive responses instantly. More over you can see whether your mate is online and available for talking or not.

There exists similar technologies to Jabber like WhatsApp Messenger, Facebook Messenger, Signal, Telegram, WeChat, QQ and other. There are, however, quite a few differences.

XMPP is an open standard which means everybody can know how it works, everybody can implement their own software connecting to the network both client and server side.

The server side is actually the biggest difference and advantage. Many companies have offices in different locations, and such instant messaging technology could be very useful to employees for communication. Companies are not inclined to allow confidential discussions to go outside the company's network. Especially if it is not very secure to leave such information on third party public servers.

XMPP servers on the other hand, allows you to deploy server software on your own company network. Employees can then talk securely and all information remains on the company's secure network. Of course if offices are located in different locations or countries then all messages are transmitted over the public network - the Internet. This is not a problem since XMPP supports SSL/TLS - secure encrypted connections which helps you protect your discussion.

Then if your employees need to contact customers outside your company, the whole discussion can go through your server and a server located on the customer side.

There are many other scenarios and use cases but I hope this brief introduction gives you an idea of the differences and advantages of XMPP technology.

## How to Use Tigase Service

### This Article Describes How to use tigase.im Service for Instant Communications

You have to install and run a Jabber client application to use the service.

There are multiple domains available: tigase.im, sure.im, xmpp.cloud (and you can host your own domain as well)



## Short instructions:

Usually you just need to enter the user name of the form: `user@tigase.im` [<mailto:user@tigase.im>]. Your XMPP client should take care of all other things as our service doesn't need any special settings. If you don't have an account on `tigase.im` server yet just tick the option to register new account. That's it!

## Long Instructions:

Good news is that there are many programs to choose from which allow you to communicate through our server. So you can pick up your favorite application or use an existing one that is compatible and start using our service.

All clients presented below support multiple accounts on Jabber servers. What this means is that you can have a few Jabber accounts on different Jabber servers and you can still use just one program to connect to all of them at the same time.

The full list of all known XMPP clients [<https://xmpp.org/software/clients.html>] is very long. You can obviously try them all but below is a selection which is recommended by the Tigase team. The selected programs might not be the best choice for you, but these programs have been tested and we can offer help with using them. Here is a list of recommended instant messaging clients:

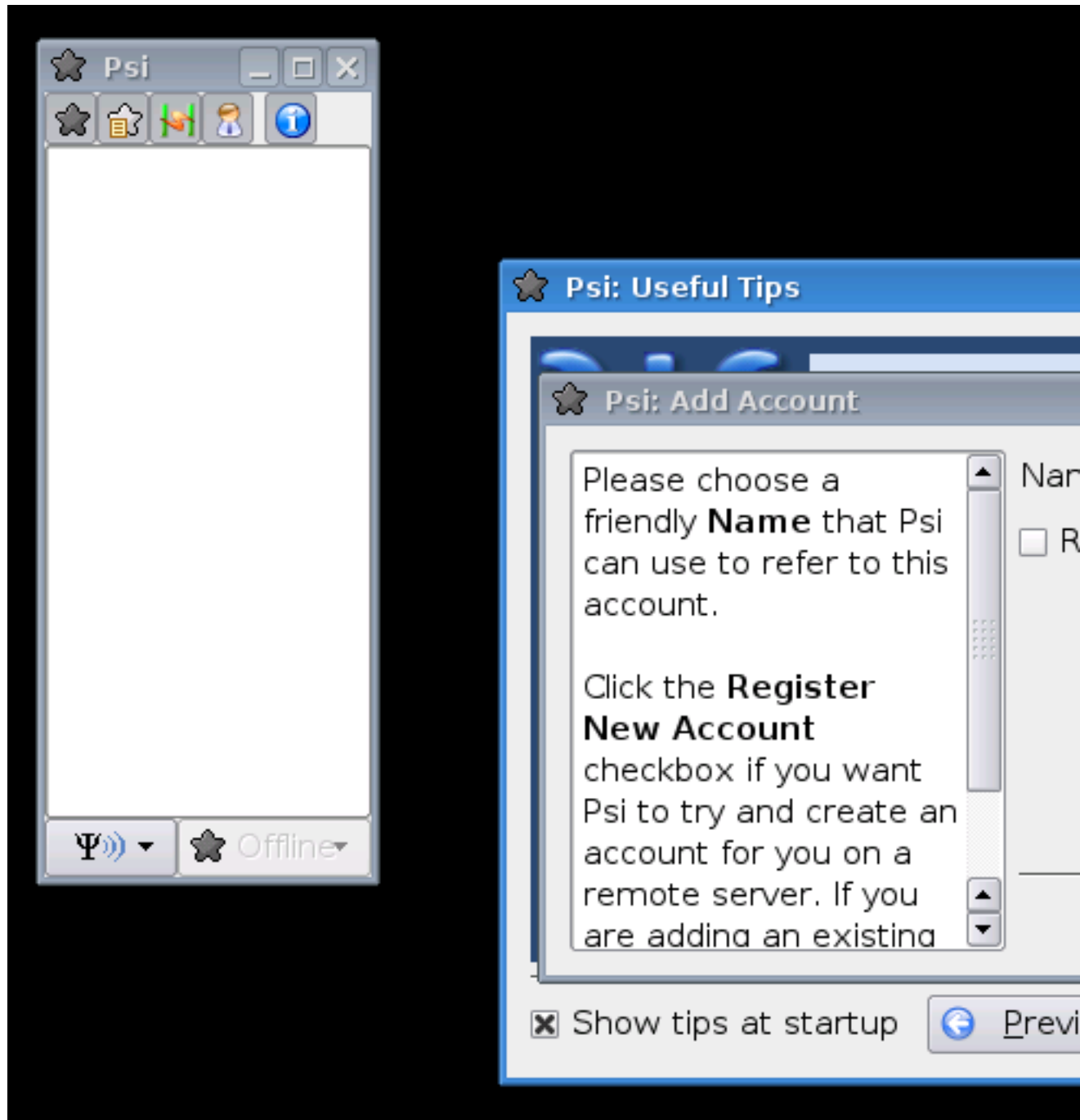
- Beagle.im [<https://beagle.im/>] - macOS desktop client developed by Tigase team supporting all the latest and greatest features
- Tigase Messenger for iOS [<https://itunes.apple.com/us/app/tigase-messenger/id1153516838>] - lightweight, powerful XMPP client developed by Tigase, Inc. It provides an easy way to start using the XMPP Protocol (formerly known as Jabber) if you've never used it before. Veterans of the protocol will find many features here they are familiar with along with enhancements to reduce data use and extend battery life.
- Tigase Messenger for Android [<https://play.google.com/store/apps/details?id=org.tigase.messenger.phone.pro>] - mobile chat client to use with XMPP services and servers. The totally revamped v3.0 now has new features, a better design, and Google integration. Application supports any XMPP server, from free services like `sure.im` or `Tigase.im`, to a server you may host on your own.
- `tigase.im`[`Tigase.im`] - web-based client allowing to easily chat with friends independently of platform.
- Psi [<http://psi-im.org/>] Pure Jabber client. Although it supports only Jabber network it is a very user friendly and comfortable program. It works on most popular operating systems like Linux, MS Windows, and Apple MacOS X.
- Gajim [<http://www.gajim.org/>] This is another Jabber only client. Very user friendly and works on most of Linux distributions, FreeBSD, and MS Windows.
- Pidgin [<http://www.pidgin.im/>] (previously Gaim [<http://gaim.sourceforge.net/>]) This is not just a Jabber client. This type of application is called multicomunicator as apart from Jabber it supports many other instant messaging networks/protocols such as: AIM/ICQ, MSN, Yahoo, Gadu-Gadu, IRC, and a few others. So it is especially convenient if you have friends using other messaging networks. Pidgin works on most Linux distributions, and on MS Windows.
- Kopete [<http://kopete.kde.org/>] This is a KDE [<http://www.kde.org/>] component and although it only works on Linux based system it also supports many of the most popular instant messaging protocols apart from Jabber like: AIM, Gadu-Gadu, ICQ, IRC, MSN, Yahoo.

Install the Jabber client of your choice and set up for a Tigase account:

# Configuration instructions for Psi

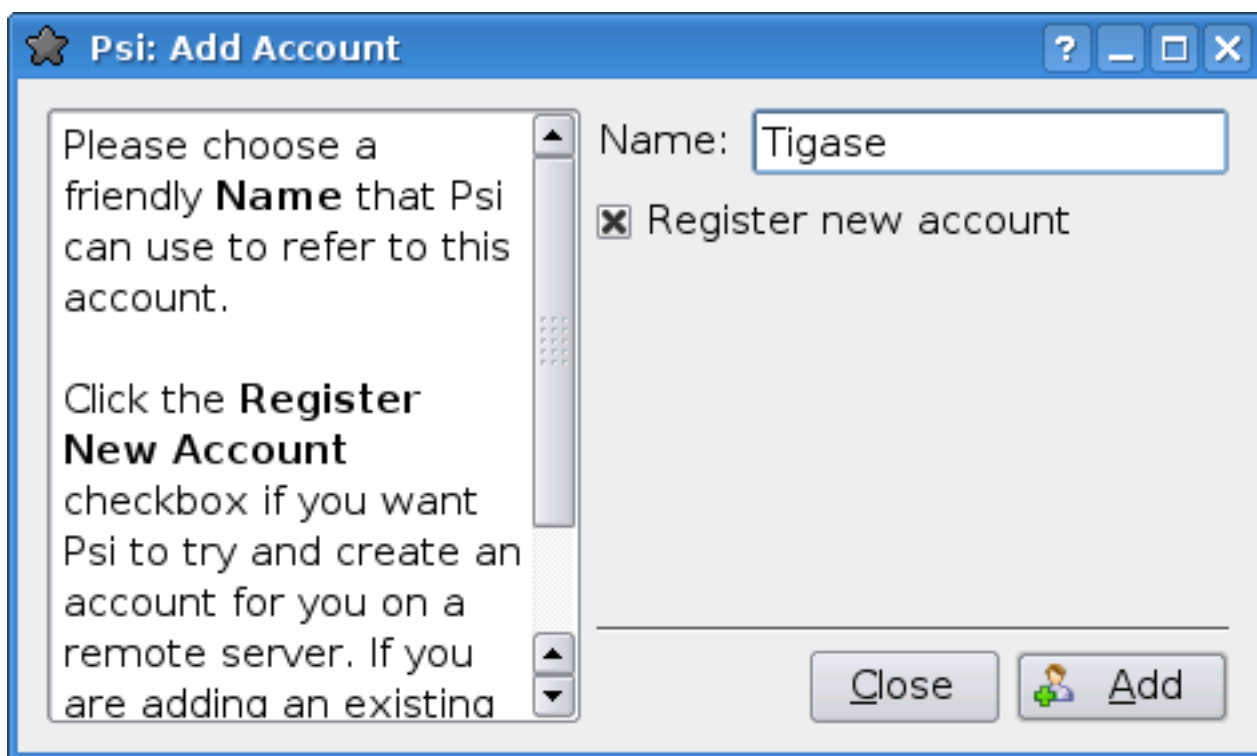
## Psi - Initial configuration

The first time you run Psi you see a screen like this:



To connect to tigase.org server we need to configure the program. Below are step-by-step instructions for novice users on how to setup Psi.

1. Psi can connect to many Jabber servers at the same time so we have to identify each connection somehow. The first thing to do is assign a name to the connection we just created. As we are going to define connection to tigase.org server let's just name it: **Tigase**.



**Note!** At the moment you can register an account through the Web site only. This is a single account for both services: The Drupal website and Jabber/XMPP service on the tigase.org domain. If you want to have a Jabber account on the tigase.org server go to the registration page, un-tick "Register new account", and go to the point no 5. You can use guide points 2-4 to register a Jabber account on any other Jabber server.

2. When you press the Add button you will see next window where you can enter your Jabber account details:



The image shows a window titled "Psi: Register Account" with a star icon and standard window controls. It contains three sections: "Account", "Proxy", and "Advanced".

**Account**

Jabber ID:   
*Example: juliet@capulet.com*

Password:

Confirm Password:

**Proxy**

None

**Advanced**

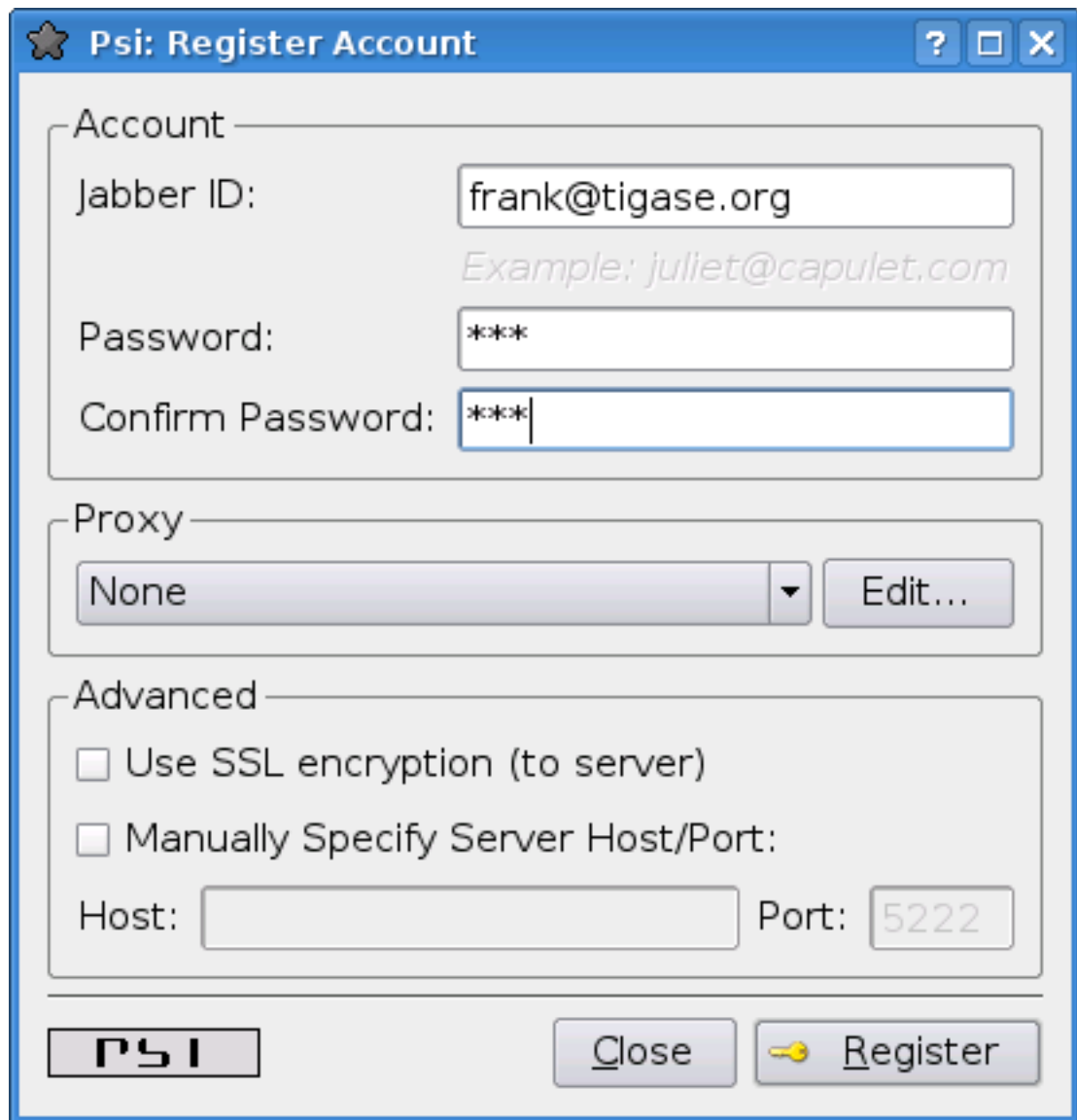
☐ Use SSL encryption (to server)

☐ Manually Specify Server Host/Port:

Host:  Port:

At the bottom, there is a "PSI" logo, a "Close" button, and a "Register" button with a key icon.

3. Invent your user name for the account on Tigase server. Let's assume your user name is: **frank**. Jabber ID's however consist of 2 parts - your user name and server address. Exactly the same as an e-mail address. As you are registering an account on [tigase.org](http://tigase.org) server, you will have to enter in this field: **frank@tigase.org**. Next enter the password of your choice and click the Register button.



The image shows a window titled "Psi: Register Account" with a star icon and standard window controls. It contains three sections: "Account", "Proxy", and "Advanced".

**Account**

Jabber ID:   
*Example: juliet@capulet.com*

Password:

Confirm Password:

**Proxy**

**Advanced**

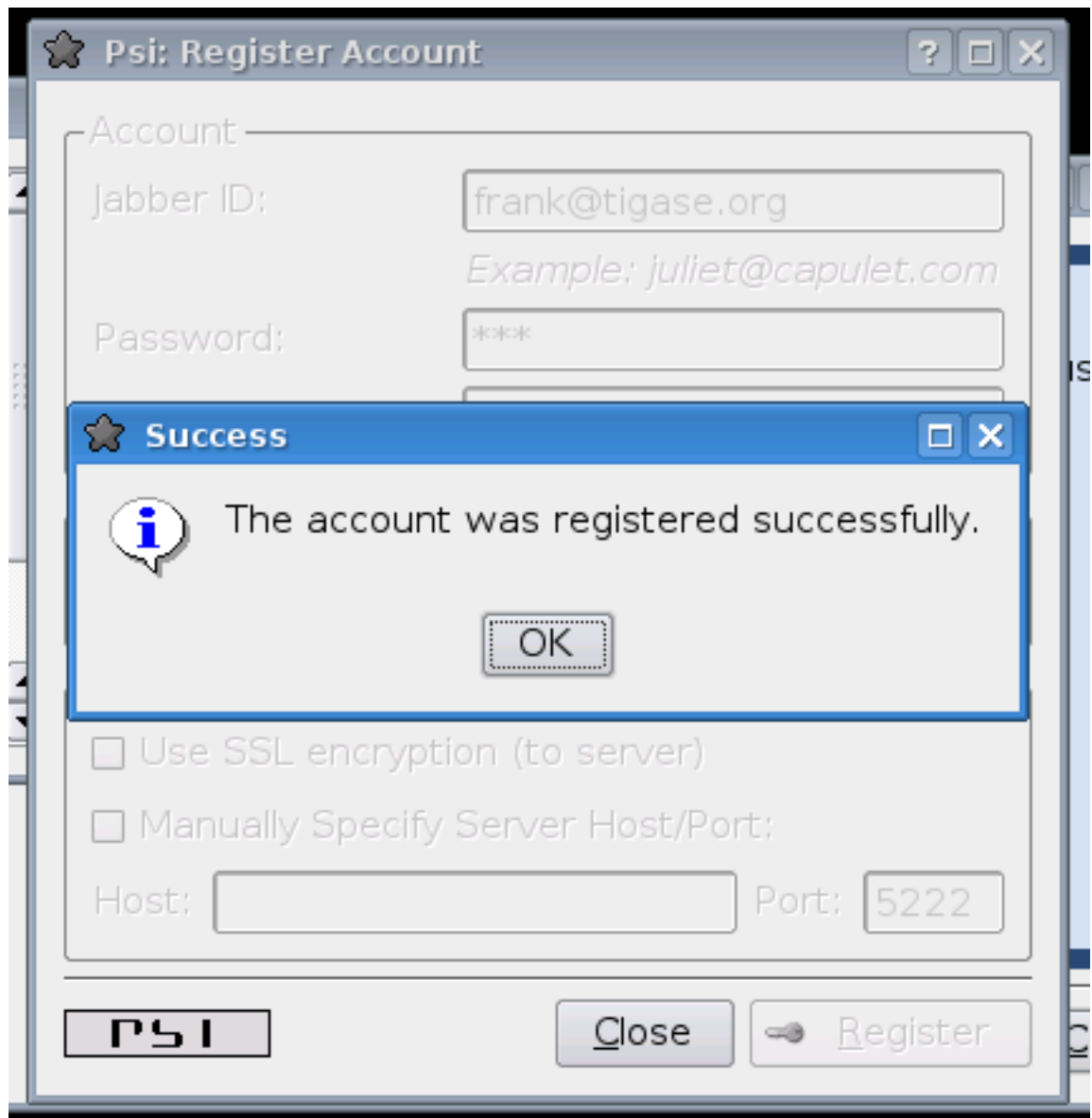
☐ Use SSL encryption (to server)

☐ Manually Specify Server Host/Port:

Host:  Port:

At the bottom, there is a "PSI" logo, a "Close" button, and a "Register" button with a key icon.

4. On successful registration you will receive a confirmation message and you should see a window like this:



It may happen that somebody earlier registered an account with the same name you've selected for yourself. If so, you will receive error message. You will then have to select another user name and try to register again.

5. After clicking the **OK** button you will see a window with your connection and account setup. You can stick with default values for now.



Just click the **Save** button and this window closes.

6. Now you have your account configured and ready to use but you are still off-line. You can find out whether you are on-line or off-line by looking at the bottom of main Psi window. There you can see **Offline** text.

Click on this **Offline** text and you will see a list of possible options. Just select **Online**.



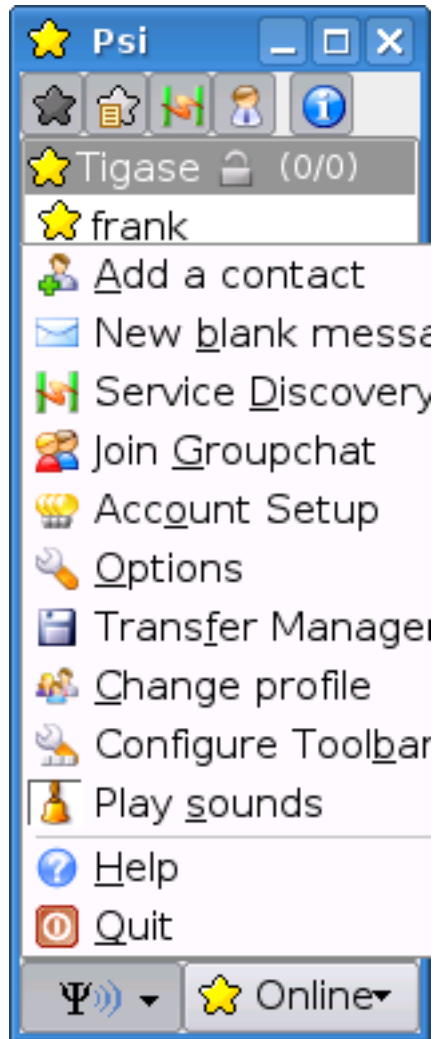
Now you are connected!

Well, you are now connected but how to talk to other people? How to add friends to the contact list? You can send a message to your friends straight away using the **Psi menu** option **New blank message**. It is much more convenient however, if you could see which of your friends is online and available for chatting and if you could start talking to your friend just by clicking on his name.

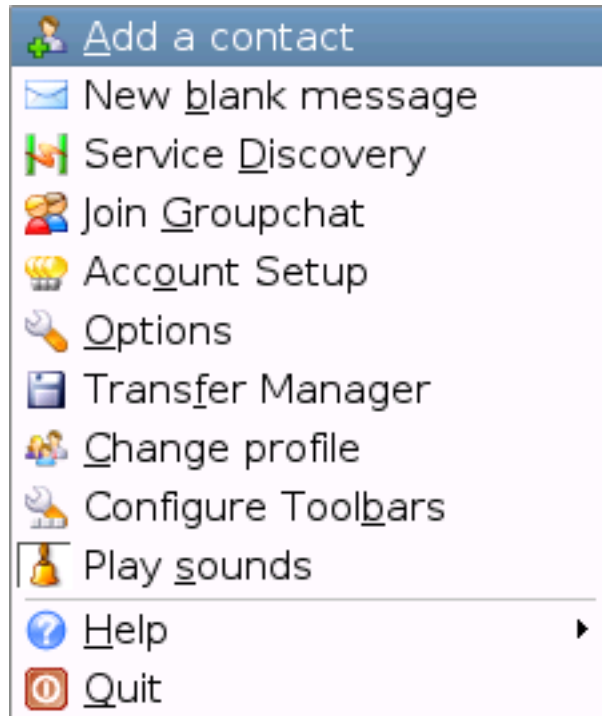
## Short Instructions How to Add Your First Contact

1. Click on Psi menu - the button next to the **Online** text. You will see something like this:

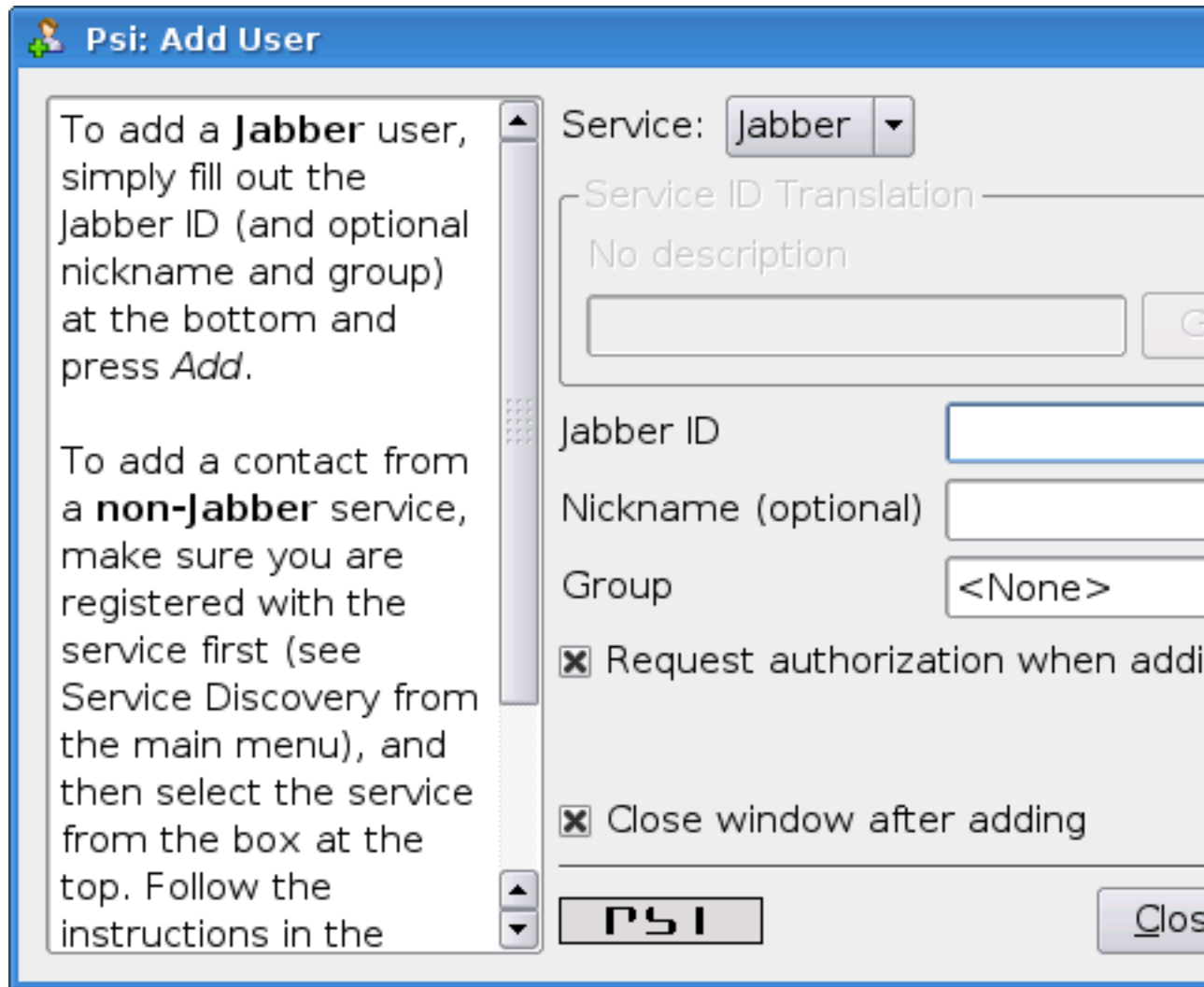




From all menu options select the top one - Add a contact:



2. The next window will display where you can enter your contact details:



The image shows a screenshot of the 'Psi: Add User' dialog box. The title bar is blue with a small icon and the text 'Psi: Add User'. The dialog is divided into two main sections. The left section contains two paragraphs of instructional text. The right section contains form fields for adding a user. The 'Service' dropdown is set to 'jabber'. The 'Service ID Translation' section is faded. The 'Jabber ID' field is empty. The 'Nickname (optional)' field is empty. The 'Group' dropdown is set to '<None>'. There are two checked checkboxes: 'Request authorization when adding' and 'Close window after adding'. At the bottom, there is a 'PSI' button and a 'Close' button.

**Psi: Add User**

To add a **Jabber** user, simply fill out the Jabber ID (and optional nickname and group) at the bottom and press *Add*.

To add a contact from a **non-Jabber** service, make sure you are registered with the service first (see Service Discovery from the main menu), and then select the service from the box at the top. Follow the instructions in the

Service: **jabber**

Service ID Translation  
No description

Jabber ID

Nickname (optional)

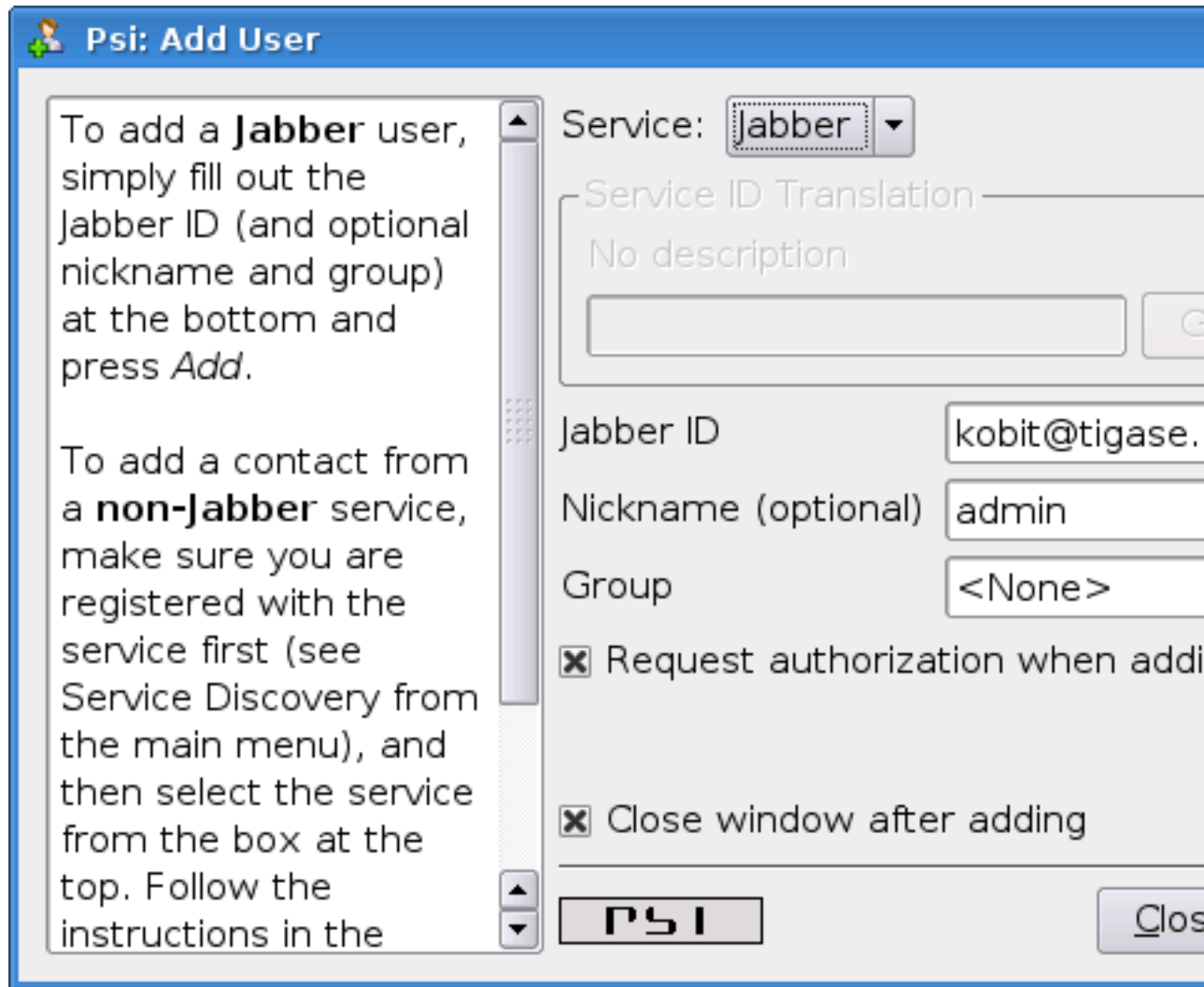
Group: <None>

☒ Request authorization when adding

☒ Close window after adding

**PSI** Close

You have to know the Jabber ID of the person you want to add to your contact list. Let's assume, for example, you want to add Tigase server administrator's Jabber ID to your contact list. So, after you enter these details the window will look like this:



The image shows a dialog box titled "Psi: Add User". On the left, there is a scrollable text area with instructions. The main area on the right contains several input fields and checkboxes. The "Service" dropdown is set to "jabber". The "Jabber ID" field contains "kobit@tigase.". The "Nickname (optional)" field contains "admin". The "Group" dropdown is set to "<None>". There are two checked checkboxes: "Request authorization when adding" and "Close window after adding". At the bottom, there is a "PSI" button and a "Close" button.

**Psi: Add User**

To add a **Jabber** user, simply fill out the Jabber ID (and optional nickname and group) at the bottom and press *Add*.

To add a contact from a **non-Jabber** service, make sure you are registered with the service first (see Service Discovery from the main menu), and then select the service from the box at the top. Follow the instructions in the

Service: **jabber**

Service ID Translation  
No description

Jabber ID: kobit@tigase.

Nickname (optional): admin

Group: <None>

☒ Request authorization when adding

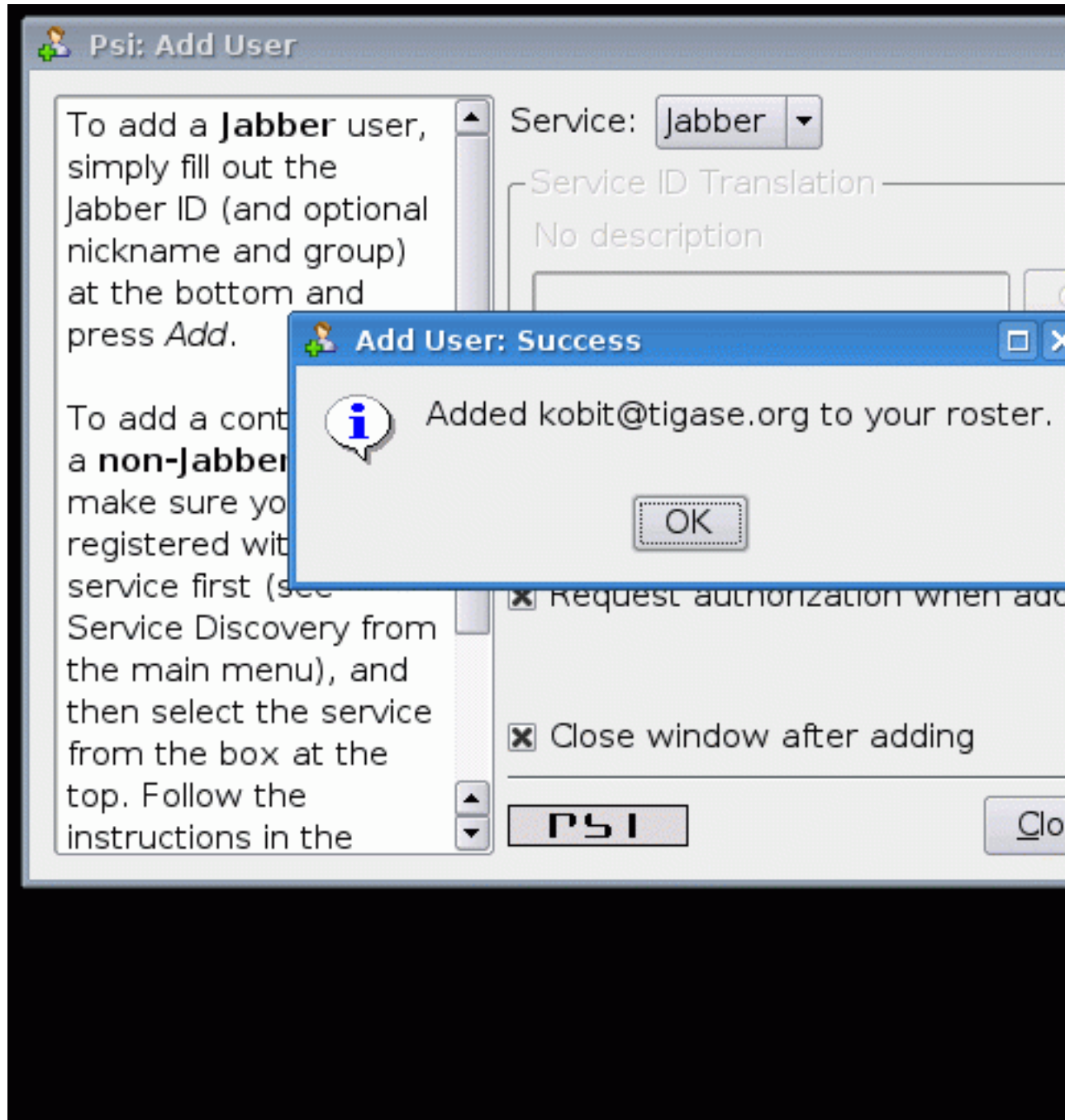
☒ Close window after adding

PSI

Close

Click the **Add** button.

- Now you will see a confirmation window that a new person has been added to your contact list:



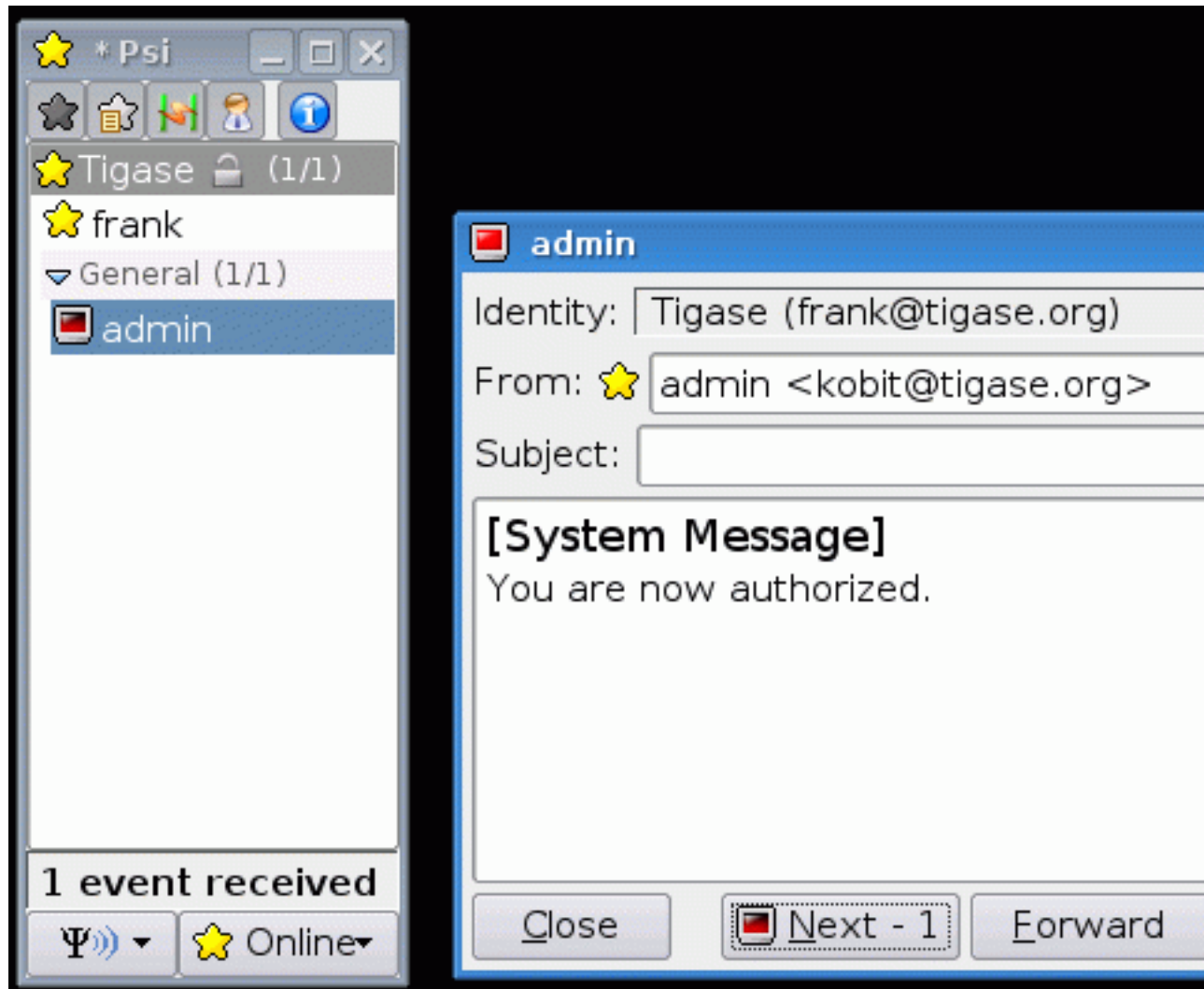
But there is more behind the scenes. Adding a contact to your **Roster** (contact list) usually means you can see whether the person is online and available to talk or not. The person however, may not wish you to see his presence. So, to make sure the other person accepts you as a friend Psi sent a request to the address you just entered with the question of whether he agrees to show his presence to you.

You won't be able to see the users availability until he sends confirmation.

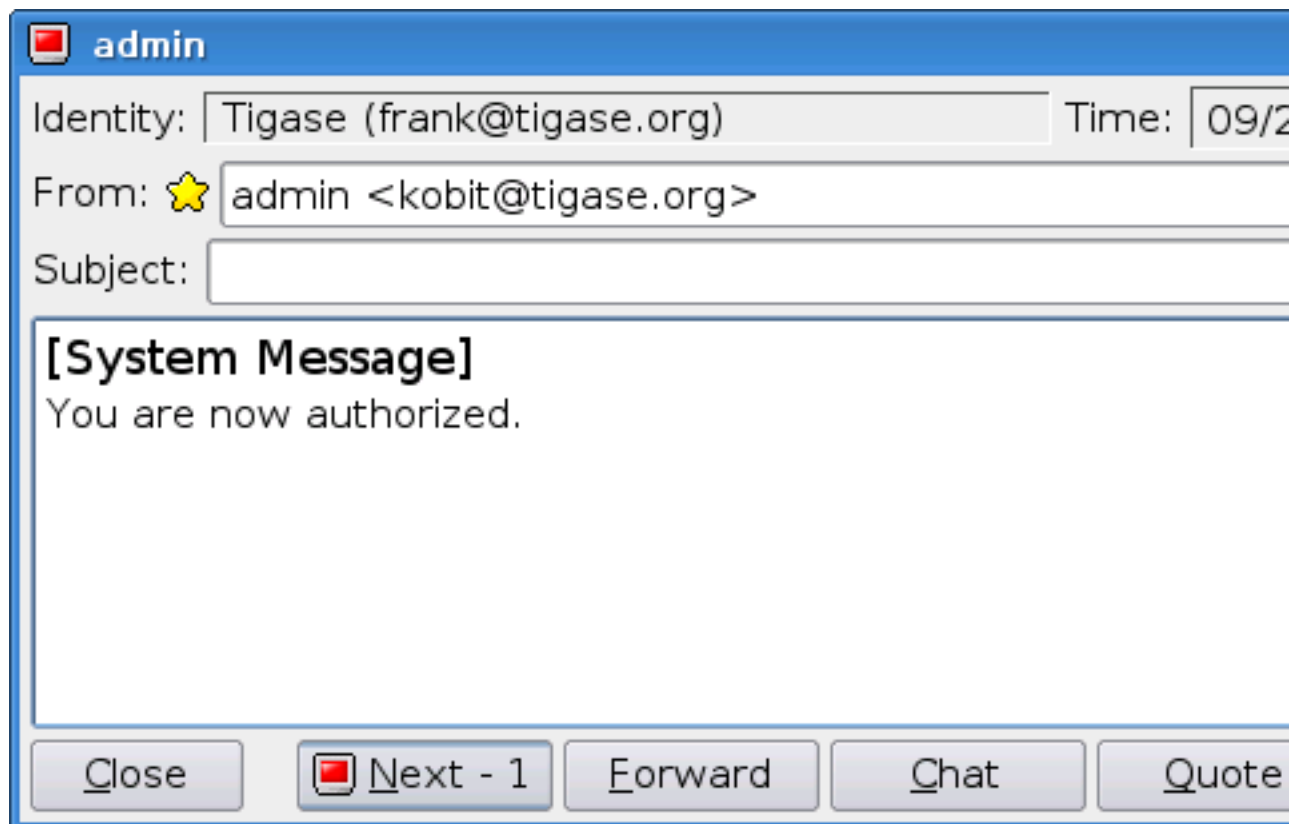
4. Once the other user sends confirmation back, you will usually receive 2 system events:



5. Click on the contact to see a window with these messages:



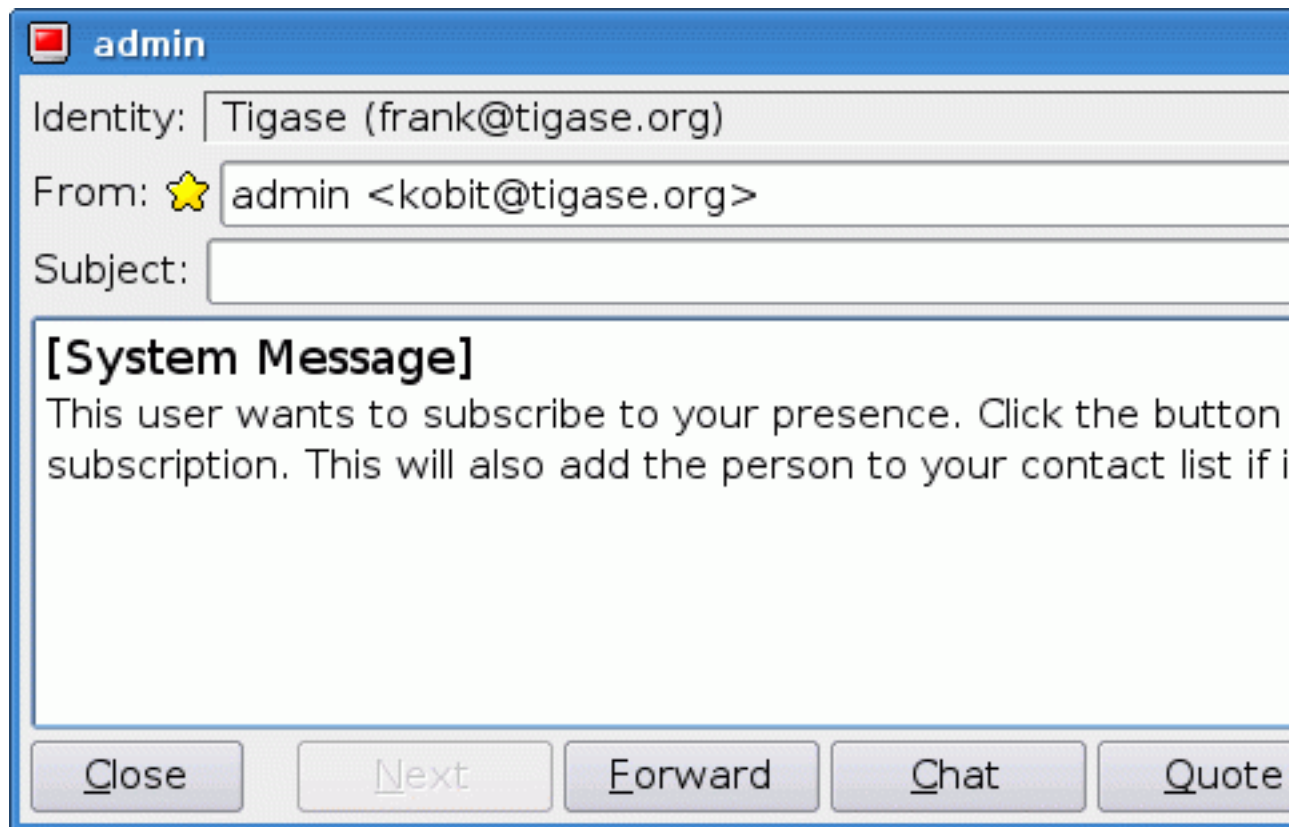
6. One message just says you have been authorized by the other user:



So you simply click **Next** to see the second message.

7. The second message is a bit more interesting. It contains the question of whether you also authorize the other user to see your presence. If you want to accept this request just click **Add/Auth**.





8. Finally main Psi window with your new contact:



Well done!

You are ready to start Jabbering. Good luck.

Where to go next? For detailed Psi documentation refer to the program Wiki page: [http://psi-im.org/wiki/Main\\_Page](http://psi-im.org/wiki/Main_Page)

Welcome to the Tigase Administration Guide.

---

# Chapter 3. About Tigase XMPP Server

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

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

## Robust and reliable

This means that the server can handle many concurrent requests/connections and can run for a long time reliably. The server is designed and implemented to handle millions of simultaneous connections.

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

## Security

There are a few elements of the security related to XMPP servers: 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** Tigase software installation does not impact network security. Companies usually have their networks divided into 2 parts: **DMZ** which is partially open to the outside world and the **Private network** which is closed to the outside world.

If the XMPP server is to provide an effective way of communication between company employees regardless if they are in a secure company office or outside (perhaps at a customer site), it needs to accept both internal and external connections. So the natural location for the server deployment is the **DMZ**. However, this solution has some considerations: each company has normally established network users base and integrated authorization mechanisms. However, that information should be stored outside the DMZ to protect internal security, so how to maintain ease of installation and system security?

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

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

## Flexibility

There are many different XMPP server implementations. The most prevalent are:

- Used as a business communication platform in small and medium companies where the server is not under a heavy load. For such deployments security is a key feature.
- For huge community websites or internet portal servers is, on the other hand, usually under very heavy load and has to support thousands or millions of simultaneous connections. For such a deployment we need a different level of security as most of the service is open to the public.
- For very small community deployments or for small home networks the key factor is ease to deploy and maintain.

Architecture based on components provides the ability to run selected modules on separate machines so the server can be easily applied in any scenario.

For simple installation the server generates a config file which can be used straight away with very few modifications or none at all. For complex deployments though, you can tweak configurations to your needs and setup XMPP server on as many physical machines as you need.

## Extensibility

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

**Tigase server** has been designed to be extensible using a modular architecture. You can easily replace components which do not fulfill your requirements with others better fitting your needs. But that is not all, another factor of extensibility is how easy is to replace or add new extensions. A great deal of focus has been put into the server design API to make it easy for other software developers to create extensions and implement new features.

## Ease of Use

Complex computer networks consisting of many servers with different services are hard to maintain. This requires employing professional staff to operate and maintain 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 and a HTTP server. They might want to add an XMPP server to the collection of their services and don't want to dedicate resources on setup and maintenance. For such users our default configuration is exactly what they need. If the operating system on the server is well configured, then Tigase should automatically pickup the correct hostname and be ready to operate immediately.

Tigase server is designed and implemented to allow dynamic reconfiguration during runtime so there is no need to restart the server each time you want to change configuration settings.

There are also interfaces and handlers available to make it easy to implement a web user interface for server monitoring and configuring.

---

# Chapter 4. Licensing and Open Source

As mentioned previously, Tigase is open source under AGPLv3. If you are not familiar with open source software, or the environment, here are some frequently asked questions that might provide some answers.

**What does open source mean?** This means that Tigase's source code is available to the public to see how Tigase works. There are no 'black boxes' for packets where things just happen, everything is out in the open, whereas other companies may consider this proprietary information. In addition, we have the benefit of many talented people working with Tigase to constantly improve Tigase server and related projects. These people not only include the Tigase development team, but other members of the community who submit code improvements, patches, enhancements, or other changes to Tigase.

**Does this mean that the binaries are open to malicious code?** Although we accept patches from contributors, our repository does not accept them directly. Code may be submitted through our [tigase.tech](http://tigase.tech) [http://tigase.tech] page and our developers will review the code before it is added. All builds are tested for functionality and security when they are built.

**Does this mean it is less secure?** Not at all. Although anybody can see the source code, and know how Tigase works; your installation, connections, and settings are uniquely yours. Tigase is regularly tested and written to be as secure as possible using the latest encryption and secure connection protocols.

**Is Tigase free?** Tigase is free for download and use in its unmodified state. Our commercial grade products such as Advanced Clustering Strategy is available for free use for testing & development.

**Does this mean I cannot use it in my product or commercial environment?** Not necessarily, consult the Affero General Public License Agreement v3 to see if your use qualifies. Tigase is offered under commercial license if your use is not covered by AGPLv3.

**Are there options for closed code or extensions?** Yes! Commercial licenses can be custom made for each client, and software written for your company may be made private or part of our open source distributions at your discretion.

**Can I contribute code?** Sure! Register an account in here [<https://projects.tigase.org/account/register>] and submit patches through our redmine!

---

# Chapter 5. Tigase Server Binary Updates

Most open source projects try to make sure that the nightly builds compile correctly so that these builds can be used. However, we at Tigase believe that these builds should be separated until they are thoroughly tested and released. Although lots of installations out there we know of just run from our nightly builds, this puts an extra responsibility to make sure all code is functional and will constantly work. Therefore, our general approach is to run all basic functionality tests before each code commit to make sure it works correctly. This does not guarantee that there will never be a problem, but it is a precaution from preventing bad builds from arriving in the hands of our customers.

Some users on the other hand, like to be on the bleeding edge and regularly use our nightly builds exploring new code changes and playing with new features before they are put to a full release. Others prefer to stick to stable and fully tested public releases. Others however, want something from the middle, the most recent features, but bug fixes, something like a beta or a release-candidate state.

Should you choose to use the nightly builds, a few things you should consider:

- Changes may be made to the code that can negatively affect performance.
- Changes may be made to the code that can negatively affect security.

We **highly** recommend testing these builds in your environments before upgrading.

With these considerations in mind, we provide nightly builds at this link [<https://build.tigase.net/nightlies/dists/>] which provides directories by date.

Standard naming format is `tigase-server-<version>-SNAPSHOT-b<build>-<type>` where `<version>` is in the form of `major.minor.bugfix`

## Note

individual days may have the same builds as noted by the byyyy section of the file.\*

Just like the standard distributions, the builds are available with the following extensions (`<type>`):

1. `javadoc.jar` - Java installer for javadoc only
2. `dist.zip` - Compressed binaries with no dependencies.
3. `dist.tar.gz` - tarball compressed binaries with no dependencies.
4. `dist-max.zip` - Compressed binaries with all dependencies.
5. `dist-max.tar.gz` - tarball compressed binaries with all dependencies.

We also provide automated testing of each of our nightly builds for each supported databases. Tests are done with both functional and low memory parameters in mind, and are available at this link [<https://build.tigase.net/nightlies/tests/>]. These tests can provide a quick examination of function before upgrading your current build.

---

# Chapter 6. Quick Start Guide

## Minimum Requirements

Before you begin installing Tigase server onto your system, please make sure the minimum requirements are met first:

- **Java Development Kit v8 or later** - We recommend OpenJDK
- **Administrator access** - We recommend that you install Tigase Server from a user login with administrator access.

## Contents

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>].

- Installation Using Web Installer
- Manual installation in console mode
- Installing Tigase on Windows
- Network settings for Tigase
- Running Tigase XMPP Server as a service

## Installation Using Web Installer

When Tigase XMPP Server starts up, it looks for the default configuration file: `etc/config.tdsl`. If this file has not been modified you can run the web installer. Which will step you through the process of configuring Tigase. If you are installing Tigase in a Windows environment, please see the Windows Installation section.

## 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.tigase.org/nightlies/dists/>]. Once you have the distribution binary extract it and navigate to the directory:

```
$ tar --xf tigase-server-<version>-dist-max.tar.gz
$ cd tigase-server-<version>
```

### Tip

Do not run as root user!

## Start the Server

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

## 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:

This setup page is restricted access, however for first setup there is a default account set to setup Tigase:  
Username: admin Password: tigase

This combination will only be valid once as it will be removed from `config.tdsl` file on completion of setup process. After this point the setup page will only be accessible using the following:

1. JID accounts listed as administrators in admins line in `config.tdsl` file.
2. Username and password combinations added to `config.tdsl` file manually, or at the last page in this process.

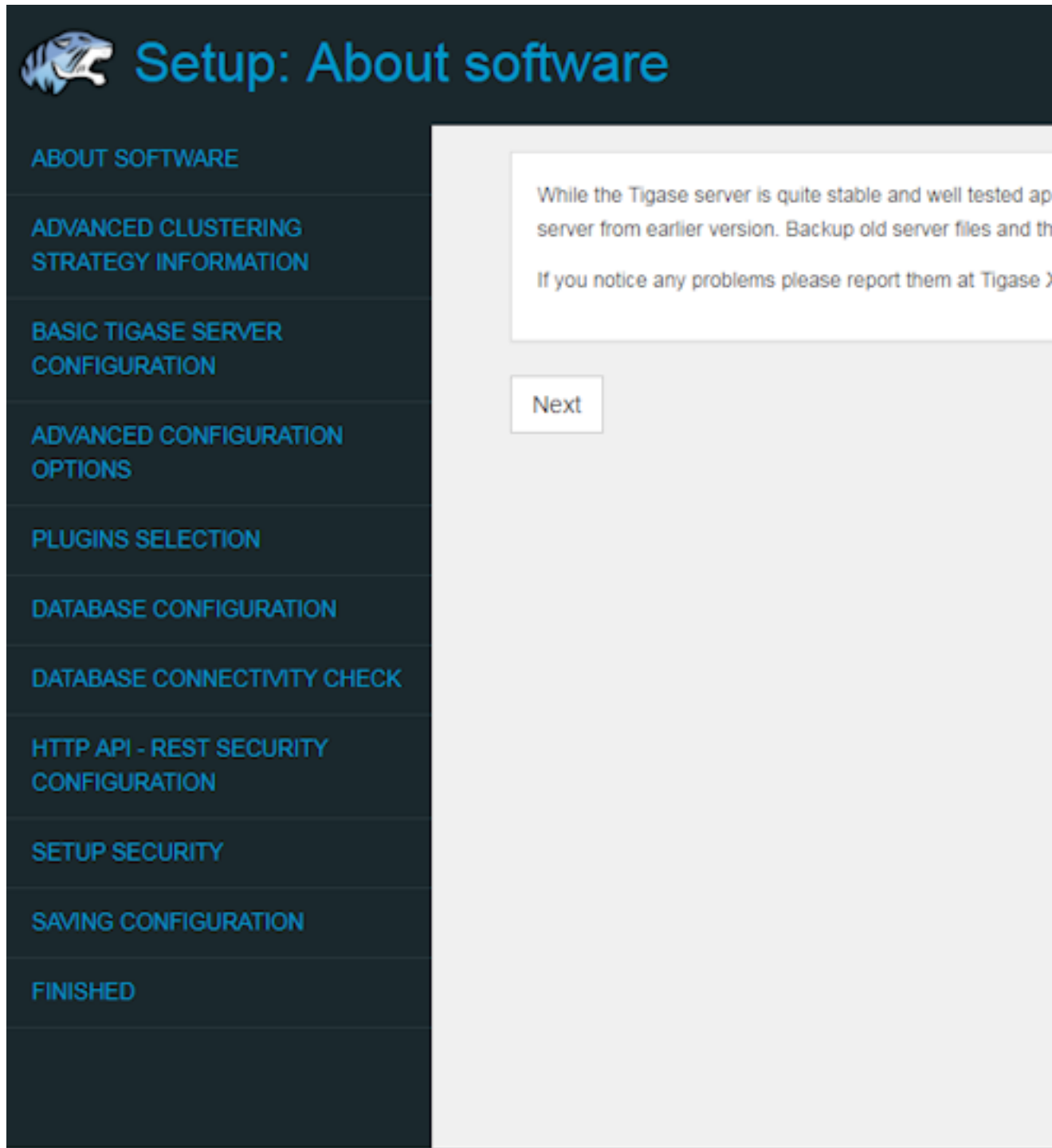
Point your browser to **`http://localhost:8080/setup/`** unless you are working remotely. You can also use the domain name, or IP address.

Enter the username and password above to gain access.

## Step Through the Installation Process

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





Read it and then click "Next"



# Setup: Advanced Clustering S

ABOUT SOFTWARE

ADVANCED CLUSTERING  
STRATEGY INFORMATION

BASIC TIGASE SERVER  
CONFIGURATION

ADVANCED CONFIGURATION  
OPTIONS

PLUGINS SELECTION

DATABASE CONFIGURATION

DATABASE CONNECTIVITY CHECK

HTTP API - REST SECURITY  
CONFIGURATION

SETUP SECURITY

SAVING CONFIGURATION

FINISHED

This installation package contains  
use for the term of your agreement

ACS is not open source software,

The free trial granted hereunder d  
connection with production system  
purposes only. Any use of ACS oth

If you activate the ACS software u  
regular basis. If ACS cannot acces  
representative to resolve the issue  
information will be sent to Tigase's

If you enjoy your free trial version  
software. The full commercial vers

By activating this free trial version  
names, number of online users, n  
accept and confirm that such infor  
consent to such transfer and waiv  
information transfer.

To confirm please enter your na

Next

Here is some information about our commercial products and licensing. Please read though the agreement, type in your name or company and click "Next".



# Setup: Basic Tigase server co

ABOUT SOFTWARE

ADVANCED CLUSTERING  
STRATEGY INFORMATION

**BASIC TIGASE SERVER  
CONFIGURATION**

ADVANCED CONFIGURATION  
OPTIONS

PLUGINS SELECTION

DATABASE CONFIGURATION

DATABASE CONNECTIVITY CHECK

HTTP API - REST SECURITY  
CONFIGURATION

SETUP SECURITY

SAVING CONFIGURATION

FINISHED

On this panel you can specify bas

Based on your selection here mor  
created.

You can optionally restart the serv

**Configuration type**

**Your XMPP (Jabber) domains**

**Server administrators**


**Admin password**

**Select database**

**Advanced configuration  
options**

Next

This page will look over your basic configuration settings, these include the server type, domains you wish to use, and gives you a chance to specify an administrator for the domain. Also, you will be selecting what type of database Tigase server will be using (configuration comes later). If you do not specify an administrator and password, one is made for you, which will be `admin@yourdomain` and password is `tigase`. If you wish to configure your server beyond the basics, check Advanced configuration options.



# Setup: Advanced configuration

ABOUT SOFTWARE

ADVANCED CLUSTERING  
STRATEGY INFORMATION

BASIC TIGASE SERVER  
CONFIGURATION

ADVANCED CONFIGURATION  
OPTIONS

PLUGINS SELECTION

DATABASE CONFIGURATION

DATABASE CONNECTIVITY CHECK

HTTP API - REST SECURITY  
CONFIGURATION

SETUP SECURITY

SAVING CONFIGURATION

FINISHED

This panel offer advanced configuration options for what you are doing.

Select optional components

IM AM

Bosh connection

Client connection

Distributed

External Service Discovery

HTTP server integrati

Message Archiving C

Monitor C

The Advanced configuration page. Select what components and configurations you need. Some may be highlighted red to indicate conflicts or unmet requirements.



# Setup: Plugins selection

ABOUT SOFTWARE

ADVANCED CLUSTERING  
STRATEGY INFORMATION

BASIC TIGASE SERVER  
CONFIGURATION

ADVANCED CONFIGURATION  
OPTIONS

**PLUGINS SELECTION**

DATABASE CONFIGURATION

DATABASE CONNECTIVITY CHECK

HTTP API - REST SECURITY  
CONFIGURATION

SETUP SECURITY

SAVING CONFIGURATION

FINISHED

Please select/deselect

Advanced Message Processing



Plugins which will be loaded by the server, most plugins are enabled by default. Some may be highlighted red to indicate conflicts or unmet requirements.



# Setup: Database configuration

ABOUT SOFTWARE

ADVANCED CLUSTERING  
STRATEGY INFORMATION

BASIC TIGASE SERVER  
CONFIGURATION

ADVANCED CONFIGURATION  
OPTIONS

PLUGINS SELECTION

**DATABASE CONFIGURATION**

DATABASE CONNECTIVITY CHECK

HTTP API - REST SECURITY  
CONFIGURATION

SETUP SECURITY

SAVING CONFIGURATION

FINISHED

You have selected derb  
Please enter all required

Name of the database that will be

tigasedb

Address of the database instance

localhost

Name of the user that will be cre

tigase\_user

Password of the user that will be

.....

Database root account username

root

Database root account password

....

Enable SSL support for databases  
(if your database supports it)

This is where the database is setup. The type of database selected in step 3 will influence available options.  
**BE SURE TO SPECIFY DATABASE ROOT USER ACCOUNT AND PASSWORD**



# Setup: Database connectivity

ABOUT SOFTWARE

ADVANCED CLUSTERING  
STRATEGY INFORMATION

BASIC TIGASE SERVER  
CONFIGURATION

ADVANCED CONFIGURATION  
OPTIONS

PLUGINS SELECTION

DATABASE CONFIGURATION

**DATABASE CONNECTIVITY CHECK**

HTTP API - REST SECURITY  
CONFIGURATION

SETUP SECURITY

SAVING CONFIGURATION

FINISHED

You have selected data

Checking connection to

Checking if database e

Loading schema: Tigas

Loading schema: Tigas

Loading schema: Tigas

Loading schema: Tigas

Adding XMPP admin a

Post installation action

You should see a page like this after a successful database setup. This page will reveal any issues with your database setup such as invalid URIs, passwords, and schemas. You may hit Back on your browser to check credentials and settings and try again.



# Setup: HTTP API - REST security

ABOUT SOFTWARE

ADVANCED CLUSTERING  
STRATEGY INFORMATION

BASIC TIGASE SERVER  
CONFIGURATION

ADVANCED CONFIGURATION  
OPTIONS

PLUGINS SELECTION

DATABASE CONFIGURATION

DATABASE CONNECTIVITY CHECK

HTTP API - REST SECURITY  
CONFIGURATION

SETUP SECURITY

SAVING CONFIGURATION

FINISHED

## Configuration of security

Select security model of REST API

- ☐ Access forbidden (REST API disabled)
- ☒ Access requires API keys:
- ☐ Open access

Next

Next a page asking if you'd like to provide an API Access Key to access HTTP REST commands. It is highly recommended that you either specify an API key or block access. Open API keys allow any REST command to be interpreted by the server.



# Setup: Setup security

ABOUT SOFTWARE

ADVANCED CLUSTERING  
STRATEGY INFORMATION

BASIC TIGASE SERVER  
CONFIGURATION

ADVANCED CONFIGURATION  
OPTIONS

PLUGINS SELECTION

DATABASE CONFIGURATION

DATABASE CONNECTIVITY CHECK

HTTP API - REST SECURITY  
CONFIGURATION

**SETUP SECURITY**

SAVING CONFIGURATION

FINISHED

## Security of setup.

By default Tigase XMPP Server is

However in rare conditions it may  
*etc/init.properties* file.

Please enter username and password

Username:

Password:

Next



The Setup Access Page will be locked from the admin/tigase user as specified above. This is your chance to have the setup pages add a specific user in addition to admin accounts to re-access this setup process later. If left blank, only JIDs listed in admin will be allowed to access.



# Setup: Saving configuration

ABOUT SOFTWARE

ADVANCED CLUSTERING  
STRATEGY INFORMATION

BASIC TIGASE SERVER  
CONFIGURATION

ADVANCED CONFIGURATION  
OPTIONS

PLUGINS SELECTION

DATABASE CONFIGURATION

DATABASE CONNECTIVITY CHECK

HTTP API - REST SECURITY  
CONFIGURATION

SETUP SECURITY

**SAVING CONFIGURATION**

FINISHED

## Installation of Tigase XMPP Server

Configuration created during installation

```
admins = [  
  'Admin@xmppserver.com'  
]  
'config-type' = 'default'  
debug = [ 'server' ]  
'virtual-hosts' = [ 'XMPPServer'  
dataSource () {  
  default () {  
    uri = 'jdbc:derby:tigase'  
  }  
}  
'sess-man' () {  
  'http://jabber.org/protocol'  
}
```

Do you wish to save the configuration?

The installation is complete and this is what the `config.tdsl` file will look like. If you have a custom setup, or would like to put your own settings in, you may copy and past the contents here to edit the current `config.tdsl` file. Click "Save" to write the file to disk.



# Setup: Finished

ABOUT SOFTWARE

ADVANCED CLUSTERING  
STRATEGY INFORMATION

BASIC TIGASE SERVER  
CONFIGURATION

ADVANCED CONFIGURATION  
OPTIONS

PLUGINS SELECTION

DATABASE CONFIGURATION

DATABASE CONNECTIVITY CHECK

HTTP API - REST SECURITY  
CONFIGURATION

SETUP SECURITY

SAVING CONFIGURATION

**FINISHED**

Installation of Tigase X

*etc/init.properties* confi  
new configuration.

You have now finished the installation, proceed to the next step to restart the server.

## Restart the Server

It is recommended at this point to stop the server manually and restart it using the proper script for your OS. From the Tigase base directory enter

```
./scripts/tigase.sh stop  
  
./scripts/{OS}/init.d/tigase start etc/tigase.conf
```

where {OS} is your type of Linux, gentoo, debian, mandriva, or redhat

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.

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

## Windows Instructions for using Web Installer

There are a few steps involved with setting up Tigase with the web installer in a Windows environment. Please follow this guide.

First step is to extract the distribution archive in it's entirety to the intended running directory. Once there, run the `Setup.bat` file inside the `win-stuff` folder. This will move the necessary files to the correct folders before Tigase begins operation.

From here, you have a few options how to run Tigase; `run.bat` will operate Tigase using a java command, or `tigase.bat` which will start Tigase using the wrapper. You may also install Tigase and run it as a service.

One this setup is finished, web installer will continue the same from here.

## Manual Installation in Console Mode

Our preferred way to install the Tigase server is using Web 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 [<https://tigase.tech/projects/tigase-server/files>].

In many cases however it is not always possible to use the web 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.

### Note

Please note that these instructions are for \*nix operating systems, and some modifications may be required for other Operating Systems!

## Get the Binary Package

Have a look at our download area [<https://projects.tigase.org/projects/tigase-server/files>]. 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-<version>-b<build>-<type>.<archive>`, where `<version>` is in the form of `major.minor.bugfix`, `<type>` can be either `dist` (basic package) or `dist-max` (extended set of components) and archive type can be either `tar.gz` or `zip`.

## 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 88
drwxr-xr-x 2 tigase tigase 4096 Aug 15 18:17 certs
-rw-r--r-- 1 tigase tigase 0 Aug 15 18:26 ChangeLog
drwxr-xr-x 2 tigase tigase 12288 Aug 15 18:17 database
drwxrwxr-x 4 tigase tigase 4096 Oct 12 09:48 docs
drwxrwxr-x 2 tigase tigase 4096 Oct 12 09:48 etc
drwxrwxr-x 2 tigase tigase 4096 Oct 12 09:48 jars
-rw-r--r-- 1 tigase tigase 34203 Aug 15 18:26 License.html
drwxr-xr-- 2 tigase tigase 4096 Aug 15 18:26 logs
-rw-r--r-- 1 tigase tigase 3614 Aug 15 18:26 package.html
-rw-r--r-- 1 tigase tigase 2675 Aug 15 18:26 README
drwxr-xr-x 9 tigase tigase 4096 Aug 15 18:17 scripts
drwxr-xr-x 5 tigase tigase 4096 Aug 15 18:17 tigase
drwxrwxr-x 4 tigase tigase 4096 Oct 12 09:48 win-stuff
```

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

```
$ ls --l etc/
total 36
-rw-r--r-- 1 tigase tigase 153 Aug 15 18:11 bosh-extra-headers.txt
-rw-r--r-- 1 tigase tigase 325 Aug 15 18:11 client-access-policy.xml
-rw-r--r-- 1 tigase tigase 124 Aug 15 18:11 config.tdsl
-rw-r--r-- 1 tigase tigase 263 Aug 15 18:11 cross-domain-policy.xml
```

```
-rw-r--r-- 1 tigase tigase 2337 Aug 15 18:11 jmx.access
-rw-r--r-- 1 tigase tigase 2893 Aug 15 18:11 jmx.password
-rw-r--r-- 1 tigase tigase  735 Aug 15 18:11 logback.xml
-rw-r--r-- 1 tigase tigase 3386 Aug 15 18:11 snmp.acl
-rw-r--r-- 1 tigase tigase 1346 Aug 15 18:11 tigase.conf
```

## Configure tigase.conf

Tigase.conf is a file that contains general program operating parameters, and java settings for Tigase to run. For now, the only setting we need to set is the **JAVA\_HOME** directory.

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

Replace `${JDKPath}` with a path to Java JDK installation on your system.

## Configure config.tdsl

You need also to edit the `config.tdsl` file. It contains initial parameters normally set by the configuration program. As this is a manual installation, you will have to edit this document yourself. It contains already a few lines:

```
'config-type' = -'setup'

http () {
    setup () {
        -'admin-user' = -'admin'
        -'admin-password' = -'tigase'
    }
}
```

You will need to set a few things in order to get Tigase up and running.

### Step 1: Change config-type

Refer to `config-type` property description for details, but for most operations, change `setup` to `default`.

### Step 2: Set virtual host

Without a virtual host, your XMPP server has no domain with which to operate. To set a virtual host use the following configuration:

```
'default-virtual-host' = -'hostname'
```

You have to replace `hostname` with a domain name used for your XMPP installation. Let's say this is **jabber.your-great.net**. Your setting should look like this:

```
'default-virtual-host' = -'jabber.your-great.net'`
```

There are many other settings that can be configured visit this section for details.

### Step 3: Set Administrators

At least one administrator is required, and once the database is setup will have the default password of `tigase`. Be sure to change this once you have finished setting up your server. To add admins, use the following line in the `config.tdsl` file:

```
`admins = [ -'admin@jabber.your-great.net', -'user2jabber.your-great.net' -]`
```

## Step 4: Set databases

You will also need to configure connection to the database. First you have to decide what database you want to use: Derby, MySQL, PostgreSQL, MSSQL, or MondoDB. Each database will have slightly different configurations. If we are using derby, in a directory called `tigasedb`, your configuration would look like this:

```
dataSource () {  
    default () {  
        uri = -'jdbc:derby:tigasedb;create=true'  
    -}  
}
```

Consult `dataSource` property for more configuration info.

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

## Install Database

Creating the database is the next step. Previously, we had scripts to handle this process, but we now have the advantage of functions in the `tigase.sh` script that can be used. Setting up the database can now be done using a single command.

```
./scripts/tigase.sh install-schema etc/tigase.conf --T derby --D tigasedb --H local
```

This command will install tigase using a Derby database on one named `tigasedb` hosted on `localhost`. The username and password editing the database is `tigase_pass` and `root`. Note that `-J` explicitly adds the administrator, this is highly recommended with the `-N` passing the password. You may customize this command as needed, refer to the `install-schema` section of the documentation for more information.

On a windows system, you need to call the program directly:

```
C:\tigase>java --cp -"jars/*" tigase.db.util.SchemaManager -"install-schema" --T derby
```

If this successfully passes, you should see some information printed out

```
LogLevel: CONFIG  
2017-10-12 20:05:47.987 [main] DBSchemaLoader.init()  
Oct 12, 2017 8:05:48 PM tigase.util.DNSResolverDefault <init>  
WARNING: Resolving default host name: ubuntu took: 7  
Oct 12, 2017 8:05:49 PM tigase.db.util.SchemaManager loadSchemas  
INFO: found 1 data sources to upgrade...  
Oct 12, 2017 8:05:49 PM tigase.db.util.SchemaManager loadSchemas  
INFO: begining upgrade...  
LogLevel: CONFIG  
2017-10-12 20:05:49.877 [main] DBSchemaLoader.init()  
2017-10-12 20:05:49.877 [main] DBSchemaLoader.validateDBConnection()  
2017-10-12 20:05:50.932 [main] DBSchemaLoader.validateDBConnection()  
2017-10-12 20:05:50.932 [main] DBSchemaLoader.validateDBConnection()  
2017-10-12 20:05:50.933 [main] DBSchemaLoader.validateDBExists()  
2017-10-12 20:05:50.936 [main] DBSchemaLoader.withConnection()  
2017-10-12 20:05:50.937 [main] DBSchemaLoader.lambda$validateDBExists$
```



```
2017-10-12 20:05:50.939 [main] DBSchemaLoader.loadSchemaFile()
2017-10-12 20:05:50.941 [main] DBSchemaLoader.withConnection()
2017-10-12 20:05:51.923 [main] DBSchemaLoader.lambda$loadSchemaFile$28
2017-10-12 20:05:51.925 [main] DBSchemaLoader.loadSchemaFile()
2017-10-12 20:05:51.926 [main] DBSchemaLoader.withConnection()
2017-10-12 20:05:52.209 [main] DBSchemaLoader.lambda$loadSchemaFile$28
2017-10-12 20:05:52.210 [main] DBSchemaLoader.loadSchemaFile()
2017-10-12 20:05:52.211 [main] DBSchemaLoader.withConnection()
2017-10-12 20:05:52.305 [main] DBSchemaLoader.lambda$loadSchemaFile$28
2017-10-12 20:05:52.306 [main] DBSchemaLoader.loadSchemaFile()
2017-10-12 20:05:52.307 [main] DBSchemaLoader.withConnection()
2017-10-12 20:05:52.731 [main] DBSchemaLoader.lambda$loadSchemaFile$28
2017-10-12 20:05:52.732 [main] DBSchemaLoader.addXmppAdminAccount()
2017-10-12 20:05:52.732 [main] DBSchemaLoader.addXmppAdminAccount()
Oct 12, 2017 8:05:52 PM tigase.db.jdbc.DataRepositoryImpl initialize
INFO: Table schema found: jdbc:derby:tigasedbx;create=true, database type: derby,
Oct 12, 2017 8:05:52 PM tigase.db.jdbc.DataRepositoryImpl initialize
INFO: Initialized database connection: jdbc:derby:tigasedbx;create=true
2017-10-12 20:05:52.884 [main] DBSchemaLoader.addXmppAdminAccount()
2017-10-12 20:05:52.884 [main] DBSchemaLoader.postInstallation()
2017-10-12 20:05:52.891 [main] DBSchemaLoader.withConnection()
2017-10-12 20:05:52.892 [main] DBSchemaLoader.lambda$postInstallation$
2017-10-12 20:05:52.893 [main] DBSchemaLoader.lambda$postInstallation$
2017-10-12 20:05:52.895 [main] DBSchemaLoader.shutdownDerby()
2017-10-12 20:05:53.129 [main] DBSchemaLoader.withConnection()
```

```
=====
Failure: Database -'tigasedbx' shutdown.
=====
```

```
Oct 12, 2017 8:05:53 PM tigase.db.util.SchemaManager loadSchemas
INFO: schema upgrade finished!
```

```
=====
Schema installation finished
```

```
Data source: default with uri jdbc:derby:tigasedbx;create=true
Checking connection to database ok
Checking if database exists ok
Loading schema: Tigase XMPP Server (Core), version: 8.0.0 ok
Loading schema: Tigase Message Archiving Component, version: 1.3.0 ok
Loading schema: Tigase MUC Component, version: 2.5.0 ok
Loading schema: Tigase PubSub Component, version: 3.3.0 ok
Adding XMPP admin accounts ok
Post installation action ok
```

```
Example etc/config.tdsl configuration file:
```

```
-'config-type' = -'default'
```

```
debug = [ -'server' -]
-'virtual-hosts' = [ -'ubuntu' -]
dataSource () {
    default () {
        uri = -'jdbc:derby:tigasedbx;create=true'
    }
}
-}
amp () {}
bosh () {}
c2s () {}
eventbus () {}
http () {}
-'message-archive' () {}
monitor () {}
muc () {}
pubsub () {}
s2s () {}
ws2s () {}
=====
```

Note at the end, the script will output a recommended example file. You may use this in conjunction with your written config file, but some settings may not be set using this configuration. Again, it is only an **EXAMPLE**.

## Start the Server

You can start the server using the `tigase` file found in the `scripts` sub-directory of Tigase server base directory. There, select the type of linux you have, `debian`, `gentoo`, `mendriva` or `redhat`. In the root server directory type the following command:

```
./scripts/{OS}/init.d/tigase start etc/tigase.conf
```

Where `{OS}` is your \*nix operating system.

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
```

The first 2 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 an XMPP client of your choice with the administrator account you setup earlier.

## Windows Installation

Tigase XMPP Server can also work on Microsoft Windows systems and servers, although some slight modifications may be necessary to get things ready to run.

Although you may wish to use command line, take note that commands entered in shell may require quotations in some cases.

Make sure that you have Java JDK v8 installed on your system prior to installing Tigase. It will also help to fully setup whatever database software you will be using as well.

### Step 1: Initial Setup

Download the Tigase XMPP Server archive from our repository [<https://projects.tigase.org/projects/tigase-server/files>] and extract it to a directory of your choice.

Once that is completed, enter the directory `win-stuff` and run the `setup.bat` program. This program when run, will extract the necessary files to appropriate places on your computer. The bat file should look like the following:

```
copy -"tigase.ico" -"..\\"
copy -"wrapper\wrapper.jar" -"..\jars"
copy -"wrapper\wrapper.dll" -"..\jars"
copy -"wrapper\wrapper.exe" -"..\\"
copy -"wrapper\wrapper.conf" -"..\\"
copy -"wrapper\wrapper-community-license-1.2.txt" -"..\\"
copy -"scripts\*.*)" -"..\\"
```

### Step 2: Starting Server

To start the server you may use a command prompt from the installation directory

```
java --cp -"jars/*" tigase.server.XMPPServer
```

#### Note

this may freeze the command window, and will only display output from Tigase.

Or you may run `wrapper.exe` or `tigase.bat` from the GUI.

### 2A: Installing as a service

The cleanest way to operate Tigase in a Windows environment is to install Tigase as a Service by running the `InstallTigaseService.bat` program. This will install Tigase as a system service, and now the server can be controlled from the `services.msc` panel. This allows for stopping, starting, and pausing of Tigase XMPP Server and allowing for graceful shutdowns.

For a basic installation, MySQL is recommended over Derby DB. For that purpose, we have included a basic installation guide for MySQL on Windows systems here:

## 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 against the screenshots in the guide below.

In Welcome window just press 'Next'.(pic.1)



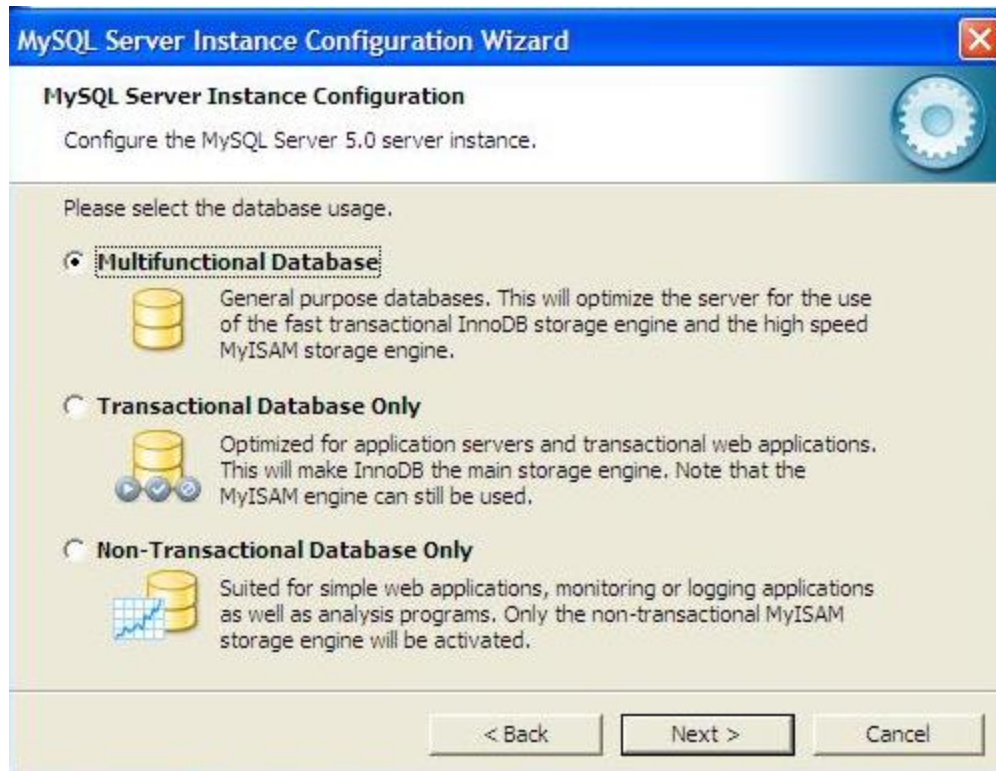
In the next window select option: 'Detailed Configuration' and press 'Next' (pic. 2)



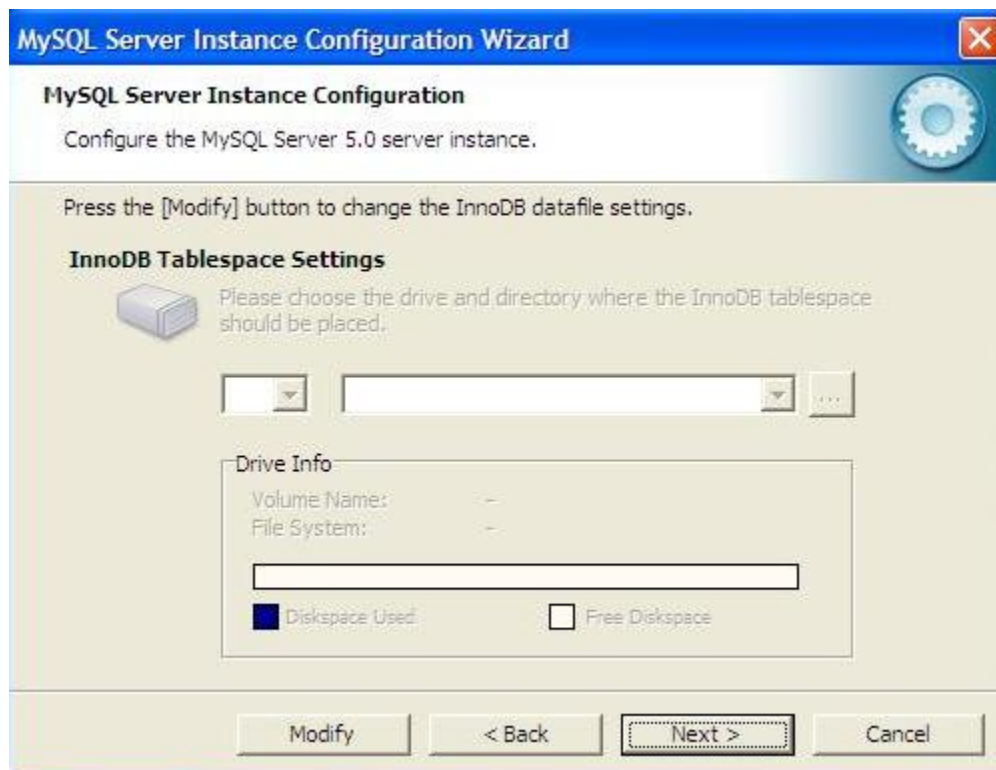
On the next screen select option: 'Server Machine' and press 'Next' (pic. 3)



On the forth windows leave the default "Multi-functional Database" and press 'Next' (pic. 4)

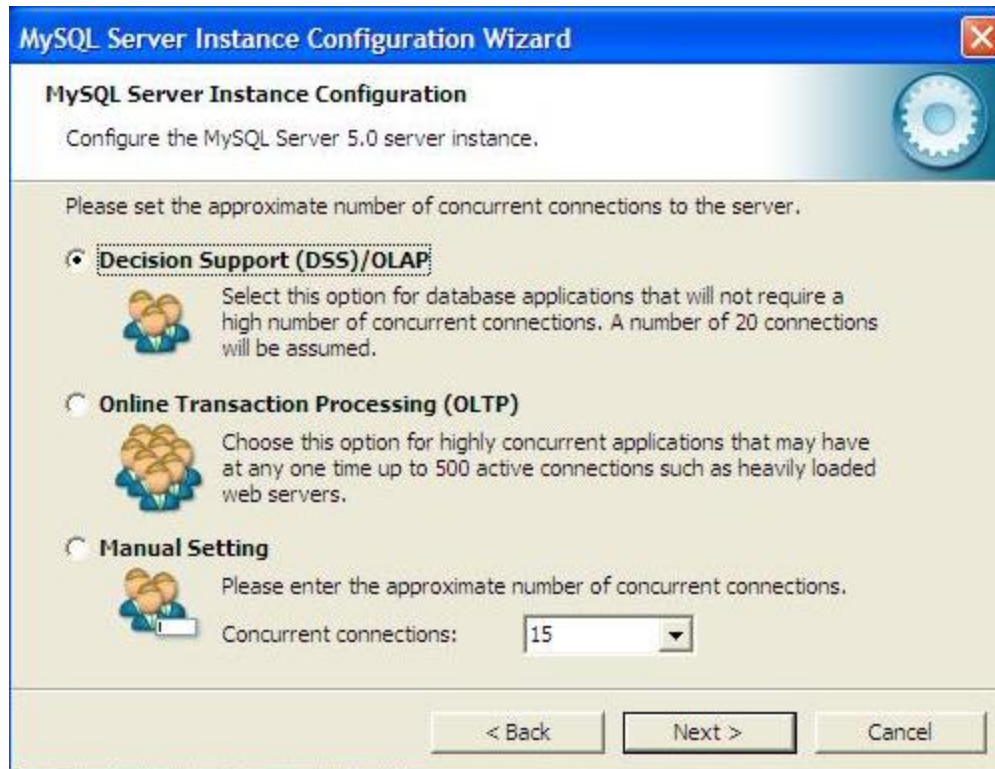


On the step number five just press 'Next' using defaults. (pic. 5)

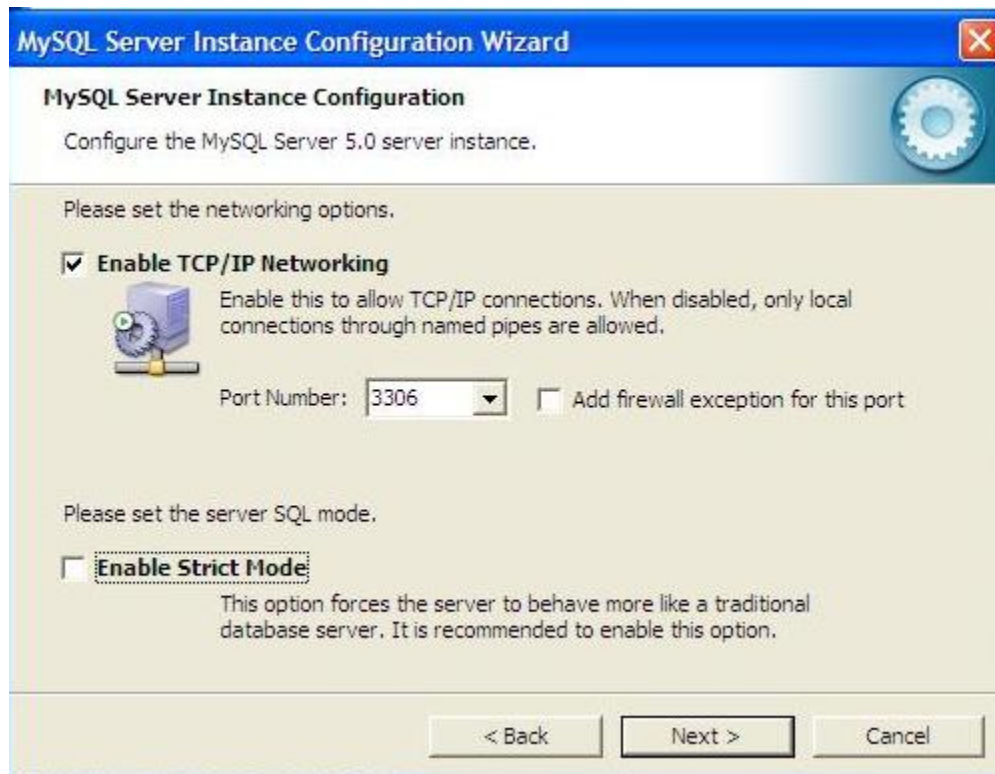


Again, on window 6 select the default - 'Decision Support (DSS)/OLAP' and press 'Next' (pic.6)

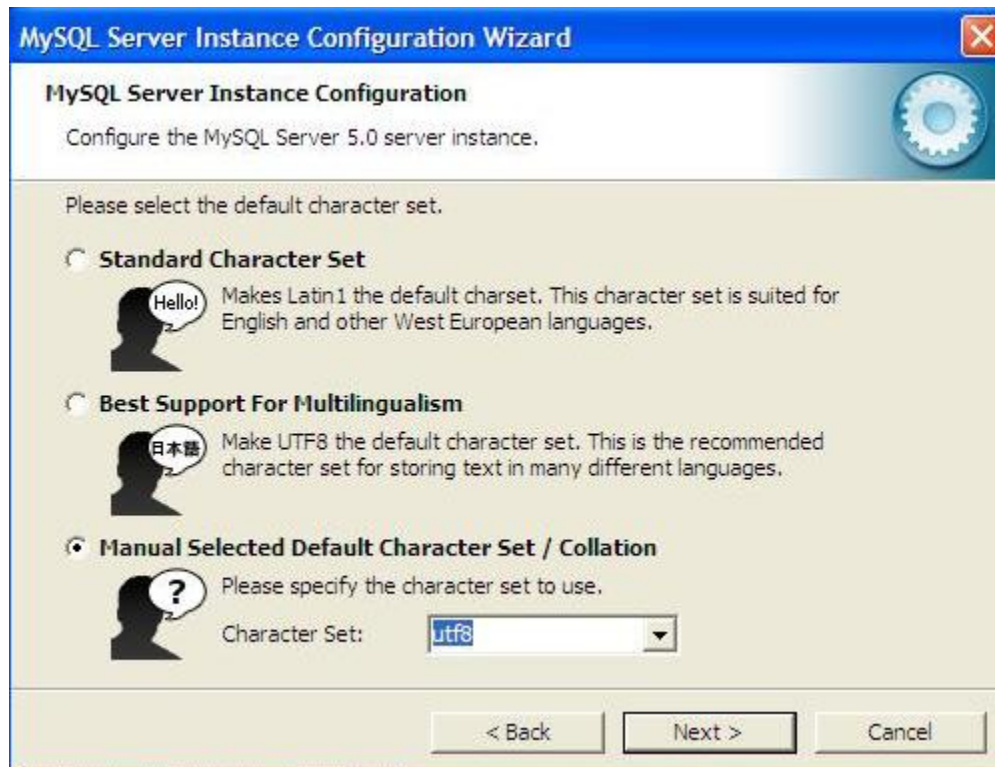




Make sure you switch OFF the 'Strict mode' and and press 'Next' (pic. 7)



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)



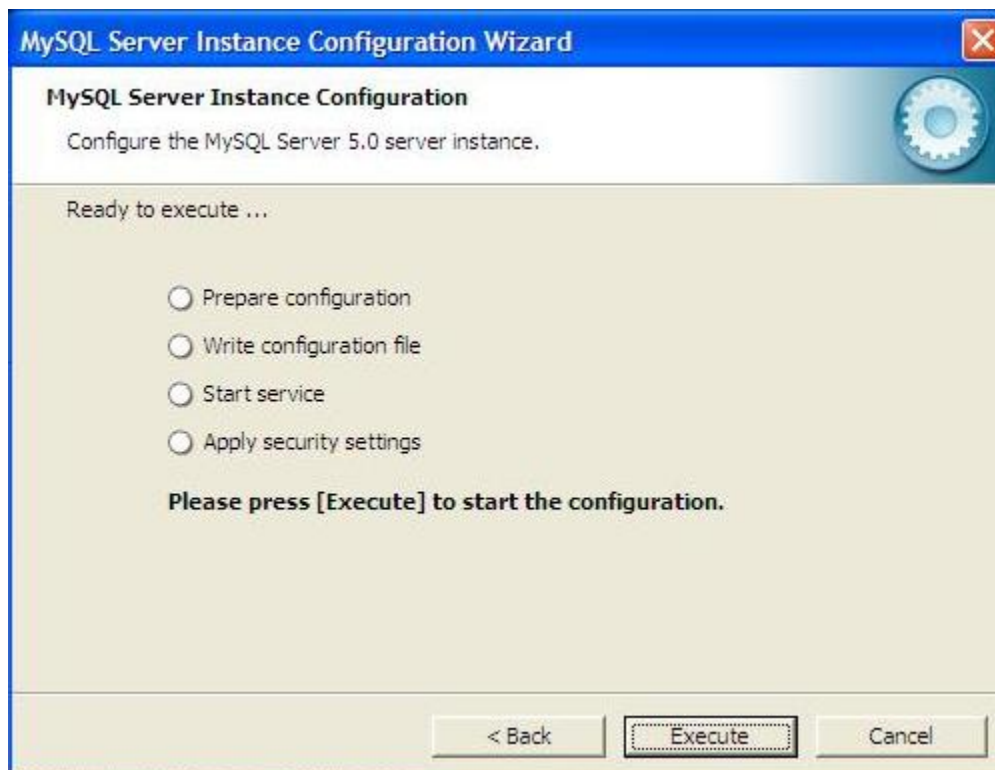


On this window just enter the database super user password and make sure you remember it. When ready press 'Next' (pic. 10)



The screenshot shows the 'MySQL Server Instance Configuration Wizard' window. The title bar is blue with the text 'MySQL Server Instance Configuration Wizard' and a close button. The main window has a blue header with a gear icon. Below the header, the text 'MySQL Server Instance Configuration' is followed by 'Configure the MySQL Server 5.0 server instance.' The main area is light beige and contains the instruction 'Please set the security options.' There are two main sections: 'Modify Security Settings' which is checked, and 'Create An Anonymous Account' which is unchecked. The 'Modify Security Settings' section includes a 'root' user icon, three password input fields labeled 'Current root password:', 'New root password:', and 'Confirm:', and an unchecked checkbox for 'Enable root access from remote machines'. The 'Create An Anonymous Account' section includes a question mark icon and a warning text: 'This option will create an anonymous account on this server. Please note that this can lead to an insecure system.' At the bottom, there are three buttons: '< Back', 'Next >', and 'Cancel'.

This is the last screen. Press 'Execute' to save the configuration parameters. (pic. 11)



The screenshot shows the 'MySQL Server Instance Configuration Wizard' window at the 'Ready to execute' step. The title bar and header are the same as in the previous screen. The main area is light beige and contains the text 'Ready to execute ...'. Below this, there are four radio button options: 'Prepare configuration', 'Write configuration file', 'Start service', and 'Apply security settings'. A bold instruction reads: 'Please press [Execute] to start the configuration.' At the bottom, there are three buttons: '< Back', 'Execute', and 'Cancel'.

When the configuration is saved you can repeat all the steps and change settings at any time by running:  
**START ⇒ Programs ⇒ MYSQL# MYSQL server machine# MySQL Server Instance Config Wizard**

Now we have to setup Tigase database. 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'  
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;
```

When the system is up and running you can connect with any XMPP client (Psi for example) to your server to see if it is working.

## Tigase Server Network Instructions

One you have installed Tigase XMPP Server on a machine, you're going to want to use it. If you are just using for local communications on a network behind a router, you're all set. Enjoy and use!

However, if you want to have people from other computers outside your network connect to your server, you're going to have to go through a few more steps to show your server out to the public.

### Note

This guide is merely a recommendation of how to get a local server to be open to incoming communications. Any time you open ports, or take other security measures you risk compromising your network security. These are only recommendations, and may not be appropriate for all installations. Please consult your IT Security expert for securing your own installation.

XMPP, being a decentralized communication method, relies on proper DNS records to figure out where and how an XMPP server is setup. Operating an XMPP Server will require you to properly setup DNS routing so not only can clients connect to you, but if you decide to run a federated server and enable server to server communication, you will need to do the same. If you already have a DNS server already, you should have little issue adding these records. If you do not have a DNS setup pointing to your server, you may use a free dynamic name service such as dynu.com.

## A Records

You will not be able to use an IP Address or a CNAME record to setup an XMPP Server. While it's not required, an A record can provide some other benefits such serving as a backup in case the SRV record is not configured right.

## SRV Records

You will need to set SRV records both for client-to-server (c2s) communication and, if you plan to use it, server to server (s2s) communication. We recommend both records are entered for every server as some resources or clients will check for both records. For this example we will use `tigase.org` as our domain, and `xmpp` as the xmpp server subdomain.

SRV records have the following form:

```
_service._protocol.name. TTL class SRV Priority weight port target.
```

The key is as follows:

- **service:** is the symbolic name of the desired service, in this case it would be *xmpp-client* or *xmpp-server*.
- **protocol:** is the transport protocol, either TCP or UDP, XMPP traffic will take place over *TCP*.
- **name:** the domain name where the server resides, in this case *tigase.org*.
- **TTL:** a numeric value for DNS time to live in milliseconds, by default use *86400*.
- **class:** DNS class field, this is always *IN*.
- **priority:** the priority of the target host with lower numbers being higher priority. Since we are not setting up multiple SRV records, we can use *0*.
- **weight:** the relative weight for records with the same priority. We can use *5*.
- **port:** the specific TCP or UDP port where the service can be found. In this case it will be *5222* or *5269*.
- **target:** the hostname of the machine providing the service, here we will use *xmpp.tigase.org*.

For our example server, the SRV records will then look like this:

```
_xmpp-client._TCP.tigase.org 86400 IN SRV 0 5 5222 xmpp.tigase.org
_xmpp-server._TCP.tigase.org 86400 IN SRV 0 5 5269 xmpp.tigase.org
```

## Tigase and Vhosts

If you are running multiple vhosts or subdomains that you wish to separate, you will need another record. In this case an A record will be all you need if you are using default ports. If you are using custom ports, you will need to have a new SRV record for each subdomain.

## Hosting VIA Tigase.me

If you don't want to do all the hosting yourself, you can still have an XMPP service running in 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.

Although SRV records are required by the XMPP specifications 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 x.x.x.x
```

This displays x.x.x.x IP address associated to your DNS domain 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.
```

This displays `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 and expand it using the DNS name rather than IP address.

If you want to have MUC and PubSub available under your domain as subdomains, you have to setup DNS for your `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. We recommend checking with the Add and Manage Domains section of the documentation on setting that up. 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.

## Checking setup

If you have a cell phone on a separate network with an XMPP client, you can now try to login to test the server. If that is not handy, you can use an online tool to check proper DNS records such as kingant's: [https://kingant.net/check\\_xmpp\\_dns/](https://kingant.net/check_xmpp_dns/) and it will tell you if anything is missing.

## Ports description

Once your server is setup, you may need to open at least two ports. By default XMPP communication happens on ports 5222/5269, to which point SRV records. Other ports used by the server are:

- 3478 (TURN or STUN, plain socket, TCP and UDP)
- 5349 (TURN or STUN, over TLS, TCP and UDP)
- 5222 (default XMPP socket port)
- 5223 (legacy XMPP socket port)

- 5269 (default s2s port, i.e.: federation support)
- 5277 (component protocol port, e.g.: for external components)
- 5280 (default BOSH port)
- 5290 (default WebSocket port)
- 8080 (HTTP API component port)
- 9050 (JMX Monitoring)

If for any reason you can't use default ports and have to change them it's possible to point SRV records those ports. Please keep in mind, that you have to open those ports for incoming connections in your firewall. In case you are using `iptables` you can use following command to include those ports in your rules:

```
iptables --A INPUT --p tcp --m tcp ---dport 5222 --j ACCEPT
iptables --A INPUT --p tcp --m tcp ---dport 5223 --j ACCEPT
iptables --A INPUT --p tcp --m tcp ---dport 5269 --j ACCEPT
iptables --A INPUT --p tcp --m tcp ---dport 5277 --j ACCEPT
iptables --A INPUT --p tcp --m tcp ---dport 5280 --j ACCEPT
iptables --A INPUT --p tcp --m tcp ---dport 5290 --j ACCEPT
iptables --A INPUT --p tcp --m tcp ---dport 8080 --j ACCEPT
iptables --A INPUT --p tcp --m tcp ---dport 9050 --j ACCEPT
```

Both ports should be setup to use TCP only. If for any reason you want to make service available for different ports you can:

1. change ports in Tigase configuration and update DNS SRV records;
2. forward those ports to default Tigase ports (this is especially useful under \*nix operating system if you want to utilize ports lower than 1024 while running, as recommended, Tigase service from user account - there is a limitation and user accounts can bind to ports lower than 1024), for example using `iptables` rules (in following example we are making available Tigase SSL websocket port available under port 443, which is usually opened in corporate firewalls):

```
iptables --t nat --A PREROUTING --p tcp ---dport 443 --j REDIRECT ---to-ports 52
```

## Tigase Script Selection

As mentioned in each of the quick start sections, each distribution of Tigase XMPP server comes with a number of scripts that are customized for different versions of Linux.

**Table 6.1. init.d chart**

Operating system	init.d file path	Types of Operating Systems
Systemd	tigase-server/scripts/systemd/*	Systemd-based distributions
Debian	tigase-server/scripts/debian/tigase.init.d	Knoppix, Ubuntu (before v15.04), Raspbian or Duvian
Gentoo	tigase-server/scripts/gentoo/init.d/tigase	CoreOS (before v94.0.0), Tin Hat Linux or other *too based systems

Operating system	init.d file path	Types of Operating Systems
Mandriva	<code>tigase-server/scripts/mandriva/init.d/tigase</code>	Specific init.d file for Mandriva Linux
Redhat	<code>tigase-server/scripts/redhat/init.d/tigase</code>	RedHat (before v7.0) and other RPM based linux derivatives like CentOS (before v.7.14), openSUSE (before v12.2)

## Note

If your operating system is a systemd-based linux distribution, we recommend to use systemd service scripts. It may be possible to use (in this case legacy) `init.d` startup files as before, but usage of systemd startup scripts will allow better control of the startup process and will even allow for automatic restart of the Tigase XMPP Server in the case of JVM crash.

## Configuration: For Linux Distributions using systemd

To set up Tigase XMPP Server as a system service it is required to copy `tigase-server.service` file to `/etc/systemd/system/` directory

```
sudo cp $SCRIPT_FILE_PATH/tigase-server.service /etc/systemd/system/
```

This file contains following parameters which may need to be adjusted:

- **User** - Specifies the user that will run the program. This should be a user with SU permissions.
- **WorkingDirectory** - Specifies installation directory (*default: `/home/tigase/tigase-server`*)
- **ExecStart** - Specifies startup command (*default: `runs scripts/tigase.sh start etc/tigase.conf` in the Tigase installation directory*)
- **ExecStop** - Specifies shutdown command (*default: `runs scripts/tigase.sh stop etc/tigase.conf` in the Tigase installation directory*)
- **PIDFile** - Specifies location of the PID file (*default: `logs/tigase.pid` file in the Tigase installation directory*)

It is also required to copy options file `tigase-server` to `/etc/default/` directory

```
sudo cp $SCRIPT_FILE_PATH/tigase-server /etc/default/
```

With those files in place you need to reload `systemctl` daemon

```
sudo systemctl daemon-reload
```

## Note

If you are upgrading from the previous version of the Tigase XMPP Server which was not running as the systemd system service it is required to uninstall old service and remove old service files.

## Configuration: For All Linux Distributions

Once you've located the appropriate distribution scripts (please take a look at the table above), copy it to your system's `init.d` folder (usually it's `/etc/init.d/`):

```
sudo cp $SCRIPT_FILE_PATH -/etc/init.d/tigase
```

You may also need to make it executable:

```
sudo chmod +x -/etc/init.d/tigase
```

It is recommended that you open the script files or configuration files as some have some parameters that you will need to specify.

## Gentoo

The conf.d script must contain the following parameters:

```
TIGASE_HOME="/home/tigase/tigase-server"  
TIGASE_USER=tigase  
TIGASE_CONF="etc/tigase.conf"
```

The following should be configured:

- TIGASE\_HOME - Specifies the Tigase Server installation directory.
- TIGASE\_USER - Specifies the user that will run the program. This should be a user with SU permissions.
- TIGASE\_CONF - The location of tigase.conf file, relative to the TIGASE\_HOME directory.

## Mandriva

Mandriva has a single init.d file, however it should be configured:

```
...  
export JAVA_HOME=/usr/java/jdk1.8.0  
export TIGASE_DIR=/opt/tigase/server/  
tigase=$TIGASE_DIR/scripts/tigase.sh  
prog=tigase  
config=$TIGASE_DIR/etc/tigase.conf  
...
```

The following should be configured:

- JAVA\_HOME - The location of your JDK Installation.
- TIGASE\_DIR - Tigase Server installation directory.
- tigase - The location of your tigase.sh script. This should not need adjusting if you maintain the default file structure.
- config - The location of your tigase.conf file. This should not need adjusting if you maintain the default file structure.

pid file will be stored in /var/run/ser.pid

## Redhat

Similar to Mandriva, you will need to configure the init.d file:

```
...
```

```
JAVA_HOME=/usr/lib/jvm/java/
```

```
USERNAME=tigase
```

```
USERGROUP=tigase
```

```
NAME=tigase
```

```
DESC="Tigase XMPP server"
```

```
TIGASE_HOME=/home/tigase/tigase-server
```

```
TIGASE_LIB=${TIGASE_HOME}/jars
```

```
TIGASE_CONFIG=/etc/tigase.conf
```

```
TIGASE_OPTIONS=
```

```
TIGASE_PARAMS=
```

```
PIDFILE=
```

```
TIGASE_CONSOLE_LOG=
```

```
...
```

- USERNAME - Username running Tigase, should have su permissions.
- USERGROUP - The usergroup of the username.
- NAME - OS name for Tigase program.
- DESC - Optional description.
- TIGASE\_HOME - The location of your Tigase Server installation directory.
- TIGASE\_LIB - The location of your Tigase Jars folder, you should not need to adjust this if you set TIGASE\_HOME properly, and maintain the default file structure.
- TIGASE\_CONFIG - The location of your tigase.conf file relative to TIGASE\_HOME
- TIGASE\_OPTIONS - Legacy options for Tigase, most are now handled in config.tdsl or tigase.conf.
- TIGASE\_PARAMS - Parameters passed to command line when launching Tigase.
- PIDFILE - Location of Tigase PID file if you wish to use custom directory. Default will be located in /logs or /var/temp directory.
- TIGASE\_CONSOLE\_LOG - Location of Tigase Server console log file if you wish to use a custom directory. Default will be located in /logs directory, failing that /dev/null.

After you've copied the script, in order to install sysinit script you have to add it to the configuration:

```
/sbin/chkconfig ---add tigase
```

Service can be enabled or disabled service with:

```
/sbin/chkconfig tigase <on|off|reset>
```

## Debian

As with other distributions you should copy init.d script to the correct location. Afterwards it should be edited and correct values for variables need to be set:

```
...
```



```
USERNAME=tigase
USERGROUP=tigase
NAME=tigase
DESC="Tigase XMPP server"

TIGASE_HOME=/usr/share/tigase
TIGASE_CONFIG=/etc/tigase/tigase.config
TIGASE_OPTIONS=
TIGASE_PARAMS=

PIDFILE=
TIGASE_CONSOLE_LOG=
...
```

- USERNAME - Username running Tigase, should have su permissions.
- USERGROUP - The usergroup of the username.
- NAME - OS name for Tigase program.
- DESC - Optional description.
- TIGASE\_HOME - The location of your Tigase Server installation directory.
- TIGASE\_CONFIG - The location of your tigase-server.xml file relative (old configuration format)
- TIGASE\_OPTIONS - command line arguments passed to Tigase server (which may include path to `init.properties` (if correct `tigase.conf` configuration will be found then it will translate to `TIGASE_OPTIONS="--property-file etc/config.tdsl "`
- TIGASE\_PARAMS - Parameters passed to command line when launching Tigase.
- PIDFILE - Location of Tigase PID file if you wish to use custom directory. Default will be located in `/var/run/tigase/tigase.pid` or under (in this case relative to `tigase home directory`)`logs/tigase.pid`.
- TIGASE\_CONSOLE\_LOG - Location of Tigase Server console log file if you wish to use a custom directory. Default will be located in `/logs` directory, failing that `/dev/null`.

Afterwards we need to install service in the system with following command:

```
update-rc.d tigase defaults
```

## Running Tigase as a system service

There are a number of benefits to running Tigase as a service, one of which is to ensure that the program will run even in the event of a power outage or accidental server restart, Tigase will always be up and running.

### For systemd-based linux distributions

Once installation is complete you may start Tigase as a typical systemd service using following command:

```
sudo systemctl start tigase-server
```

To stop it, you may run following command:

```
sudo systemctl stop tigase-server
```

It is also possible to enable service, to make it start during startup of the operating system:

```
sudo systemctl enable tigase-server
```

## For other linux distributions

Once installation is complete, you should be able to start Tigase using the following command:

```
service tigase start
```

Tigase should begin running in the background. Since Tigase is now installed as a service, it can be controlled with any of the service commands, such as:

- `service tigase stop`
- `service tigase restart`

## Shutting Down Tigase

Although Tigase XMPP Server can be terminated by ending the process, it is preferred and recommended to use it's own shutdown scripts instead. Not only does this allow for a proper purge of Tigase from the system, but allows for all shutdown functions to operate, such as amending logs and completing statistics. To trigger a shutdown of Tigase server, the following command can be used from the tigase directory:

```
./scripts/tigase.sh stop
```

You may specify the config file if you want, but it will make no differences

This will:

- Begin shutdown thread
- Stop accepting new connections
- Close all current connections
- Collect runtime statistics
- Write statistics to log
- Dump full stacktrace to a file
- Run GC and clear from memory

## Shutdown statistics

Upon shutdown, statistics for the server's runtime will be appended to the log file. For a description of the statistics and what they mean, refer to the Statistics Description portion of the documentation.

## Shutdown StackTrace Dump

To aid with troubleshooting purposes, the full stacktrace will be dumped to a separate file located at `$serverdir/logs/threads-dump.log.#` Stacktrace logs will follow the same log file numbering scheme described in Log file description.

This feature is enabled by default, however you may disable this by adding the following to your `config.tdsl` file:

```
'shutdown-thread-dump' = false
```

## Shutting Down Cluster Nodes

Starting with v8.0.0 you can now shut down individual cluster nodes without shutting down the whole server. This command will use the *SeeOtherHost* strategy to direct traffic to other nodes and update the cluster map to gracefully shut down the single node

Shutting down individual nodes can be done VIA Ad-hoc command and fill out the response forms. The command is available from message-router as `http://jabber.org/protocol/admin#shutdown`.

## Upgrading to v8.0.0 from v7.1.0

There have been a number of changes to the user and auth databases since v7.1.0. As a result, if you are upgrading from older versions, you will need to follow this guide.

### Note

We recommend installing Tigase XMPP Server 8.0.0 in a separate directory.

## Backup your data

As with any migration it is highly recommended that you backup your repository before conducting any upgrade operations.

For MySQL databases:

```
mysqldump [dbname] ---routines --u [username] --p [password] > [filename].sql
```

## Setup Tigase XMPP Server 8.0.0

After downloading Tigase XMPP Server 8.0.0 from our website, or using `wget`, extract the files to a separate directory.

Copy the `tigase.conf` and `init.properties` files from the old directory to v8.0.0 directory.

```
cd tigase-server-8.0.0
cp ../tigase-server/etc/tigase.conf etc/
cp ../tigase-server/etc/init.properties etc/
```

Import the database.

```
mysql --h [host address] [dbname] --u [username] --p [password] < [filename].sql
mysql --h 198.27.120.213 tigase_tpub --u USERNAME --p <../tpub.2017-05-30.sql
Enter password:
```

## Upgrade configuration file

Tigase XMPP Server has a utility that can be called using `upgrade-config` that will update your old `init.properties` file and create a new file using DSL.

```
./scripts/tigase.sh upgrade-config etc/tigase.conf
```

When everything is ready it will printout following information

```
=====
Configuration file etc/init.properties was converted to DSL format.
Previous version of a configuration file was saved at etc/init.properties.old
=====
```

## Connect new database

Edit your new `config.tdsl` file to connect to the new database you created during the import step.

```
dataSource {
  default () {
    uri = -'jdbc:mysql://localhost/tigase_tpub?user=tigase_user&password=mypas
  -}
}
userRepository {
  default () {}
}
authRepository {
  default () {}
}
```

## Upgrade Database schema

Upgrading database schemas is now possible using the `upgrade-schema` option. Do this now.

```
./scripts/tigase.sh upgrade-schema etc/tigase.conf
```

### Warning

Your database schema **MUST** be v8 or conversion will not occur properly!

You will be asked the following prompts:

```
Database root account username used to create tigase user and database at 198.27.1
```

```
Database root account password used to create tigase user and database at 198.27.1
```

Upon success, you should see the following:

```
=====
Schema upgrade finished

Data source: default with uri
jdbc:mysql://HOST/DATABASE?user=USERNAME&password=PASSWORD
Checking connection to database ok
Checking if database exists      ok
Loading schema: Tigase XMPP Server (Core), version: 8.0.0      ok
Loading schema: Tigase Message Archiving Component, version: 1.3.0  ok
Loading schema: Tigase MUC Component, version: 2.5.0      ok
Loading schema: Tigase PubSub Component, version: 3.3.0 ok
```

```
Adding XMPP admin accounts      warning
      Message: Error: No admin users entered
Post installation action         ok
```

=====

Start Tigase!

## Help?

Both upgrade commands also have a build in help function, they can be called if needed from the command line. You can also run these commands for help.

```
scripts/tigase.sh upgrade-config etc/tigase.conf --help
scripts/tigase.sh upgrade-schema etc/tigase.conf --help
```

## Upgrade/Restore with a script [experimental!]

To make upgrade process easier it's possible to utilize `tigase-upgrade.sh` [`files/tigase-upgrade.sh`] \*nix shell script. It permits upgrading to new version (supports downloading version from provided URL).

It's usage is as follows:

```
./tigase-upgrade.sh {upgrade|rollback} install_package install_directory [tar|dir]
```

Where: \* `{upgrade|rollback}` - defines whether to perform upgrade or rollback to previous version  
\* `install_package` - package to which perform upgrade (can be URL) in case of upgrade or backed-up installation (from which we want to restore) in case of rollback \* `install_directory` - destination directory (both in upgrade and rollback); can be symlink in which case it will be preserved with upgraded/restored path as target \* `[tar|dir]` - (optional) type of backup (either simply copy directory or also archive it using `tar` command); by default `dir` is used.

To upgrade installation to version `tigase-server-8.0.0-SNAPSHOT-b5285-dist-max.tar.gz` execute the script as follows:

```
$ ./tigase-upgrade.sh upgrade tigase-server-8.0.0-SNAPSHOT-b5285-dist-max.tar.gz
```

To rollback from `tigase-server-8.0.0-SNAPSHOT-b5264_backup-18-11-05_1712` backup execute script as follows:

```
bash --x ./tigase-upgrade.sh rollback tigase-server-8.0.0-SNAPSHOT-b5264_backup-1
```

---

# Chapter 7. Configuration

When the user tries to setup the client for the first time he comes across 2 configuration files: `tigase.conf` and `config.tdsl` in the `/etc` folder. 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. `config.tdsl` file is a simple text file with server parameters in form: **key = value**. When the XML configuration file is missing the Tigase server reads `config.tdsl` file and uses parameters found there as defaults for generation of the XML file. Therefore if you change the `config.tdsl` file you normally have to stop the server, remove the XML file and start the server again. All the settings from the `config.tdsl` 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.
2. `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.

## DSL file format

In previous Tigase XMPP Server releases configuration was stored in properties based configuration file. From Tigase XMPP Server 8.0.0 release it will be required to use new DSL based configuration file format. This file format was inspired by Groovy language syntax and new core feature of Tigase XMPP Server - Tigase Kernel Framework.

## Why new format?

In properties configuration format each line contained key and value with optional definition of type of stored value:

```
c2s/ports[i]=5222,5223
```

where `c2s/ports` was name of property, `[i]` defined that type of value is array of integers, and `5222,5223` was comma separated list of values.

This format worked but in fact `c2s/ports` was not name of property you configured but key which was later split on `/` char to parts which defined by names path to property which name was in last part. From that you can see that it was domain based setting of properties.

Except from this multi-part keys we also used properties starting with `--` which were global properties accessible for every part of application, i.e.: to add new component and set some properties you needed to write:

```
--comp-name-1=pubsub
--comp-class-1=tigase.pubsub.PubSubComponent
pubsub/test[B]=true
pubsub/pubsub-repo-url="jdbc:XXXX:XXXX/db_name"
```

This lead to mistakes like duplicated definition of name and class for same number of component or redefined property value in other place of a configuration file - especially in cases where configuration was big.

In this configuration structure it was hard to tell where is configuration for particular component or what databases this installation uses. This could be defined all over the file.

In this release we are introducing Tigase Kernel Framework, which allows to configure beans in configuration file and even define usage of new beans loaded from external jars which can modify behavior of Tigase components. This would make configuration file even more complex, difficult and less readable.

## What is DSL?

DSL stands for domain-specific language - in this case language created for storage of configuration.

Now we use domain based configuration which means that our configuration file is not a flat key=value storage but it defines objects, it's properties and assigned values.

To illustrate it better let's start with a simple example. In properties file in order to configure PubSub component named pubsub you would use following properties:

```
--comp-name-1=pubsub
--comp-class-1=tigase.pubsub.PubSubComponent
pubsub/test[B]=true
```

In DSL based configuration this would be replaced by following block

```
pubsub (class: tigase.pubsub.PubSubComponent) {
    # comment
    test = true
}
```

in which we define bean with name pubsub and set it's class inside () block to tigase.pubsub.PubSubComponent. We also use block between {} chars to define properties which are related to bean. Which means this properties will be passed only to this instance of Tigase PubSub Component, same as it was before where we needed to add prefix. Entries after # are comments, to pass # you need to wrap whole part containing it in ', ie. 'test#242'

### Warning

If a string value assigned to a property contains any char from a following list = : , [ ] # + - \* / it needs to be wrapped in a ' '.

## Why DSL?

DSL configuration format provides a number of advantages over the old system of configuration. . All configurations for components are related in a single block, so they are not spread out over several different lines. . No need for long property names, no longer have to invoke a long string of settings for multiple values. . Support is provided for environment variables. . No longer need to escape certain characters, making settings far more readable at a glance. . Values may be set using basic calculations, such as 100 \* 200 \* 2 rather than 40000. . Parameter type values are no longer necessary, no more [i], [S], [B] etc.. . Comma separated values can now be simplified lists with separate entries being able to be in multiple lines.

Although the format may seem more complex, looking like a section of java code, the formatting is consistent and can be far more readable. After some experience with DSL format, you'll find it's far more intuitive and user friendly than it may appear. Of course if there's any real confusion, Tigase can automatically convert old style properties files to the DSL format using the following command:

```
./scripts/tigase.sh upgrade-config etc/tigase.conf
```

## Setting property

To set property you just write property name followed by = and value to set. This is always done in context of bean which configuration property you want to set.

```
test=true
```

It is also possible to set property in main context by placing property outside of any context. This sets property which value is available to access by any bean.

## Setting global property

Like in properties file it is still possible to use property names starting with -- without any context or any other properties at global scope. Format is the same as in case of setting property but they are defined without scope (in global scope). These properties are global and accessible by any bean but also set as system property in JVM.

## Defining bean

You can configure bean by using following format:

```
beanName (class: className, active: activeValue, exportable: exportableValue) {  
    # scope of bean properties  
}
```

where beanName is name under which you want to configure bean. beanName must be wrapped in ' ', if beanName contains characters like = : , [ ] # + - \* / and is recommended, if beanName is numeric only.

Inside block between ( and ) you can define:

- class which will be used as a bean, in example above we set class as className. *(default: if you try to configure bean under name which has default class assigned with it in Tigase framework then this assigned class will be used. In other case you need to pass name of class to use as a bean)*
- active (boolean) whether you want the bean to be active or not (beans with active set to false are not loaded). *(default: true)*
- exportable (boolean) defines if this bean should be exported and available for use for beans in inner scopes. This is advanced option in most cases it is recommended to omit this field in configuration. *(default: false)*

Spaces between beanName and block between ( ) is optional as well as space between block ( ) and block { }. It is recommended that properties of bean would be placed in separate lines with indentation and first property will be placed in new line.

## Important

Usage of ( ) block is very important. When this block is used in configuration it automatically sets active property of bean definition for bean for which it is used to true. This is done due to fact that default value of active is true.

If you omit it in configuration, you will set bean configuration but it may remain inactive. In this state bean will not be loaded and as a result will not be used by Tigase XMPP Server.



## Configuring bean

If you know that bean is defined and you do not want to change it's activity or class then you can just pass properties to configure bean in following way:

```
beanName {
    # scope of bean properties
    test = true
}
```

where `beanName` is name of bean to configure and `test` is name of property to set to `true` in this bean.

## Format of values

In properties based configuration file every property was defined as a string and only by defining expected format it was properly converted to expected value. In DSL it is possible to set values in two ways:

as an object

Using this format you set list as a list and integer is set as an integer.

Type	Description
<b>string</b>	Wrap it in ' ', ie. to set <code>test</code> as string you use <code>'test'</code>
<b>integer</b>	Just put value, ie. to set 543 use 543
<b>long</b>	Put value and follow it with L, ie. to set 23645434 as long use 23645434L
<b>float</b>	Put value and follow it with f, ie. to set 231.342 use 231.342f
<b>boolean</b>	To set value just use <code>true</code> or <code>false</code>
<b>list</b>	<p>Lists can be of many types and to make it simple we decided to use as a comma separated list of values in proper format wrapped in [ ].</p> <ul style="list-style-type: none"> <li>• of strings - [ 'alfa', 'beta', 'gamma' ]</li> <li>• of integers - [ 1, 2, 3, 4 ]</li> </ul> <p>You can write it in multiple lines if you want:</p> <pre>[     -'alfa'     -'beta'     -'gamma' ]</pre>
<b>map</b>	<p>Maps can be written as a block of properties wrapped in { }. This format of map is the same as used for passing configuration to bean properties. Keys and values can be written in separate lines (<i>recommended</i>):</p> <pre>{     test = true     ssl = false     ssl-certificate = -'/test/cert.pem'     another-map = {         key = -'value'     -} }</pre> <p>or in single line (<i>separation with spaces is not required</i>):</p>

Type	Description
	<code>{ test = true, ssl = false, ssl-certificate = -'/test/cert.pem' -}</code>

as a plain string

Very similar to properties based configuration, in fact values are passed in same format and later are converted to correct type by checking type expected by bean. *(Not recommended)*

Type	Description
<b>string</b>	Just put value, ie. to set test use test
<b>integer</b>	Just put value, ie. to set 543 use 543
<b>long</b>	Put value, ie. to set 23645434 as long use 23645434
<b>float</b>	Put value, ie. to set 231.342 use 231.342
<b>boolean</b>	To set value just use true or false
<b>list</b>	List needs to be written as comma separated list of values, ie. test,abc,efg or 1,2,3
<b>map</b>	Not possible

## Using values from System Properties and Environment Variables

Now it is possible to use values of system properties [<https://docs.oracle.com/javase/tutorial/essential/environment/sysprop.html>] and environment variables [<https://docs.oracle.com/javase/tutorial/essential/environment/env.html>] and assign them to bean properties. For this purpose we added functions which can be used in DSL and which will return values of:

system property                      `prop('property-name')`                      or                      `prop('property-name','default value')`

environment variable                      `env('variable-name')`

**Example of setting value of system property and environment variable to bean user.**

```
user {
    name = env('USER')
    home = prop('user.home')
    paths = [ prop('user.home'), prop('user.dir') -]
}
```

### Warning

For properties which accepts lists it is not allowed to set value using variable/property with comma separated values like value1,value2 wrapped in [], ie. `property = [ env('some-variable') ]`. It needs to be set in following way `property = env('some-variable')`

## Computed values

With DSL configuration format we introduce support for computable values for properties. It is now possible to set value which is result of a computation, ie. concatenation of a strings or very simple mathematical expression. We currently support only following mathematical operations:

- add

- subtract
- multiply
- divide

**Example of setting environment variable related path and computed timeout.**

```
bean {  
    # setting path to `some-subdirectory` of user home directory  
    path = prop('user.home') + -'/some-subdirectory/'  
  
    # setting timeout to 5 minutes (setting value in milliseconds)  
    timeout = 5L * 60 * 1000  
    # previously it would need to be configured in following way:  
    # timeout = 300000L  
}
```

**Warning**

For properties which accepts lists it is not allowed to set value using computed values with comma separated values like `value1,value2` wrapped in `[]`, ie. `property = [ env('some-variable') + ',other-value' ]`. It needs to be set in following way `property = env('some-variable') + ',other-value'`.

**Period / Duration values**

Some configuration options allow control of execution of tasks with particular period or within certain duration. DSL file format accepts strings denoting particular amount of time, which follows Java's native structures (see: `Period` [<https://docs.oracle.com/javase/8/docs/api/java/time/Period.html#parse-java.lang.CharSequence->] and `Duration` [<https://docs.oracle.com/javase/8/docs/api/java/time/Duration.html#parse-java.lang.CharSequence->] for detailed explanation).

- Duration formats accepted are based on the ISO-8601 duration format `PnDTnHnMn.nS` with days considered to be exactly 24 hours, for example:
  - `PT20.345S` - 20.345 seconds
  - `PT15M` - 15 minutes (where a minute is 60 seconds)
  - `PT10H` - 10 hours (where an hour is 3600 seconds)
  - `P2D` - 2 days (where a day is 24 hours or 86400 seconds)
  - `P2DT3H4M` - 2 days, 3 hours and 4 minutes
- Period format is based on the ISO-8601 period formats `PnYnMnD` and `PnW`, for example, the following are valid inputs:
  - `P2Y` - 2 years
  - `P3M` - 3 months
  - `P4W` - 4 weeks
  - `P5D` - 5 days

- P1Y2M3D - 1 year, 2 months, 3 days
- P1Y2M3W4D - 1 year, 2 months, 3 weeks, 4 days

## Example configuration file in DSL

```
# Enable cluster mode
--cluster-mode = true
# Enable debugging for server and xmpp.impl
--debug = -'server,xmpp.impl'
# Set list of virtual hosts (old way)
--virt-hosts = -'example.com,test-1.example.com,test-2.example.com'

# Configure list of administrator jids
admins = [ -'admin@zeus', -'http@macbook-pro-andrzej.local' -]
# Set config type
config-type = -'--gen-config-def'

# Configure dataSource bean with database configuration
dataSource {
    # Configure default data source (using default implementation so class is omitted)
    default () {
        uri = -'jdbc:postgresql://127.0.0.1/tigase?user=test&password=test&autoCreateTables=true'
    }

    # Configure data source with name example.com (will be used by domain example.com)
    -'example.com' () {
        uri = -'jdbc:mysql://127.0.0.1/example?user=test&password=test&autoCreateTables=true'
    }
}

# Configure C2S component
c2s {
    # Enable Stream Management bean
    -'urn:xmpp:sm:3' () {}

    # Register tigase.server.xmppclient.SeeOtherHostDualIP as seeOtherHost bean
    seeOtherHost (class: tigase.server.xmppclient.SeeOtherHostDualIP) {}

    # Add additional port 5224 which is SSL port and disable port 5223
    connections () {
        -'5224' () {
            socket = ssl
        }
        -'5223' (active: false) {}
    }
}

# Configure HTTP API component
http {
    # Set list of API keys
    api-keys = [ -'test1234', -'test2356' -]
    rest {
```

```

        # Set value of environment property as a path to look for REST scripts
        rest-scripts-dir = env('TIGASE_REST_SCRIPTS_DIR')
    -}
}

# Register pubsub-2 (class is passed as pubsub-2 name do not have default class as
pubsub-2 (class: tigase.pubsub.cluster.PubSubComponentClustered) {
    # Set configuration bean properties
    pubsubConfig {
        persistentPep = true
    -}
    # Use tigase.pubsub.cluster.ClusteredNodeStrategy as advanced clustering strat
    strategy (class: tigase.pubsub.cluster.ClusteredNodeStrategy) {}
}

# Configure Session Manager
sess-man {
    # Here we enable pep, urn:xmpp:mam:1 processors and disable message-archive-xe
    pep () {}
    -'urn:xmpp:mam:1' () {}
    message-archive-xep-0136 (active: false) {}

    # Define class used as clustering strategy (it is different than default so cl
    strategy (class: tigase.server.cluster.strategy.OnlineUsersCachingStrategy) {}
}

```

## Default configuration

Tigase XMPP Server is packaged with a basic `config.tds1` file that tells the server to start up in setup mode.

```

'config-type' = -'setup'

http () {
    setup () {
        -'admin-user' = -'admin'
        -'admin-password' = -'tigase'
    -}
}

```

This tells Tigase to operate in a setup mode, and tells the http component to allow login with the username and password admin/tigase. With this you can enter the setup process that is covered in this section.

There are other options for config-type: default, session-manager, connection-managers, and component. For more information, visit [Config Type property description](#).

## Startup File for tigase.sh - tigase.conf

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

Config file for startup script simply sets number of environment variables with the 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`.
- `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 initial parameters for your environment here.

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/config.tds1 -"
```

Please note encoding settings. JVM by default uses encoding set in operating system environment. XMPP protocol, however uses UTF-8 for all data processing. So the ENC settings enforces UTF-8 encoding for all operations.

Another significant setting is `\CLASSPATH`. It is intentionally set to an empty string. The **tigase.sh** startup script builds the **CLASSPATH** on it's own from files found in **jars/** and **libs/** directories. It is advised 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 can cause a long startup time and high system loads.

## Linux Settings for High Load Systems

There are a few basic settings you have to adjust for high load systems to make sure the server has enough resources to handle a 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 by executing the command:

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

If you plan to run high load service with large 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 by executing the command:

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

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

You can see current settings by executing the command:

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

You can change this setting by executing the command:

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

## TCP\_keepalive

According to [www.gnu.org/ \[http://www.gnu.org/keepalive.html\]](http://www.gnu.org/keepalive.html) some keepalive settings should be changed to improve reliability - it will enable keep alive functionality (checking if the connection is established and valid) and, by decreasing times and interval - will make detection of broken connections faster.

```
# 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

The 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 the 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 65000
net.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 have 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 the init scripts will respect configured limits.

## JVM settings and recommendations

Tigase configuration file `tigase.conf` (described in more detail in the section called "Startup File for `tigase.sh` - `tigase.conf`") mentioned a couple of environmental variables which are related to the operation of the JVM. In this guide we would like to expound on those configuration options and provide hints for the optimal settings.

Settings included in the `etc/tigase.conf` are as follows:

```
#GC="-XX:+UseBiasedLocking --XX:+UseConcMarkSweepGC --XX:+UseParNewGC --XX:NewRatio=1"
#EX="-XX:+OptimizeStringConcat --XX:+DoEscapeAnalysis --XX:+UseNUMA"
```

```
#GC_DEBUG=" --XX:+PrintTenuringDistribution --XX:+PrintGCDetails --XX:+PrintGCDateAndTime"
```

```
#PRODUCTION_HEAP_SETTINGS=" --Xms5G --Xmx5G -" # heap memory settings must be adjusted
JAVA_OPTIONS="${GC} ${GC_DEBUG} ${EX} ${ENC} ${DRV} ${JMX_REMOTE_IP} --server ${PR
```

And while this file utilizes bash variables, JVM configuration options can be used in the same manner on all operating systems.

The guide will consists of two main parts - memory settings and Garbage Collector tweaks descriptions and hints.

We recommend using `-server` JVM parameter in all cases.

## Heap Sizing

For the non-production deployments (development or staging environments) we recommend using default memory settings of the JVM (which depends on the underlying operating system), which result i automatic memory allocation and, by the rule of thumb - are the safest in such environments.

For the production environments we recommend a fixed size HEAP - both initial and maximum size, which can be set with (respectively) `-Xms` and `-Xmx` JVM flags - ideally to the same value (which should be roughly 95% of the available memory, if Tigase will be the only service on the machine) to avoid allocation and deallocation.

For convenience it's possible to uncomment line with `PRODUCTION_HEAP_SETTINGS` and adjust parameters accordingly.

## GC settings

Let's start with stating that there is no "one to rule them all" - each deployment and use-case is different, however we will try to give a couple of pointers and recommendations proceed with short introduction to GC itself.

XMPP is quite specific in terms of memory allocation - short-lived objects (various types of stanzas) usually exceed number of long-lived objects (user connections and related data). This is important bit of information in the context of how usually JVM HEAP is organized and how Garbage Collector works. On the most basic level Heap is separated into couple of regions:

### Generations

- **Young Generation**, which is further divided in to:
  - **Eden** - the region when the objects are usually allocated when they are created;
  - **Survivor Spaces** - (*to* and *from* - one of which is always empty) - responsible for storing all live object remaining after collecting **Young Generation** (process is repeated several times until objects are finally considered *old enough*);
- **Old Generation** - (*Tenured Space*) - responsible for live objects remaining after running GC on **Survivor Spaces** - those would be *long-lived* objects (usually user connections and associated data);

### Minor, Major and Full GC - optimizing

General thinking suggests that:

- **Minor GC** cleans Young generation;
- **Major GC** cleans Tenured space;
- **Full GC** cleans all heap.

However, while we can certainly state that Minor GC cleans Young generation it's a bit more difficult to differentiate Major and Full GC, especially considering that Major GC can be quite often triggered by Minor GC and some garbage collectors can perform cleaning concurrently. Instead of focusing of distinguishing phases one should pay closer attention to actual operations of Garbage Collector itself - uncommenting the line `GC_DEBUG=" -XX:+PrintTenuringDistribution -XX:+PrintGCDetails -XX:+PrintGCDateStamps -XX:+PrintGCTimeStamps -Xloggc:logs/jvm.log -verbose:gc "` in `etc/tigase.conf` (or adding same properties to the java commandline) and subsequently analyzing the results should prove more helpful. In addition monitoring GC operation using for example VisualVM (with VisualGC plugin) will also be helpful.

### Settings for XMPP

Ideally we should limit both number of GC pauses as well as their duration. After running rather tests following conclusions were made:

- Garbage Collection is the faster the more dead objects occupies given space, therefore on high-traffic installation it's better to have rather large YoungGen resulting in lower promotion of the objects to the OldGen;
- with JVM8 default sizing of Young / Old generation changed, even tho NewRatio is still defaulting to "2" - setting it explicitly to "2" brought back previous sizing;

- Concurrent Mark and Sweep (CMS) enabled (applies to Tenured space only) with explicit configuration of NewRatio set to default value of 2 (i.e. `-XX:+UseConcMarkSweepGC -XX:+UseParNewGC -XX:NewRatio=2`) in general behaves best;
- For small installations (few core CPU, less memory) with low traffic default Parallel collector may be a better solution;
- Using Heap size adjusted to the actual usage is better as the larger the heap the larger are spaces over which collection needs to be performed thus resulting in longer pauses; in case of huge heaps G1 collector may be better solution to avoid longer pauses;

Considering all of the above using following options should be a good starting point toward further optimizing of Garbage Collection:

```
GC="-XX:+UseBiasedLocking      -XX:+UseConcMarkSweepGC      -XX:+UseParNewGC
-XX:+CMSIncrementalMode      -XX:-ReduceInitialCardMarks      -
XX:CMSInitiatingOccupancyFraction=70 -XX:+UseCMSInitiatingOccupancyOnly"
```

## GC settings worth considering

In addition to the general recommendation to use CMS collector, following options (or changes to the options) may be worth considering:

- `-XX:NewRatio=2` - defines the ratio between the young and tenured generation is 1:2. In other words, the combined size of the eden and survivor spaces will be one-third of the total heap size. The parameters NewSize and MaxNewSize bound the young generation size from below and above. Setting these to the same value fixes the young generation, just as setting `-Xms` and `-Xmx` to the same value fixes the total heap size.
- `-XX:CMSInitiatingOccupancyFraction=percent` - sets the percentage of the old generation occupancy (0 to 100) at which to start a CMS collection cycle.
- `-XX:+UseCMSInitiatingOccupancyOnly` - instructs the JVM not to base its decision when to start a CMS cycle on run time statistics but instead it uses the value of `CMSInitiatingOccupancyFraction` for every CMS cycle.
- `-XX:ParallelGCThreads=x` - sets the number of threads used for parallel garbage collection in the young and old generations. The default value depends on the number of CPUs available to the JVM. If the Tigase JMV is the only one running on the installation default value is recommended.
- `-XX:ConcGCThreads=x` - sets the number of threads used for concurrent GC. The default value depends on the number of CPUs available to the JVM. If the Tigase JMV is the only one running on the installation default value is recommended.
- `-XX:+UseBiasedLocking` and `-XX:+DoEscapeAnalysis` - designed to eliminate locking overhead, however their effect on performance is unpredictable therefore testing is required; reduced locking should improve concurrency and, on current multi-core hardware, improve throughput.
- `-XX:+OptimizeStringConcat` - enables the optimization of String concatenation operations. This option is enabled by default.
- `-XX:+UseNUMA` - enables performance optimization of an application on a machine with nonuniform memory architecture (NUMA - most modern computers are based on NUMA architecture) by increasing the application's use of lower latency memory. By default, this option is disabled and no optimization for NUMA is made. The option is only available when the parallel garbage collector is used (`-XX:+UseParallelGC`).

- `-XX:-UseCompressedOops` — disables the use of compressed pointers. By default, this option is enabled, and compressed pointers are used when Java heap sizes are less than 32 GB. When this option is enabled, object references are represented as 32-bit offsets instead of 64-bit pointers, which typically increases performance when running the application with Java heap sizes less than 32 GB. This option works only for 64-bit JVMs.

## What to use with Machine x, y, z?

### Server class machine (non-VM), > 16GB, >= 8 core CPU

For such setup enabling CMS garbage collector is recommended. Depending on the traffic usage and particular use-case adjusting NewRatio may be needed. Adjusting Xms and Xmx sizes for actual available memory is needed (or better yet, for the actual traffic!). Following should be used:

```
GC="-XX:+UseBiasedLocking --XX:+UseConcMarkSweepGC --XX:+UseParNewGC --XX:NewRatio=10"
EX="-XX:+OptimizeStringConcat --XX:+DoEscapeAnalysis --XX:+UseNUMA"
```

```
#GC_DEBUG=" --XX:+PrintTenuringDistribution --XX:+PrintGCDetails --XX:+PrintGCDateAndTime"
```

```
PRODUCTION_HEAP_SETTINGS=" --Xms15G --Xmx15G -" # heap memory settings must be adjusted
JAVA_OPTIONS="{GC} {GC_DEBUG} {EX} {ENC} {DRV} {JMX_REMOTE_IP} --server {PR}"
```

For installation with lot of available memory and intention to utilize it all, using G1GC collector may be a better idea :

```
GC="-XX:+UseG1GC --XX:ConcGCThreads=4 --XX:G1HeapRegionSize=2 --XX:InitiatingHeapOccupancyPercent=10"
EX="-XX:+OptimizeStringConcat --XX:+DoEscapeAnalysis --XX:+UseNUMA"
```

```
#GC_DEBUG=" --XX:+PrintTenuringDistribution --XX:+PrintGCDetails --XX:+PrintGCDateAndTime"
```

```
PRODUCTION_HEAP_SETTINGS=" --Xms60G --Xmx60G -" # heap memory settings must be adjusted
JAVA_OPTIONS="{GC} {GC_DEBUG} {EX} {ENC} {DRV} {JMX_REMOTE_IP} --server {PR}"
```

### VM machine, 8GB of RAM, 4 core CPU equivalent

For such setup enabling CMS garbage collector is also recommended. Depending on the traffic usage and particular use-case adjusting NewRatio may be needed (and configuring NewRatio is a must!). Adjusting Xms and Xmx sizes for actual available memory is needed (or better yet, for the actual traffic!). Following should be used:

```
GC="-XX:+UseBiasedLocking --XX:+UseConcMarkSweepGC --XX:+UseParNewGC --XX:NewRatio=10"
EX="-XX:+OptimizeStringConcat --XX:+DoEscapeAnalysis --XX:+UseNUMA"
```

```
#GC_DEBUG=" --XX:+PrintTenuringDistribution --XX:+PrintGCDetails --XX:+PrintGCDateAndTime"
```

```
PRODUCTION_HEAP_SETTINGS=" --Xms7G --Xmx7G -" # heap memory settings must be adjusted
JAVA_OPTIONS="{GC} {GC_DEBUG} {EX} {ENC} {DRV} {JMX_REMOTE_IP} --server {PR}"
```

### VM machine with 4GB or less of RAM, and less than 4 core CPU equivalent

Small installations with limited resources could operate better with default (for JVM versions up to 8, which is the most current at the moment of the writing). Again - depending on the traffic usage and particular use-case adjusting NewRatio may be needed. Adjusting Xms and Xmx sizes for actual available

memory is recommended (or better yet, for the actual traffic!). Following should be used (i.e. GC line should be commented so the defaults will be used):

```
#GC="-XX:+UseBiasedLocking --XX:+UseConcMarkSweepGC --XX:+UseParNewGC --XX:NewRatio=1.1"
EX="-XX:+OptimizeStringConcat --XX:+DoEscapeAnalysis --XX:+UseNUMA"
```

```
#GC_DEBUG=" --XX:+PrintTenuringDistribution --XX:+PrintGCDetails --XX:+PrintGCDateAndTime"
```

```
PRODUCTION_HEAP_SETTINGS=" --Xms3G --Xmx3G -" # heap memory settings must be adjusted
JAVA_OPTIONS="{GC} {GC_DEBUG} {EX} {ENC} {DRV} {JMX_REMOTE_IP} --server {PR
```

## Additional resources

- Sizing the Generations [<https://docs.oracle.com/javase/8/docs/technotes/guides/vm/gctuning/sizing.html>]
- About Java, parallel garbage collection and processor sets [<http://www.c0t0d0s0.org/archives/6617-About-Java,-parallel-garbage-collection-and-processor-sets.html>]
- GC Threads [<http://hiroshiyamauchi.blogspot.cl/2009/12/gc-threads.html>]
- GCViewer readme [<https://github.com/chewiebug/GCViewer#readme>]
- Java HotSpot™ Virtual Machine Performance Enhancements [<http://docs.oracle.com/javase/7/docs/technotes/guides/vm/performance-enhancements-7.html>]
- Java Garbage Collection handbook [<https://plumbr.eu/java-garbage-collection-handbook>]
- Useful JVM Flags
  - Part 1 - JVM Types and Compiler Modes [<https://blog.codecentric.de/en/2012/07/useful-jvm-flags-part-1-jvm-types-and-compiler-modes/>]
  - Part 2 - Flag Categories and JIT Compiler Diagnostics) [<https://blog.codecentric.de/en/2012/07/useful-jvm-flags-part-2-flag-categories-and-jit-compiler-diagnostics/>]
  - Part 3 - Printing all XX Flags and their Values [<https://blog.codecentric.de/en/2012/07/useful-jvm-flags-part-3-printing-all-xx-flags-and-their-values/>]
  - Part 4 - Heap Tuning [<https://blog.codecentric.de/en/2012/07/useful-jvm-flags-part-4-heap-tuning/>]
  - Part 5 - Young Generation Garbage Collection [<https://blog.codecentric.de/en/2012/08/useful-jvm-flags-part-5-young-generation-garbage-collection/>]
  - Part 6 - Throughput Collector [<https://blog.codecentric.de/en/2013/01/useful-jvm-flags-part-6-throughput-collector/>]
  - Part 7 - CMS Collector [<https://blog.codecentric.de/en/2013/10/useful-jvm-flags-part-7-cms-collector/>]
  - Part 8 - GC Logging [<https://blog.codecentric.de/en/2014/01/useful-jvm-flags-part-8-gc-logging/>]

## Session Manager

Tigase Session Manager is where most of Tigase basic options can be configured, and where many operations are controlled from. Changes to session manager can effect operations throughout an entire XMPP installation, so care must be made when changing settings here.

## Mobile Optimizations

By default, Tigase employs XEP-0352 Client State Indication which allows for a more streamlined mobile experiencing by allowing the XMPP server to suppress or reduce the number of updates sent to a client thereby reducing the number of stanzas sent to a mobile client that is inactive. This employment is contained within the processor `ClientStateIndication` and is independent from the `MobileV1`, `MobileV2`, `MobileV3` settings.

However, this can be fine tuned by using mobile plugins from Tigase which can be used at the same time by adding the following line to the `config.tdsl` file:

```
}
'sess-man' {
  -'urn:xmpp:csi:0' {
    logic = -'tigase.xmpp.impl.MobileV1'
  }
}
```

Logic Options are:

### MobileV1

Keeps all presence stanzas in queue until client is active.

```
logic = -'tigase.xmpp.impl.MobileV1'
```

### MobileV2

This setting delays delivery of presences while client is in inactive state, but only keeps the last presence for each full `jid`. **This is the default setting for CSI logic.**

```
logic = -'tigase.xmpp.impl.MobileV2'
```

### MobileV3

Keeps the same presence logic as `MobileV2`, but also queues Message Carbons. **Currently not supported by CSI processor, will cause issues.**

```
logic = -'tigase.xmpp.impl.MobileV3'
```

## Disabling CSI

If you wish to not use the `ClientStateIndication` processor, set the following in your `config.tdsl` file:

```
'sess-man' () {
  -'urn:xmpp:csi:0' (active: false) {}
}
```

## A note about Mobile Plugins

Previously, you could enable Mobile optimization logic using `--sm-plugins=+Mobile_V1`.

If you have used these in the past, it is recommended you change your system to use the CSI processor with the appropriate mobile processing logic.

If you require v3 logic, or do not wish to use CSI, be sure to disable it using the above option.

## threads-pool

The `threadsNo` property allows you to fine-tune the SM plugin's (processors) thread pool. With the default settings every plugin gets his own thread pool. This guarantees the best performance and optimal resource usage. The downside of this setting is that packets can arrive out of order if they are processed within different thread pools.

We can even fine tune this packet processing. Let's say you want most of the plugins to be executed within a single thread pool to preserve packet ordering for them, but for some selected plugins that should execute within separate thread pools to improve performance. Let's say, authentication packets and user registration can be actually executed in a separate thread pools as we do not worry about an order for them. Users cannot send or receive anything else before they authenticates anyway. The solution is to specify a number of threads for the selected plugin. For example, setting a common thread pool for all plugins but registration and authentication can be done with following configuration:

```
'sess-man' () {
  - 'amp' () {
    threadsNo = 30
  }
  - 'presence-state' () {
    threadsNo = 27
  }
}
```

This replaces the old `--sm-threads-pool` property, as well as specifying thread pools in `--sm-plugins`.

## Thread Pool factor

Session manager can control the number of available thread pools for each processor. By adding the following line to the `config.tdsl` file, the global thread pool can be increased by a specified factor:

```
'sess-man' () {
  - 'sm-threads-factor' = 3
}
```

In this case, the global thread pools is increased by a factor of 3.

## Strategy

The `Strategy` property allows users to specify Clustering Strategy class which should be used for handling clustering environment; by default `SMNonCachingAllNodes` is used.

Any class implementing `tigase.cluster.strategy.ClusteringStrategyIfc` interface may be used for this setting.

Example:

```
'sess-man' () {
  strategy (class: tigase.cluster.strategy.SMCachingAllNodes)
}
```

This replaces the old `--sm-cluster-strategy-class` setting from v7.1.

## Virtual Hosts in Tigase Server

Tigase server supports multiple virtual hosts in a single server installation. Virtual hosts can be added or removed, enabled or disabled during runtime without restarting the service or disrupting normal operation.

This document describes how virtual hosts work in Tigase server and how to get the most out of this feature in your installation.

The 'default-virtual-host' property allows to define name of the single vhost domain which will be considered a default vhost domain for this installation. It allows you just to configure the domain name. Any additional configuration needs to be configured using ad-hoc commands.

Virtual hosts should be managed using ad-hoc commands or admin ui, visit [Add and Manage Domains](#) for description of vhosts management process or visit [Specification for ad-hoc Commands Used to Manage Virtual Domains](#) for more information about ad-hoc commands.

If you have components that may not be able to handle multiple vhosts or cluster mode, we have developed a virtual component solution as well, details in the [Virtual Components for the Tigase Cluster](#) section.

You may also want to reference the Vhosts API for additional information: - [API Description for Virtual Domains Management in Tigase Server](#).

### Tip

If you have a Tigase XMPP Server running in the cluster mode hidden behind some load balancer, or if internal IP or hostname of cluster nodes differ from the DNS name under which it is available from the internet, we would suggest setting a property `installation-dns-address` of `vhost-man` component to the DNS name which allows you to connect to all cluster nodes (ie. to the DNS name of the load balancer). This will allow Tigase XMPP Server to do proper DNS lookups to verify that DNS domain name of the virtual host which you will try to add or update points to your XMPP installation.

## Default VHost configuration

It's possible to specify default configuration for all Virtual Host in TDSL configuration file (i.e. `etc/config.tdsl`) for selected parameters. To do so you should specify each configuration option within `defaults` bean belonging to `vhost-man` bean:

```
'vhost-man' () {
  -'defaults' () {
    -'domain-filter-policy' = null
    -'hardened-mode' = false
    -'s2s-secret' = null
    trusted = null
    -'vhost-disable-dns-check' = false
    -'vhost-max-users' = 0L
    -'vhost-message-forward-jid' = null
    -'vhost-presence-forward-jid' = null
    -'vhost-register-enabled' = true
    -'vhost-tls-required' = false
  }
}
```



```
}
```

## Specification for ad-hoc Commands Used to Manage Virtual Domains

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 the specification described in XEP-0050 [<http://xmpp.org/extensions/xep-0050.html>]. Extra information required to complete the command is carried as data forms described in 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 `<not-authorized />` error is returned. To grant administrator rights to an account you have to set `admins` property in the `config.tds1` configuration file.

Commands are sent to the '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 an account other than admin, 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 end-user information. The client may ignore the 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 one 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>
```

&lt;/iq&gt;

## Virtual Components for the Cluster Mode

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 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 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`

```
muc (class: 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 (class: tigase.cluster.VirtualComponent) {
  -'disco-category' = -'conference'
  -'disco-features' = -'http://jabber.org/protocol/muc'
  -'disco-name' = -'Multi User Chat'
  -'disco-node' = -''
  -'disco-type' = -'text'
  -'redirect-to' = -'muc@cluster-node-with-real-muc.domain.our'
}
```

That's it.

## Settings for Custom Logging in Tigase

Logging can be an important tool to monitor your server's health and performance. Logging may be controlled and customized on a per-component basis.

A logging bean has been implemented to allow more fine configuration of each component.

```
logging () {
  rootLevel = CONFIG
  -'packet-debug-full' = true
  loggers = {
```

```

        -'tigase.server' = {
            level = ALL
        -}
        -'tigase.conf' = {
            level = FINEST
        -}
    -}
    handlers = {
        -' java.util.logging.FileHandler' = {
            level = ALL
            append = true
            count = 5
            formatter = -'tigase.util.LogFormatter'
            limit = 10000000
            pattern = -'logs/tigase.log'
        -}
        -'java.util.logging.ConsoleHandler' = {
            level = WARNING
            formatter = -'tigase.util.LogFormatter'
        -}
    -}
}

```

You only need to specify the settings you wish to customize, otherwise they will be left as default.

- `packet-debug-full` - controls whether log entries should be obfuscated (all CData of all elements will be replaced by CData size: <length in bytes of the replaced string>) or not; default: false.
- `rootLevel` - Defines the root level of logging for all components not otherwise defined. Default is CONFIG
- `loggers` - Defines the level of logging for packages running in tigase server. This is similar to the `--debug` setting, however you must use `tigase.{package}` format. Default is NONE.
- `handlers` - Defines the level of logging for File output and Console output.
  1. `FileHandler` - is the file output for log files, with the following options:
    - a. `level` - specifies the level of logs to be written, default is ALL.
    - b. `append` - whether to append to the log or replace it during restart. Default is true.
    - c. `count` - number of individual log files to keep at set limit. Default is 5. (default settings will continue appending logs until 5 files at 10MB are reached, then the oldest file will be overwritten.)
    - d. `formatter` - specifies the package to format logging output. Default is `tigase.util.LogFormatter`.
    - e. `limit` - Byte limit for each log file. Default is 10000000 or 10MB.
    - f. `pattern` - Directory and filename of the log file with respect to the Tigase installation directory. Default is `logs/tigase.log`.
  2. `ConsoleHandler` - Determines the formatting for Tigase output to console.
    - a. `level` - specifies the level of logs to be written, default is WARNING.

- b. `formatter` - specifies the package to format logging output. Default is `tigase.util.LogFormatter`.

## Tigase Advanced Options

This section is designed to include a number of advanced configuration options available within Tigase, but may not have a relevant section yet to house them.

### Using CAPTCHA for in-band registration

To reduce false or spam registrations to Tigase XMPP Server, there is now the ability to add CAPTCHA forms as a part of the in-band registration. The CAPTCHA will generate a random math equation and ask the user registering a new account to answer it. This may be enabled as a sub-option of enabling registration in `config.tdsl`:

```
'sess-man' {
  -'jabber:iq:register' {
    captchaRequired = -'true'
  }
}
```

#### Note

Because some clients do not refresh a registration form after an unsuccessful attempt, this option allows 3 retries with the same CAPTCHA.

3 unsuccessful attempts will result in the captcha being invalidated and a client will receive an error message.

### Enabling Empty Nicknames

Tigase can now support users with empty nicknames. This can be enabled by adding the following code in `config.tdsl`.

```
'sess-man' {
  -'jabber:iq:roster' {
    empty_name_enabled = -'true'
  }
}
```

### Enable Silent Ignore on Packets Delivered to Unavailable Resources

You can now have Tigase ignore packets delivered to unavailable resources to avoid having a packet bounce around and create unnecessary traffic. You may set this globally, within standard message handling only, or within the AMP component using the following settings:

Globally:

```
'sess-man' {
  -'silently-ignore-message' = -'true'
}
```

Message Processing Only:

```
'sess-man' {  
  message {  
    -'silently-ignore-message' = -'true'  
  }  
}
```

AMP Component:

```
'sess-man' {  
  amp () {  
    -'silently-ignore-message' = -'true'  
  }  
}
```

## Mechanism to count errors within Tigase

A new processor within statistics has been added to count the number of errors that Tigase returns. This processor, named `error-counter`, will count all errors returned by Tigase, however by default the number is always zero if it is not enabled. It can be found as an MBean object in JMX under `ErrorStatistics` and contains values for packets with `ERROR` and grouped by type. To enable counting of these errors, you must ensure the processor is included in your `sess-man` configuration:

```
'sess-man' {  
  -'error-counter' () {}  
}
```

## Including stream errors

Stream `ERROR` packets are not included in the above counter by default as they are processed separately. To enable this to be added to the counter, the following line must be in your `config.tds1` file.

```
c2s {  
  -'stream-error-counter' () {  
    active = true  
  }  
}
```

## Stream resumption default & max-timeout

`SteamManagementIOProcessor` now has a setting that can be used to change the maximum timeout time it will wait for reconnection if a client does not send a time to wait. Two settings are now available:

```
c2s {  
  -'urn:xmpp:sm:3' {  
    -'resumption-timeout' = 90  
  }  
}
```

The above setting in `config.tds1` file will change the default timeout period to 90 seconds.

```
c2s {  
  -'urn:xmpp:sm:3' {  
    -'max-resumption-timeout' = 900  
  }  
}
```

```
}
```

This setting will set the maximum time allowed for stream resumption to 900 seconds. This can be handy if you expect a number of mobile phones to connect to your server and want to avoid duplicate messages being sent back and forth.

You may setup a server to automatically approve presence subscriptions or roster authorizations for all users. Say you were hosting bots and wanted to automate the process. This can be done with the following settings:

```
'sess-man' {  
  -'jabber:iq:roster' {  
    -'auto-authorize' = -'true'  
  -}  
  presence {  
    -'auto-authorize' = -'true'  
  -}  
}
```

Both of these settings are false by default, and you may use them together or separately. The following behavior is followed when they are both activated:

- Upon sending a subscription request - Both contacts will each others' subscription and be added to each others' roster. Presence information will immediately be exchanged between both parties.
- Upon sending presence with type either unsubscribe or unsubscribed follows the rules defined in RFC regarding processing of these stanzas (i.e. adjusting subscription type of user/contact), but without forwarding those stanzas to the receiving entity to avoid any notifications to the client. However, a roster push is generated to reflect changes to presence in user roster in a seamless manner.
- Simply adding an item to the roster (i.e. with <iq/> stanza with correct semantics) will also cause an automatic subscription between the user and the contact in a matter explained above.



---

# Chapter 8. Security

The articles here cover advanced security features built into Tigase Server, and some options for adding your own levels of security.

## XEP-0191: Blocking Command

The simplest security feature, however, inside an XMPP server is the ability to block users and JIDS. XEP-0191 [<http://xmpp.org/extensions/xep-0191>] specifies the parameters of simple blocking without using privacy lists. Below is a breakdown and some sample commands you may find helpful. To enable this feature, be sure the following is in your `config.tds1` file:

```
}
'sess-man' {
  -'urn:xmpp:blocking' () {}
}
```

If you have other plugins running, then just add `'urn:xmpp:blocking' () {}` to the list to activate this feature.

To confirm if your installation of Tigase supports this feature, a quick `disco#info` of your server should reveal the following feature:

```
<feature var='urn:xmpp:blocking' />
```

Blocked users are stored on the server on a per-JID basis, so one user may only see his or her blocked JIDs. Lists of blocked JIDs will return as an IQ stanza with a list of `<item>` fields. To retrieve the blocklist, the following command is issued:

```
<iq type='get' id='blockedjids'>
  <blocklist xmlns='urn:xmpp:blocking' />
</iq>
```

The server responds:

```
<iq type='result' id='blockedjids'>
  <blocklist xmlns='urn:xmpp:blocking'>
    <item jid='user1@domain.net' />
    <item jid='admin@example.com' />
  </blocklist>
</iq>
```

To block a JID, a similar stanza to the one above is sent to the server with the items of the blocked JIDs you wish to add:

```
<iq from='admin@domain.net' type='set' id='block'>
  <block xmlns='urn:xmpp:blocking'>
    <item jid='user2@domain.net' />
  </block>
</iq>
```

The server will then push an unavailable presence to blocked contacts. Communication between a contact that is blocked, and an entity that blocked it will result in a `<not-acceptable>` error:

```
<message type='error' from='user2@domain.net' to='admin@domain.net'>
  <body>Hello, are you online?</body>
  <error type='cancel'>
    <not-acceptable xmlns='urn:ietf:params:xml:ns:xmpp-stanzas' />
    <blocked xmlns='urn:xmpp:blocking:errors' />
  </error>
</message>
```

Unblocking a contact is just as easy as blocking, send an unblock stanza to the server:

```
<iq from='admin@domain.net' type='set' id='unblock'>
  <unblock xmlns='urn:xmpp:blocking'>
    <item jid='user2@domain.net' />
  </unblock>
</iq>
```

The server will begin pushing presence information to unblocked contacts and resources so long as permissions have not changed between.

You may also opt to unblock all contacts and essentially clear out your blocked list using the following command:

```
<iq type='set' id='unblockall'>
  <unblock xmlns='urn:xmpp:blocking' />
</iq>
```

## Account Registration Limits

In order to protect Tigase servers from DOS attacks, a limit on number of account registrations per second has been implemented. This may be configured by adding the following line in the `config.tdsl` file:

```
'registration-throttling' () {
  limit = 10
```

This setting allows for 10 registrations from a single IP per second. If the limit is exceeded, a `NOT_ALLOWED` error will be returned.

It is possible to create two separate counters as well:

```
'registration-throttling' () {
  limit = 10
}
c2s {
  -'registration-throttling' (class: tigase.server.xmppclient.RegistrationThrottling) {
    limit = 3
  }
}
```

Here we have one for c2s with a limit of 3, and a global for all other connection managers set at 10.

You can also set individual components limits as well:

```
ws2s {
  -'registration-throttling' (class: tigase.server.xmppclient.RegistrationThrottling) {
    limit = 7
  }
}
```

```
-}  
}
```

## Brute-force attack prevention

Brute-force Prevention is designed to protect Tigase Server against user password guessing. It counts invalid login tries and when it is above limit, it locks login ability for specific time (soft ban). When invalid login counter reaches second level, account will be disabled permanently.

### Configuration

Brute-force Prevention is configured by VHost. There is following list of configuration parameters:

<code>brute-force-lock-enabled</code>	boolean	Brute Force Prevention Enabled
<code>brute-force-lock-after-fails</code>	long	Number of allowed invalid login
<code>brute-force-period-time</code>	long	Time [sec] in what failed login tries are counted
<code>brute-force-disable-after-fails</code>	long	Threshold beyond which account will be permanently disabled
<code>brute-force-lock-time</code>	long	Time [sec] of soft ban (first threshold)
<code>brute-force-mode</code>	string	Working mode (see the section called “Working modes”)

### Working modes

There are three working modes:

- `Ip` - it counts invalid login tries from IP, and locks login ability (soft ban) for IP what reach the threshold
- `IpJid` - it counts tries from IP to specific user account. Soft ban locks ability of login to specific JID from specific IP.
- `Jid` - similar to `IpJid` but checks only JID. Soft ban locks ability of login to specific JID from all IPs.

#### Note

Only in modes `Jid` and `IpJid` account may be permanently disabled.

### Permanent ban

In modes `Jid` and `IpJid`, when invalid login counter reach threshold `brute-force-disable-after-fails`, account status will be set to disabled. To enable it again you should use Re-Enable User [<https://xmpp.org/extensions/xep-0133.html#reenable-users>] Ad-hoc Command.

## Server Certificates

- Creating and Loading the Server Certificate in pem Files

- Installing StartCom Certificate in Your Linux System

## Creating and Loading the Server Certificate in pem Files

### Server Certificates

Server certificates are needed when you use secure socket connections - SSL/TLS.

For secure socket connection a 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.

### Generating your Own Certificates

Self-signed certificates can be generated easily on a Linux system. Although it may not be considered a 'trusted' certificate authority, it can be useful to test server installations. **We do not recommend regular use of self-signed certificates.**

#### Note

that Tigase v5.0 and later can automatically create self signed PEM files if needed. However we will cover doing this process by hand.

This tutorial assumes you are running a Linux-based operating system with access to command shell, and the 'Openssl' package is installed on the system.

The process takes the following steps: 1. Create a local private key. This file ends with .key extension. It is recommended to create a new private key for the process. 2. Generate a certificate request. This file ends with the .csr extension and is the file sent to the Certificate Authority to be signed. 3. CA signs private key. This can be done by your own computer, but can also be done by private CAs for a fee. 4. Results are obtained from the CA. This is a .crt file which contains a separate public certificate. 5. Combine the .csr and .crt file into a unified .pem file. Tigase requires keys to be non-password protected PEM files.

#### Generate local private key.

```
openssl genrsa --out [domain.com.key] 1024
```

This command generates a private key using a 1024 bit RSA algorithm. -out designates the name of the file, in this case it will be domain.com.key. The exact name is not important, and the file will be created in whatever directory you are currently in.

#### Generate a certificate request:

```
openssl req --nodes --key domain.com.key --out domain.com.csr
```

This command generates a certificate request using the file specified after -key, and the result file will be domain.com.csr. You will be asked a series of questions to generate the request.

```
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 []:
```

**Sign the Certificate Request:** Now the .csr file will be signed by a Certificate Authority. In this tutorial, we will be self-signing our certificate. This practice however is generally not recommended, and you will receive notifications that your certificate is not trusted. There are commercial offers from companies to sign your certificate from trusted sources. Please see the Certificate From Other Providers section for more information.

```
openssl x509 --req --days 365 --in domain.com.csr --signkey domain.com.key --out d
```

This command signs the certificate for 365 days and generates the domain.com.crt file. You can, of course use any number of days you like.

**Generate PEM file.** You should now have the following files in the working directory: ..\domain.com.key domain.com.csr domain.com.crt

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

If the certificate is issued by third-party authority you will have to attach the certificate chain, that being 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, if you have a certificate from XMPP federation you need to download StartCom root certificate [<http://www.startssl.com/certs/ca.pem>] **and** intermediate ICA certificate [<http://www.startssl.com/certs/sub.class1.server.ca.pem>]. In such cases the pem file is created using following command:

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

The result file should looks similar to:

```
-----BEGIN CERTIFICATE-----
MIIG/TCCBeWgAwIBAgIDAOWZMA0GCSqGSIb3DQEBBQUAMIGMMQswCQYDVQQGEWJJ
.
.
.
pSLqw/PmSLSmUNIr8yQnhy4=
-----END CERTIFICATE-----
-----BEGIN RSA PRIVATE KEY-----
WW91J3JlIGtpZGRpbmchISEKSSBkb24ndCBzaG93IHlvdSBvdXIgcHJpdmF0ZSBz
.
.
.
ZXkhISEhCkNyZWf0ZSB5b3VyIG93biA7KSA7KSA7KQo=
-----END RSA PRIVATE KEY-----
-----BEGIN CERTIFICATE-----
MIIHyTCCBbGgAwIBAgIBATANBgkqhkiG9w0BAQUFADB9MQswCQYDVQQGEWJJTDEW
.
.
.
xV/stleh
```

-----END CERTIFICATE-----

For Tigase server as well as 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.**

## Installing/Loading Certificate To the Tigase Server

Installing and 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. By default, and to make things easy, we recommend the `Tigase/certs` directory.

## 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 XMPP servers, especially if it's one from a new provider. Here is an example list of providers:

- LetsEncrypt - please see the section called “Installing LetsEncrypt Certificates in Your Linux System” for details
- CAcert [<https://www.cacert.org/>] - free certificates with an excellent Web GUI for managing generated 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 a third party authority you have to go to its website and request the certificate using certificate request generated above. I cannot provide any instructions for this as each of the providers listed have different requirements and interfaces.

We **highly** recommend using LetsEncrypt keys to self-sign and secure your domain. Instructions are in the next section.

## Using one certificate for multiple domains

By default, each virtual hosts will require it's own certificate. However, if you choose to use one certificate for all virtual hosts, Tigase supports that option. For example, if you have `host1.example.net`, `host2.example.net`, and `host3.example.net` each vhost will need some configuration:

```
'certificate-container' {  
  - 'custom-certificates' {  
    - 'host1.example.net' = - '/home/tigase/certs/host1.pem'  
    - 'host2.example.net' = - '/home/tigase/certs/host2.pem'  
    - 'host3.example.net' = - '/home/tigase/certs/host3.pem'  
  }  
}
```

```
}
```

This may be time consuming if you have many Vhosts, or expect to generate many more. The good news is, now one certificate can be used for ALL Vhosts using the following configuration line:

```
'certificate-container' {  
  - 'custom-certificates' {  
    - '*.example.net' = - '/home/tigase/certs/certificate.pem'  
  -}  
}
```

Now any Vhosts created will use the same certificate located at `/home/tigase/certs/certificate.pem`.

### Note

This is an all or nothing option, if you wish to customize each Vhost, you will need to do so individually.

## Installing LetsEncrypt Certificates in Your Linux System

LetsEncrypt is a trusted CA that provides free security certificates. Unlike previously self-signed certificates, we can use LetsEncrypt Certificates to certify your domains from a trusted source.

Please refer to official [certbot User Guide](<https://certbot.eff.org/docs/using.html>) for details how to install and operate the tool, choosing desired method of domain authentication (DNS or webserver). After successful execution the certificate with all related files will be stored under `/etc/letsencrypt/live/$domain`

```
$ sudo ls -l /etc/letsencrypt/live/$domain  
cert.pem  chain.pem  fullchain.pem  privkey.pem  README
```

In that directory, you will find four files: - `privkey.pem` - private key for the certificate - `cert.pem` - contains the server certificate by itself - `chain.pem` - contains the additional intermediate certificate or certificates - `fullchain.pem` - all certificates, including server certificate (aka leaf certificate or end-entity certificate). The server certificate is the first one in this file, followed by any intermediates.

For Tigase XMPP Server, we are only concerned with `privkey.pem` and `fullchain.pem`.

At this point we will need to obtain the root and intermediate certificates, this can be done by downloading these certificates from the LetsEncrypt website [<https://letsencrypt.org/certificates/>].

Alternatively, you may obtain them using `wget`:

```
wget https://letsencrypt.org/certs/isrgrootx1.pem  
wget https://letsencrypt.org/certs/letsencryptauthorityx3.pem
```

These are the root certificate, and the intermediate certificate signed by root certificate.

### Note

IdenTrust cross-signed certificate will not function properly.

Take the contents of your `privkey.pem`, certificate, and combine them with the contents of `isrgrootx1.pem` and `letsencryptauthorityx3.pem` into a single pem certificate. You need to name the file after your domain such as `mydomain.com.pem` and place it under `certs/` subdirectory of Tigase XMPP Server installation

If you moved all certs to a single directory, you may combine them using the following command under \*nix operating systems:.

```
cat -./cert.pem -./privkey.pem -./letsencryptauthorityx3.pem -./isrgrootx1.pem > m
```

Your certificate should look something like this:

```
-----BEGIN PRIVATE KEY-----
MIIEvgIBADANBgkqhkiG9w0BAQEFAASCBAgEAAoIBAQDAUAqgKu7Z4odo
...
og89F9AbWr1mNmyRoScyqMXo
-----END PRIVATE KEY-----
-----BEGIN CERTIFICATE-----
cmNoIEdyb3VwMRUwEwYDVQQDEwxJU1JHIFJvb3QgWDEwHhcNMTUwNjA0MTEwNDM4
...
TzELMAkGA1UEBhMCVVMxKTAnBgNVBAoTIEludGVybmV0IFNlY3VyaXR5IFJlc2Vh
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
FhpodHRwOi8vY3BzLmxdHNlbnNyeXB0Lm9yZzCBqWYIKwYBBQUHAgIwgZ4MgZtU
...
bmCGUGFydGllcyBhbmQgb25seSBpbjBhY2NvcnRhbmNlIHdpdGggdGhlIENlcnRp
-----END CERTIFICATE-----
```

## Warning

LetsEncrypt certificates expire 90 days from issue and need to be renewed in order for them to remain valid!

# Custom Authentication Connectors

This article presents configuration options available to the administrator and describe how to set Tigase server up to use user accounts data from a different database.

The first thing to know is that Tigase server always opens 2 separate connections to the database. One connection is used for user login data and the other 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 Tigase server keeps user data in it's own database and only login data is retrieved from the external database.

At the moment Tigase 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/>].
- `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.

- **tigase-auth (DEPRECATED)** - 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.

As always the simplest way to configure the server is through the `config.tds1` 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:

```
dataSource {
  -'default-auth' () {
    uri = -'database connection url'
  -}
}
authRepository {
  default () {
    cls = -'connector'
    -'data-source' = -'default-auth'
  -}
}
```

For example if you store authentication data in a drupal database on localhost your settings would be:

```
dataSource {
  -'default-auth' () {
    uri = -'jdbc:mysql://localhost/drupal?user=user&password=passwd'
  -}
}
authRepository {
  default () {
    -'data-source' = -'default-auth'
  -}
}
```

You have to use a class name if you want to attach your own authentication connector.

Default is:

```
authRepository {
  default {
    cls = -'tigase.db.jdbc.TigaseAuth'
  -}
}
```

In the same exact way you can setup connector for any different database type.

For example, drupal configuration is below

```
authRepository {
    default {
        cls = -'tigase.db.jdbc.DrupalWPAuth'
    }
}
```

Or tigase-custom authentication connector.

```
authRepository {
    default {
        cls = -'tigase.db.jdbc.TigaseCustomAuth'
    }
}
```

The different `cls` or classes are:

- Drupal - `tigase.db.jdbc.DrupalWPAuth`
- MySQL, Derby, PostgreSQL, MS SQL Server - `tigase.db.jdbc.JDBCRepository`

You can normally skip configuring connectors for the default Tigase database format: `mysql`, `pgsql` and `derby`, `sqlserver` as they are applied automatically if the parameter is missing.

One more important thing to know is that you will have to modify `authRepository` 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 an external system. User accounts are added without notifying 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 the 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:

```
dataSource {
    default () {
        uri = -'jdbc:mysql://localhost/tigasedb?user=nobody&password=pass&autoCrea
    }
}
```

If you are interested in even further customizing your authentication connector by writing your own queries or stored procedures, please have a look at the following guides:

- Tigase Auth guide
- Tigase Custom Auth guide

## Tigase Auth Connector (DEPRECATED)

### Warning

Tigase Auth connector is **DEPRECATED** as of version 8.0.0 and will be removed in future releases

The Tigase Auth connector with shortcut name: **tigase-auth** is implemented in the class: `tigase.db.jdbc.TigaseAuth` [<https://projects.tigase.org/projects/tigase-server/repository/changes/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 the Custom Authentication Connectors guide.

To make this connector working you have to prepare your database to offer set of stored procedures for 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 repository [<https://tigase.tech/projects/tigase-server/repository/revisions/master/show/src/main/database>] for the schema definition files.

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/logouts 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

## Tigase Custom Auth Connector

The Tigase Custom Auth connector with shortcut name: **tigase-custom** is implemented in the class: `tigase.db.jdbc.TigaseCustomAuth` [<https://projects.tigase.org/projects/tigase-server/repository/changes/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 the Custom Authentication Connectors guide.

The basic configuration is very simple:

```
authRepository {
  default () {
    cls = -'tigase.db.jdbc.TigaseCustomAuth'
    -'data-source' = -'default-auth'
  }
}
```

That's it.

The connector loads correctly and starts working using predefined, default list of queries. In most cases you also might want to define your own queries in the configuration file. The shortest possible description is the following example of the content from the `config.tdsl` file:

This query is used to check connection to the database, whether it is still alive or not

```
authRepository {
  default () {
    -'conn-valid-query' = -'select 1'
  -}
}
```

This is database initialization query, normally we do not use it, especially in clustered environment

```
authRepository {
  default () {
    -'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. Tigase checks whether the JID returned from the query matches JID passed as a parameter. If they match, the authentication is successful.

```
authRepository {
  default () {
    -'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.

```
authRepository {
  default () {
    -'users-count-query' = -'{ call TigAllUsersCount() -}'
  -}
}
```

The Below query is used to add a new user account to the database.

```
authRepository {
  default () {
    -'add-user-query' = -'{call TigAddUserPlainPw(?, -?) -}'
  -}
}
```

Below query is used to remove existing account with all user's data from the database.

```
authRepository {
  default () {
    -'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. In such a case the Tigase server loads user's password from the database and compares it with data received from the client.

```
authRepository {
  default () {
    -'get-password-query' = -'select user_pw from tig_users where user_id = -?
  -}
}
```

Below query is used for user password update in case user decides to change his password.

```
authRepository {
  default () {
    -'update-password-query' = -'update tig_users set user_pw = -? where user_
  -}
}
```

This query is called on user logout event. Usually we use a stored procedure which records user logout time and marks user as offline in the database.

```
authRepository {
  default () {
    -'update-logout-query' = -'update tig_users, set online_status = online_st
  -}
}
```

This configuration specifies what non-sasl authentication mechanisms to expose to the client

```
authRepository {
  default () {
    -'non-sasl-mechs' = [ -'password', -'digest' -]
  -}
}
```

This setting to specify what sasl authentication mechanisms expose to the client

```
authRepository {
  default () {
    -'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.

Some queries can 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.

This example shows how to use a stored procedure to add a user as a query with 2 required parameters (username, and password).

```
authRepository {
  default () {
```

```

    - '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.

Query Name	Description	Arguments	Example Query
conn-valid-query	Query executed periodically to ensure active connection with the database.	Takes no arguments.	select 1
init-db-query	Database initialization query which is run after the server is started.	Takes no arguments.	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)	insert into tig_users (user_id, user_pw) values (?, ?)
del-user-query	Removes a user from the database.	Takes 1 argument: (user_id (JID))	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))	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))	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.	Takes 2 arguments: (user_id (JID), password)	select user_id from tig_users where (user_id = ?) AND (user_pw = ?)

Query Name	Description	Arguments	Example Query
	Normally this method should be only used with plain text password authentication or sasl-plain. Tigase expects a result set with <code>user_id</code> to be returned from the query if login is successful and empty results set if the login is unsuccessful.		
<code>user-logout-query</code>	This query is called when user logs out or disconnects. It can record that event in the database.	Takes 1 argument: <code>(user_id (JID))</code>	<code>update tig_users, set online_status = online_status - 1 where user_id = ?</code>
<code>non-sasl-mechs</code>	Comma separated list of NON-SASL authentication mechanisms. Possible mechanisms are: <code>password</code> and <code>digest</code> . The digest mechanism can work only with <code>get-password-query</code> active and only when password are stored in plain text format in the database.		
<code>sasl-mechs</code>	Comma separated list of SASL authentication mechanisms. Possible mechanisms are all mechanisms supported by Java implementation. The most common are: <code>PLAIN</code> , <code>DIGEST-MD5</code> , <code>CRAM-MD5</code> . "Non-PLAIN" mechanisms will work only with the <code>get-password-query</code> active and only when passwords are stored in plain text format in the database.		

## Drupal Authentication

Currently, we can only check authentication against a **Drupal** database at the moment. Full **Drupal** authentication is not implemented as of 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 a **Drupal** database based authorization is located in `tigase.db.jdbc.DrupalAuth` class. Although this class is capable of adding new users 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. Functionality for adding new users is implemented only to ease user accounts migration from different repository types from earlier **Tigase** server installations.

The purpose 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 XMPP even if user provides valid password.

There is no *Roster* management in **Drupal** yet. So Roster management have to be done from the XMPP client.

## LDAP Authentication Connector

Tigase XMPP Server offers support for authenticating users against an LDAP server in **Bind Authentication** mode.

Configuration for the LDAP support is really simple you just have to add a few lines to your `config.tdsl` file.

```
authRepository {
  default () {
    cls = -'tigase.db.ldap.LdapAuthProvider'
    uri = -'ldap://ldap.tigase.com:389'
    -'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

## Configuration of SASL EXTERNAL

In order to enable SASL External add following line to the `config.tdsl` 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, XEP-0178 [<http://xmpp.org/extensions/xep-0178.html#c2s>].

It is possible to make client certificate required:

```
c2s {
  clientCertRequired = true
}
```

If this option will be enabled, then client must provide certificate. This certificate will be verified against `clientCertCA`. If client does not provide certificate or certificate will be invalid, TLS handshake will be interrupted and client will be disconnected.



Using this options does not force client to use SASL EXTERNAL. Client still may authenticate with other SASL mechanisms.

## Packet Filtering

Tigase offers different ways to filter XMPP packets flying through the server. The most common use for packet filtering is to restrict users from sending or receiving packets based on the sender or received address.

There are also different possible 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 packet filtering 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 to allow employees 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 a 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 users can not change the settings. 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. The 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.
7. CUSTOM - user can communicate only within custom created rules set.

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 users 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 the domain filter plugin which must be loaded at startup time. It is loaded by default if the 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 is possible for each user separately, it is not practical for large installations. In most cases users are stored in the database and a 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 enables modifying 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. `CUSTOM` - list of rules defining custom communication permissions (server processes stanza according to first matched rule, similar to XEP-0016) in the following format:

```
ruleSet = rule1;rule2;ruleX;  
  
rule = order_number|policy|UID_type[|UID]
```

```
order_number = any integer;
policy = (allow|deny);
UID_type = [jid|domain|all];
UID = user JID or domain, for example pubsub@test.com; if UID_type is ALL then thi
```

For example:

```
1|allow|self;
2|allow|jid|admin@test2.com;
3|allow|jid|pubsub@test.com;
4|deny|all;
```

1. 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 the `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.

## Access Control Lists in Tigase

Tigase offers support for **Access Control List (ACL)** to allow for fine grained access to administration commands on the server.

By default, all administration commands are only accessible (visible through service discovery and can be executed) by the service administrators. Service administrators are existing accounts with JIDs (**BareJIDs**) listed in the `config.tdsl` file under `admins = [ ]` (please see the section called "admins" for details).

Additionally, other XMPP users and entities can be assigned permissions to execute a command or commands using Tigase's ACL capabilities.

The following is a list of possible ACL modifiers for administrator command accessibility:

- ALL - Everybody can execute the command, even users from different federated servers.
- ADMIN - Local server administrators can execute the command, this is a default setting if no ACL is set for a command.
- LOCAL - All users with accounts on the local server can execute the command. Users from other, federated servers will not be able to execute the command.
- NONE - No one will be allowed to execute this command
- DOMAIN\_OWNER - Only user which is owner of the domain which items are being manipulated is allowed to execute the comment. If script is not checking permissions for the manipulated item, this value will behave in the same way as LOCAL.
- DOMAIN\_ADMIN - Only user which is one of the domain administrators will be able to execute the command manipulating items related to the domain. If script is not checking permissions for the manipulated item, this value will behave in the same way as LOCAL.

- `example.com` - Only users with accounts on the selected domain will be able to execute the command. It may be useful to setup a domain specifically for admin accounts, and automatically all users within that domain would be able to run the command.
- `user@example.com` - Comma separated list of JIDs of users who can execute the command.

In any case, regardless of ACL settings, any command can be executed and accessed by the designated service wide administrators, that is accounts listed as admins in the `config.tdsl` file.

Multiple ACL modifiers can be combined and applied for any command. This may not always makes sense. For example ALL supersedes all other settings, so it does not make sense to combine it with any other modifier. However, most others can be combined with JID to broaden access to specific accounts.

On Tigase server the Access Control List is checked for the first matching modifier. Therefore if you combine ALL with any other modifier, anybody from a local or remote service will always be able to execute the command, no matter what other modifiers are added.

Please note, the ACL lists work on the command framework level. Access is verified before the command is actually executed. There might be additional access restrictions within a command itself. In many cases, even if all local users are permitted to execute a command (LOCAL modifier), some commands allow only to be executed by a domain owner or a domain administrator (and of course by the service-wide administrators as well). All the commands related to a user management such as adding a new user, removing a user, password changes, etc... belong to this category. When conducting domain (vhost) management, creation/registration of a new domain can be done by any local user (if LOCAL ACL modifier is set) but then all subsequent domain management tasks such as removing the vhost, updating its configuration, setting SSL certificate can be done by the domain owner or administrator only.

The ACL list is set for a specific Tigase component and a specific command. Therefore the configuration property must specify all the details. So the general format for configuring ACL for a command is this:

```
comp-id () {
    commands {
        -'command-id' = [ -'ACL_modifier', -'ACL_modifier', -'ACL_modifier' -]
    }
}
```

The breakdown is as such:

- `comp-id` is the Tigase server component ID such as: `sess-man`, `vhost-man`, `c2s`, etc..
- `commands` is a static text which indicates that the property is for component's command settings.
- `command-id` is a command ID for which we set the ACL such as `query-dns`, `http://jabber.org/protocol/admin#add-user`, `user-roster-management`, etc...

Here are a few examples:

*Allowing local users to create and manage their own domains*

```
'vhost-man' () {
    commands {
        -'comp-repo-item-add' = -'LOCAL'
        -'comp-repo-item-remove' = -'LOCAL'
        -'comp-repo-item-update' = -'LOCAL'
        -'ssl-certificate-add' = -'LOCAL'
    }
}
```

```
}
```

In fact all the commands except item-add can be executed by the domain owner or administrator.

*Allowing local users to execute user management commands:*

```
'sess-man' () {  
  -'commands' {  
    -'http://jabber.org/protocol/admin#add-user' = -'LOCAL'  
    -'http://jabber.org/protocol/admin#change-user-password' = -'LOCAL'  
    -'http://jabber.org/protocol/admin#delete-user' = -'LOCAL'  
    -'http://jabber.org/protocol/admin#get-online-users-list' = -'LOCAL'  
    -'http://jabber.org/protocol/admin#get-registered-user-list' = -'LOCAL'  
    -'http://jabber.org/protocol/admin#user-stats' = -'LOCAL'  
    -'http://jabber.org/protocol/admin#get-online-users-list' = -'LOCAL'  
  -}  
}
```

As in the previous example, the commands will be executed only by local users who are the specific domain administrators.

*Allowing users from a specific domain to execute query-dns command and some other users for given JIDs from other domains:*

```
'vhost-man' () {  
  -'commands' {  
    -'query-dns' = [ -'tigase.com', -'admin@tigase.org', -'frank@example.com'  
  -}  
}
```

To be able to set a correct ACL property you need to know component names and command IDs. Component IDs can be found in the service discovery information on running server or in the server logs during startup. A command ID can be found in the command script source code. Each script contains a list of metadata at the very beginning of its code. One of them is `AS:CommandId` which is what you have to use for the ACL setting.

---

# Chapter 9. Database Management

Tigase is coded to perform with multiple database types and numbers. Owing to it's versatile nature, there are some tools and procedures that may be of use to certain administrators.

## Recommended database versions

As of v8.0.0 here are the minimum and recommended versions of databases for use with Tigase:

Database	Recommended Version	Minimum Version	Additional Information
DerbyDB	10.12.1.1	10.12.1.1	Included with Tigase XMPP Server
MySQL	5.7	5.6.4	Required to properly support timestamp storage with millisecond precision
SQLServer	2014	2012	2012 needed so we can count use fetch-offset pagination feature.
PostgreSQL	9.5	9.4	New UA schema requires at least 9.4
MongoDB	3.2	3.0	
MariaDB	?	10.0.12	Basic features works with 10.0.12-MariaDB Homebrew, but is not fully tested.

Although Tigase may support other versions of databases, these are the ones we are most familiar with in offering support and advice. Use of databases outside these guidelines may result in unforeseen errors.

## Database Watchdog

It is possible to have Tigase test availability and existence of database periodically only when db connections are idle. By default this ping is sent once every 60 minutes to each connected repository. However this can be overridden as a part of the dataSource property:

```
dataSource {
  default () {
    uri = -'....'
  }
  -'test' () {
    uri = -'...'
    -'watchdog-frequency' = -'PT30M'
  }
}
```

This setting changes frequency to 30 minutes.

```
dataSource {
```

```
default () {  
    uri = -'...'   
-}  
- 'watchdog-frequency' = - 'PT15M'  
}
```

This one changes to 15 minutes.

### Note

see the section called “Period / Duration values” for format details

## Using modified database schema

If you are using Tigase XMPP Server with modified schema (changed procedures or tables) and you do not want Tigase XMPP Server to maintain it and automatically upgrade, you can disable schema-management for any data source. If schema-management is disabled for particular data source then Tigase XMPP Server will not update or modify database schema in any way. Moreover it will not check if schema version is correct or not.

**Disabling schema-management for default data source.**

```
dataSource {  
    default () {  
        uri = -'...'   
        - 'schema-management' = false  
    -}  
}
```

### Warning

If schema-management is disabled, then it is administrator responsibility to maintain database schema and update it if needed (ie. if Tigase XMPP Server schema was changed).

## Database Preparation

Tigase uses generally the same database 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 database system.

Starting with v8.0.0, most of the database tasks have been automated and can be called using simple text, or using interactive question and answer style. We **DO NOT RECOMMEND** going through manual operation, however we have kept manual activation of different databases to the Appendix. If you are interested in how we manage and update our database schemas, you may visit the Schema Maintenance [[https://tigase.tech/projects/tigase-server/wiki/Schema\\_files\\_maintenance](https://tigase.tech/projects/tigase-server/wiki/Schema_files_maintenance)] section of our Redmine installation for more detailed information.

- The DBSchemaLoader Utility
- Hashed User Passwords in Database
- Support for MongoDB

Appendix entries

- Manual installation for MySQL
- Manual installation for Derby
- Manual installation for SQLServer
- Manual installation for PostgreSQL

## Schema Utility

With the release of v8.0.0 calling the Tigase dbSchemaLoader utility now can be done using tasks instead of calling the specific method. Support for Derby, MySQL, PostgreSQL, MSSQL, and MongoDB is available.

In order to use this utility with any of the databases, you will need to first have the database environment up and running, and have established user credentials. You may use root or an account with administrator write privileges.

## Operation & Variables

### Operation

Operating the schema utility is quite easy! To use it run this command from the installation directory:

```
./scripts/tigase.sh [task] [params_file.conf] [options]
```

Operations are now converted to tasks, of which there are now three: `install-schema`, `upgrade-schema`, and `destroy-schema`.

- `upgrade-schema`: Upgrade the schema of the database specified in your `config.tdsl` configuration file. (options are ignored for this option)
- `install-schema`: Install a schema to database.
- `destroy-schema`: Destroy database and schemas. **DANGEROUS**

### Options

Use the following options to customize. Options in bold are required, *{potential options are in brackets}*:

- `--help` Prints the help for the task.
- `-I` or `--interactive` - enables interactive mode which will prompt for parameters not defined.
- `-T` or `--dbType` - database type {derby, mongodb, mysql, postgresql, sqlserver}.
- `-C` or `--components` - Allows the specification of components for use when installing a schema.

## Usage

### upgrade-schema

This task will locate any schema versions above your current one, and will install them to the database configured in the `config.tdsl` file.



## Note

To use this utility, you must have Tigase XMPP server fully setup with a configured configuration file.

```
./scripts/tigase.sh upgrade-schema etc/tigase.conf
```

Windows users will need to run the command using the following command:

```
java --cp -"jars/*" tigase.db.util.SchemaManager -"upgrade-schema" ---config-file=
```

## install-schema

This task will install a schema using the parameters provided.

**If you are setting up a server manually, we HIGHLY recommend using this method**

```
./scripts/tigase.sh install-schema [Options]
```

This command will install tigase using a Derby database on one named `tigasedb` hosted on `localhost`. The username and password editing the database is `tigase_pass` and `root`. Note that `-J` explicitly adds the administrator, this is highly recommended with the `-N` passing the password.

If you are using a windows system, you need to call the program directly:

```
java --cp -"jars/*" tigase.db.util.SchemaManager -"install-schema" [options]
```

## Options

Options for schema installation are as follows, required options are in bold

- `--help`, Outputs the help.
- `-I, --interactive` - enables interactive mode, which will result in prompting for any missing parameters.
- `-C, --components=` - list of enabled components identifiers (+/-), possible values: [amp, bosh, c2s, eventbus, ext-disco, http, mdns, message-archive, monitor, muc, pubsub, push, s2s, socks5, test, unified-archive, upload, ws2s] (default: amp,bosh,c2s,eventbus,http,message-archive,monitor,muc,pubsub,s2s,ws2s). **This is required for certain components like socks5.**
- `-T, --dbType=` - database server type, possible values are: [derby, mongodb, mysql, postgresql, sqlserver] (*required*)
- `-D, --dbName=` - name of the database that will be created (by default it is `tigasedb`). (*required*)
- `-H, --dbHostname=` - address of the database instance (by default it is `localhost`). (*required*)
- `-U, --dbUser=` - name of the user that will be created specifically to access Tigase XMPP Server database (default is `tigase_user`). (*required*)
- `-P, --dbPass=` - password of the user that will be created specifically to access Tigase XMPP Server database (default is `tigase_pass`). (*required*)
- `-R, --rootUser=` - database root account username used to create user and database (default is `root`). (*required*)

- `-A, --rootPass=` - database root account password used to create user and database (default is root). (*required*)
- `-S, --useSSL` - enable SSL support for database connection (if the database supports it) (default is false).
- `-F, --file=` - comma separated list of SQL files that will be processed.
- `-Q, --query=` - custom queries to be executed, see the section called “Query function” for details.
- `-L, --logLevel=` - logger level used during loading process (default is CONFIG).
- `-J, --adminJID=` - comma separated list of administrator JID(s).
- `-N, --adminJIDpass=` - password that will be used for the entered JID(s) - one password for all configured JIDs.
- `--getURI=` - generate database URI (default is false).
- `--ignoreMissingFiles=` - force ignoring missing files errors (default is false).

### Query function

Should you decide to customize your own functions, or have specific information you want to put into the database, you can use the `-query` function to perform a single query step.

```
./scripts/tigase.sh install-schema --T mysql --D tigasedb --R root --A root --Q -"
```

Of course this would break the schema for tigasedb by adding an unexpected table, you will receive the following message:

```
tigase.db.util.DBSchemaLoader      printInfo      WARNING      Database sche
```

But this is a demonstration how you may run a query through the database without the need to use another tool. Note that you will need to select the specific database for each query.

### destroy-schema

This will destroy the database specified in the configuration file.

#### Warning

##### THIS ACTION IS NOT REVERSIBLE

```
./scripts/tigase.sh destroy-schema etc/config.tdsl
```

Only use this if you wish to destroy a database and not have the information recoverable.

Windows users will need to call the method directly:

```
java --cp -"jars/*" tigase.db.util.SchemaManager -"destroy-schema" etc/config.tdsl
```

### A note about MySQL

If you are using these commands, you may result in the following error:

```
tigase.util.DBSchemaLoader      validateDBConnection      WARNING      Table - 'perform'
```

If this occurs, you will need to upgrade your version of MySQL using the following command:

```
mysql_upgrade --u root --p ---force
```

After entering the password and upgrading MySQL the schema error should no longer show when working with Tigase databases.

## Prepare the MySQL Database for the Tigase Server

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

The MySQL database can be prepared in many ways. Most Linux distributions contain tools which allow you to go through all steps from the shell command line. To make sure it works on all platforms in the same way, we will first show 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 from the Tigase installation directory:

```
mysql --u root --p
```

Once logged in, create the database for the Tigase server:

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

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;
```

## Installing Schemas

Starting with v8.0.0 the Schemas are no longer linked, and will need to manually be installed in the following order.

Switch to the database you have created:

```
mysql> use tigasedb;
```

### Note

We are assuming you run the mysql client in Linux from the Tigase installation directory, so all file links will be relative.

Next install the schema files:

```
mysql> source database/mysql-common-0.0.1.sql;
```

You will need to repeat this process for the following files:

```
mysql-common-0.0.1.sql
mysql-common-0.0.2.sql
mysql-server-7.0.0.sql
mysql-server-7.1.0.sql
mysql-server-8.0.0.sql
mysql-muc-3.0.0.sql
mysql-pubsub-3.1.0.sql
mysql-pubsub-3.2.0.sql
mysql-pubsub-4.0.0.sql
mysql-http-api-2.0.0.sql
```

Other components may require installation such as:

```
mysql-socks5-2.0.0.sql
mysql-push-1.0.0.sql
mysql-message-archiving-2.0.0.sql
mysql-unified-archive-2.0.0.sql
```

### Windows instructions:

On Windows you have probably to enter the full path, assuming Tigase is installed in C:\Program Files\Tigase:

```
mysql> source c:/Program Files/Tigase/database/mysql-common-0.0.1.sql;
mysql> source c:/Program Files/Tigase/database/mysql-common-0.0.2.sql;
mysql> source c:/Program Files/Tigase/database/mysql-server-7.0.0.sql;
and so on...
```

## Configuring From the Linux Shell Command Line

Follow steps below to prepare the MySQL database:

Create the database space for the Tigase server:

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

Add the `tigase_user` user and grant 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:

## Selective access configuration

### 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
```

## Schema Installation

Load the proper `mysql` schemas into the database.

```
mysql --u dbuser --p tigasedb < mysql-common-0.0.1.sql
mysql --u dbuser --p tigasedb < mysql-common-0.0.2.sql
etc..
```

You will need to repeat this process for the following files:

```
mysql-common-0.0.1.sql
mysql-common-0.0.2.sql
mysql-server-7.0.0.sql
mysql-server-7.1.0.sql
mysql-server-8.0.0.sql
mysql-muc-3.0.0.sql
mysql-pubsub-3.1.0.sql
mysql-pubsub-3.2.0.sql
mysql-pubsub-4.0.0.sql
mysql-http-api-2.0.0.sql
```

Other components may require installation such as:

```
mysql-socks5-2.0.0.sql
mysql-push-1.0.0.sql
mysql-message-archiving-2.0.0.sql
mysql-unified-archive-2.0.0.sql
```

## Configuring MySQL for UTF-8 Support

In `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 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 need to fix it using the command:

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

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

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

And then check that 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 your settings. You might also want to check your 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 that MySQL 5.0.x also needs 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/config.tds1` file and append this to the database connection string.

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

## Other MySQL Settings Worth Considering

There are a number of other useful options, especially for performance improvements. Please note, you will 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
```

These settings may not be fully optimized for your system, and have been only tested on our systems. If you have found better settings for your systems, feel free to let us know [<http://tigase.net/contact>].

## Support for emoji and other icons

Tigase Database Schema can support emojis and other icons, however by using UTF-8 in `mysqld` settings will not allow this. To employ settings to support emojis and other icons, we recommend you use the following in your MySQL configuration file:

```
[mysqld]
character-set-server = utf8mb4
collation-server = utf8mb4_bin
```

Doing this, Tigase XMPP Server Database will still use `utf8` character set, with `utf8_general_ci` as collation, and only fields which require support for emojis will be converted to `utf8mb4`.

NOTE:Database URI passed in Tigase XMPP Server config **must not** contain `&characterEncoding=UTF-8` as in other case it will override `utf8mb4` client charset with `utf8` charset! NOTE:Tigase XMPP Server databases should be created with `utf8_general_ci` collation as it will work properly and is fastest from `utf8` collations supported by MySQL

# Prepare the Derby Database for the Tigase Server

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

## Basic Setup

Preparation of 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)

### Note

You must use these sql files on order FIRST!

#### Linux

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

#### Windows

```
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 common version for common v0.1.

You will need to repeat this process again in for following order:

```
derby-common-0.0.1.sql
derby-common-0.0.2.sql
derby-server-7.0.0.sql
derby-server-7.1.0.sql
derby-server-8.0.0.sql
derby-muc-3.0.0.sql
derby-pubsub-3.1.0.sql
derby-pubsub-3.2.0.sql
derby-pubsub-4.0.0.sql
derby-http-api-2.0.0.sql
```

Other components may require installation such as:

```
derby-socks5-2.0.0.sql
derby-push-1.0.0.sql
derby-unified-archive-2.0.0.sql
```

## Connecting Tigase to database

Once the database is setup, configure the config.tds1 file in Tigase and add the following configuration:



```
dataSource {  
    default () {  
        uri = -'jdbc:derby:{location of derby database}';  
    }  
}
```

## Prepare the MS SQL Server Database for the Tigase Server

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

It's expected that a working installation of Microsoft SQL Server is present. The following guide will describe the necessary configurations required for using MS SQL Server with Tigase XMPP Server.

### Preparing MS SQL Server Instance

After installation of MS SQL Server an instance needs to be configured 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 MS SQL Server to handle. Subsequently set the TCP Port property (if missing) to the default value - 1433. A restart of the instance may be required afterwards.

### Configuration using MS SQL Server Management Studio

In order to prepare the database you can use either a wizard or execute queries directly in the Query Editor. Firstly you need to establish a connection to the MS SQL Server instance. 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-common-0.0.1.sql  
sqlserver-common-0.0.2.sql  
sqlserver-server-7.0.0.sql  
sqlserver-server-7.1.0.sql  
sqlserver-server-8.0.0.sql  
sqlserver-muc-3.0.0.sql  
sqlserver-pubsub-3.1.0.sql  
sqlserver-pubsub-3.2.0.sql  
sqlserver-pubsub-4.0.0.sql  
sqlserver-http-api-2.0.0.sql
```

### Note

These files must be done sequentially! They are not linked, and so may need to be done one at a time.

Other components may require installation such as:

```
sqlserver-socks5-2.0.0.sql  
sqlserver-push-1.0.0.sql  
sqlserver-message-archiving-2.0.0.sql  
sqlserver-unified-archive-2.0.0.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, execute following commands from the directory where Tigase XMPP Server is installed otherwise 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  
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 the following defaults are used:

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

## Tigase configuration - config.tds1

Configuration of the MS SQL Server follows general database convention.

```
dataSource {
    default () {
        uri = -'jdbc:[jtds:]sqlserver://db_hostname:port[;property=val]'
    }
}
```

where any number of additional parameters 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:

```
dataSource {
    default () {
        uri = -'jdbc:sqlserver://hostname:1433;databaseName=tigasedb;user=tigase;p
    }
}
```

## 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. Starting with the version 7.1.0 we recommend using jTDS driver that is shipped with Tigase as JDBC driver created by Microsoft can cause problems with some components in cluster installations. MS driver can be downloaded from the website: JDBC Drivers 4.0, 4.1 for SQL Server [<http://www.microsoft.com/en-us/download/details.aspx?displaylang=en&id=11774>] then unpack the archive. Copy `sqljdbc_4.0/enu/sqljdbc4.jar` file to `${tigase-server}/jars` directory.

Depending on the driver used `uri` needs to be configured accordingly.

- Microsoft driver:

```
dataSource {  
    default () {  
        uri = -'jdbc:sqlserver://...'  
    }  
}
```

- jDTS driver

```
dataSource {  
    default () {  
        uri = -'jdbc:jdts://...'  
    }  
}
```

## Prepare the PostgreSQL Database for the Tigase Server

This guide describes how to prepare PostgreSQL database for connecting to Tigase server.

The PostgreSQL database can be prepared in many ways. Below are presented two possible ways. The 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:

Add the `tigase_user`:

```
psql=# create role tigase_user with login password -'tigase123';
```

Next, Create the database for the Tigase server with `tigase_user` as owner of database:

```
psql=# create database tigasedb owner tigase_user;
```

## Schema Installation

Load database schema to initialize the Tigase server from the file that corresponds to the version of Tigase you want to use. First you need to switch to `tigasedb`.

```
psql=# \connect tigasedb
```

Begin by applying the basic Schema

```
psql=# \i database/postgresql-common-0.0.1.sql
```

Continue by adding the schema files listed below:

```
postgresql-common-0.0.1.sql  
postgresql-common-0.0.2.sql  
postgresql-server-7.0.0.sql
```

```
postgresql-server-7.1.0.sql
postgresql-server-8.0.0.sql
postgresql-muc-3.0.0.sql
postgresql-pubsub-3.1.0.sql
postgresql-pubsub-3.2.0.sql
postgresql-pubsub-4.0.0.sql
postgresql-http-api-2.0.0.sql
```

Other components may require installation such as:

```
postgresql-socks5-2.0.0.sql
postgresql-push-1.0.0.sql
postgresql-message-archiving-2.0.0.sql
postgresql-unified-archive-2.0.0.sql
```

## Configuring From the Linux Shell Command Line

Follow steps below to prepare the PostgreSQL database:

First, 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.

Create the database for the Tigase server with `tigase_user` as owner of database:

```
createdb --U admin_db_user --W --O tigase_user tigasedb
```

## Database Schema Installation

Load database schema to initialize the Tigase server

```
psql --q --U tigase_user --W tigasedb --f database/postgresql-common-0.0.1.sql
psql --q --U tigase_user --W tigasedb --f database/postgresql-common-0.0.2.sql
etc..
```

Continue by adding the schema files listed below:

```
postgresql-common-0.0.1.sql
postgresql-common-0.0.2.sql
postgresql-server-7.0.0.sql
postgresql-server-7.1.0.sql
postgresql-server-8.0.0.sql
postgresql-muc-3.0.0.sql
postgresql-pubsub-3.1.0.sql
postgresql-pubsub-3.2.0.sql
postgresql-pubsub-4.0.0.sql
postgresql-http-api-2.0.0.sql
```

Other components may require installation such as:

```
postgresql-socks5-2.0.0.sql
postgresql-push-1.0.0.sql
postgresql-message-archiving-2.0.0.sql
```

```
postgresql-unified-archive-2.0.0.sql
```

## Note

The 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.

# Preparing Tigase for MongoDB

Tigase now supports MongoDB for auth, settings, and storage repositories. If you wish to use MongoDB for Tigase, please use this guide to help you.

## Dependencies

To run Tigase MongoDB support library requires drivers for MongoDB for Java which can be downloaded from here [<https://github.com/mongodb/mongo-java-driver/releases>]. This driver needs to be placed in jars/ directory located in Tigase XMPP Server installation directory. If you are using a dist-max distribution, it is already included.

## Configuration

Note that fresh installations of MongoDB do not come with users or databases installed. Once you have setup MongoDB you will need to create a user to be used with Tigase. To do this, bring up the mongo console by running mongo.exe in a cmd window for windows, or run mongo in linux. Once connected, enter then following:

```
use admin
db.createUser( { user: -"tigase",
                  pwd: -"password",
                  customData: { employeeId: 12345 -},
                  roles: [ -"root" -]
                -}
              -)
```

Be sure to give this user a root role in order to properly write to the database. Once you receive a user successfully created message, you are ready to install tigase on MongoDB.

## Configuration of user repository for Tigase XMPP Server

To configure Tigase XMPP Server to use MongoDB you need to set dataSource in etc/config.tdsl file to proper MongoDB URI pointing to which MongoDB database should be used (it will be created by MongoDB if it does not exist). userRepository property should not be set to let Tigase XMPP Server auto-detect proper implementation of UserRepository. Tigase XMPP Server will create proper collections in MongoDB if they do not exist so no schema files are necessary.

Example configuration of XMPP Server pointing to MongoDB database tigase\_test in a local instance:

```
dataSource {
    default () {
        uri = -'mongodb://user:pass@localhost/tigase_test'
    }
}
userRepository {
```

```
    default () {}  
  }  
  authRepository {  
    default () {}  
  }
```

If Tigase Server is not able to detect a proper storage layer implementation, it can be forced to use one provided by Tigase using the following lines in `etc/config.tdsl` file:

```
userRepository {  
  default () {  
    cls = -'tigase.mongodb.MongoRepository'  
  }  
}  
authRepository {  
  default () {  
    cls = -'tigase.mongodb.MongoRepository'  
  }  
}
```

Every component should be able to use proper implementation to support MongoDB using this URI. Also MongoDB URI can be passed as any URI in configuration of any component.

## Configuration for MUC

By default, MUC component will use MongoDB to store data if Tigase is configured to use it as a default store. However, if you would like to use a different MongoDB database to store MUC message archive, you can do this by adding the following lines to `etc/config.tdsl` file:

```
muc {  
  -'history-db-uri' = -'mongodb://user:pass@localhost/tigase_test'  
}
```

If MUC components fails to detect and use a proper storage layer for MongoDB, you can force it to use one provided by Tigase by using the following line in the `config.tdsl` file:

```
muc {  
  -'history-db' = -'tigase.mongodb.muc.MongoHistoryProvider'  
}
```

## Configuration for PubSub

By default, PubSub component will use MongoDB to store data if Tigase is configured to use it as a default store. However, if you would like to use a different MongoDB database to store PubSub component data, you can do this by adding the following lines to `etc/config.tdsl` file:

```
pubsub {  
  -'pubsub-repo-url' = -'mongodb://user:pass@localhost/tigase_test'  
}
```

If the PubSub components fails to detect and use a proper storage layer for MongoDB, you can force it to use one provided by Tigase by using the following line in the `config.tdsl` file:

```
pubsub {  
  -'pubsub-repo-class' = -'tigase.mongodb.pubsub.PubSubDAOMongo'
```

```
}
```

## Configuration for Message Archiving

By default, the Message Archiving component will use MongoDB to store data if Tigase is configured to use it as a default store. However, if you would like to use a different MongoDB database to store message archives, you can do this by adding the following lines to `etc/config.tds1` file:

```
'message-archive' {  
    -'archive-repo-uri' = -'mongodb://user:pass@localhost/tigase_test'  
}
```

If Message Archiving component fails to detect and use a proper storage layer for MongoDB, you can force it to use one provided by Tigase by using the following line in the `config.tds1` file:

```
'message-archive' {  
    -'archive-repo-class' = -'tigase.mongodb.archive.MongoMessageArchiveRepository'  
}
```

## Schema Description

This description contains only basic description of schema and only basic part of it. More collections may be created if additional components of Tigase XMPP Server are loaded and configured to use MongoDB.

### Tigase XMPP Server Schema

Basic schema for UserRespository and AuthRepository consists of two collections: . `tig_users` - contains list of users . `tig_nodes` - contains data related to users in tree-like way

`tig_users` collection contains the following fields:

**Table 9.1. `tig_users`**

Name	Description
<code>_id</code>	id of user which is SHA256 hash of users jid (raw byte array).
<code>user_id</code>	contains full user jid.
<code>domain</code>	domain to which user belongs for easier lookup of users by domain.
<code>password</code>	password of user (will be removed after upgrade to 8.0.0).

`tig_nodes` collection contains the following fields

**Table 9.2. `tig_nodes`**

Name	Description
<code>_id</code>	id of row auto-generated by MongoDB.



Name	Description
uid	id of user which is SHA256 hash of users jid (raw byte array).
node	full path of node in tree-like structure separated by / (may not exist).
key	key for which value for node is set.
value	value which is set for node key.

Tigase XMPP Server also uses additional collections for storage of Offline Messages

**Table 9.3. msg\_history collection**

Name	Description
from	full user jid of message sender.
from_hash	SHA256 hash of message sender jid as raw byte array.
to	full users jid of message recipient.
to_hash	SHA256 hash of message recipient full jid as raw byte array.
ts	timestamp of message as date.
message	serialized XML stanza containing message.
expire-at	timestamp of expiration of message (if message contains AMP expire-at set).

<mongodb-schema-changes-8>  
<title>Additions for v8.0 Schema</title>

Due to changes in authentication and credentials storage in AuthRepository, we moved password field from tig\_users collection to a newly created collection called tig\_user\_credentials.

This new collection has following fields:

Name	Description
_id	id of document automatically generated by MongoDB

Name	Description
uid	SHA256 hash of a user for which credentials are stored
username	username provided during authentication (or default)
account_status	name of an account state (copy of value stored in user document from `tig_users`)

Additionally for each mechanism we store separate field in this object, so for:

- PLAIN we have PLAIN field with value for this mechanism
- SCRAM-SHA-1 we have SCRAM-SHA-1 field with value for this mechanism
- etc...

Upgrade is not done in one step, and rather will be done once a particular user will log in. During authentication if there is no data in `tig_user_credentials`, Tigase XMPP Server will check if `password` field in `tig_user` exists. If it does, and it is filled credentials will be migrated to the new collection.

</mongodb-schema-changes-8>

## Hashed User Passwords in Database

### Warning

This feature is still available, but passwords are stored encrypted by default since v8.0.0. We do not recommend using these settings.

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.

Storing passwords in hashed format in the database makes it possible to avoid using a plain-text password authentication mechanism. 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 the plain-text password authentication method is perhaps less of a problem than storing plain-text passwords in the database.

Nevertheless, it is simple enough to adjust this in Tigase's database.

### 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 a 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 including 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`. This procedure accepts plain passwords 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, in plain-text and 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 by 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)

## Tigase Server and Multiple Databases

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 sharing 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.

```
dataSource {
    default () {
        uri = -'jdbc:mysql://db2.tigase/dbname?user&password'
    -}
    -'default-auth' () {
        uri = -'jdbc:mysql://db1.tigase/dbname?user&password'
    -}
}
userRepository {
    default () {}
}
authRepository {
    default () {
        cls = -'tigase.db.jdbc.TigaseCustomAuth'
        -'data-source' = -'default-auth'
    -}
}
```

This configuration defines 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

```
dataSource {
  default () {
    uri = -'jdbc:mysql://db2.tigase/dbname?user&password'
  }
  -'default-auth' () {
    uri = -'jdbc:mysql://db1.tigase/dbname?user&password'
  }
}
userRepository {
  default () {}
}
authRepository {
  default () {
    cls = -'tigase.db.jdbc.TigaseCustomAuth'
    -'data-source' = -'default-auth'
  }
}
```

Now, we have VHost: domain1.com User authentication data for this VHost is stored in Drupal database

```
dataSource {
  -'domain1.com-auth' () {
    uri = jdbc:mysql://db7.tigase/dbname?user&password'
  }
}
authRepository {
  domain1.com () {
    cls = -'tigase/db/jdbc.TigaseCustomAuth'
    -'data-source' = -'domain1.com-auth'
  }
}
```

All other user data is stored in Tigase's standard database in MySQL

```
dataSource {
  -'domain1.com' () {
    uri = jdbc:mysql://db4.tigase/dbname?user&password'
  }
}
userRepository {
  domain1.com () {}
}
```

Next VHost: domain2.com User authentication is in LDAP server but all other user data is stored in Tigase's standard database

```
authRepository {
  domain2.com () {
```

```
        cls = -'tigase.db.ldap.LdapAuthProvider'  
        uri = -'ldap://ldap.domain2.com:389'  
        -'data-source' = -'default'  
    -}  
}
```

Now is something new, we have a custom authentication repository and separate user settings for a single domain. Please note how we define the VHost for which we set custom parameters

```
authRepository {  
    domain2.com {  
        -'user-dn-pattern' = -'cn=,ou=,dc=,dc='  
    -}  
}
```

All other user data is stored in the same as default repository

```
userRepository {  
    domain2.com () {}  
}  
dataSource {  
    domain2.com () {  
        uri = -'jdbc:mysql://db2.tigase/dbname?user&password'  
    -}  
}
```

When combined, the DSL output should look like this:

```
dataSource {  
    domain2.com () {  
        uri = -'jdbc:mysql://db2.tigase/dbname?user&password'  
    -}  
}  
userRepository {  
    domain2.com () {}  
}  
authRepository {  
    domain2.com () {  
        cls = -'tigase.db.ldap.LdapAuthProvider'  
        uri = -'ldap://ldap.domain2.com:389'  
        -'user-dn-pattern' = -'cn=,ou=,dc=,dc='  
    -}  
}
```

Next VHost: domain3.com Again user authentication is in LDAP server but pointing to a different LDAP server with different access credentials and parameters. User information is stored in a PostgreSQL database.

```
dataSource {  
    domain3.com () {  
        uri = -'jdbc:pgsql://db.domain3.com/dbname?user&password'  
    -}  
}  
userRepository {  
    domain3.com () {}  
}
```

```
}
authRepository {
  domain3.com () {
    cls = -'tigase.db.ldap.LdapAuthProvider'
    uri = -'ldap://ldap.domain3.com:389'
    -'user-dn-pattern' = -'cn=,ou=,dc=,dc='
  -}
}
```

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.

```
dataSource {
  domain4.com () {
    uri = -'jdbc:mysql://db14.domain4.com/dbname?user&password'
  -}
}
userRepository {
  domain4.com () {}
}
authRepository {
  domain4.com () {
    cls = -'tigase.db.jdbc.TigaseCustomAuth'
    -'user-login-query' = -'{ call UserLogin(?, -?) -}'
    -'user-logout-query' = -'{ call UserLogout(?) -}'
    -'sasl-mechs' = [ -'PLAIN', -'DIGEST-MD5' -]
  -}
}
```

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.

## Importing User Data

You can easily copy data between Tigase compatible repositories that is repositories for which there is a database connector. However, it is not that easy 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 a 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 the source repository class name and the initialization string. Therefore there are a few example preset parameters which you can use 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/tiga

```

```
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 will have to execute a command like this one:

```

SELECT a, b, c, d INTO OUTFILE -'import-file.txt'
FIELDS TERMINATED BY -','
LINES TERMINATED BY -'\n'
FROM test_table;

```

# Importing Existing Data

Information about importing user data from other databases.

## Connecting the Tigase Server to MySQL Database

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 with Tigase server.

This guide describes MySQL database connection parameters.

This guide is actually very short as there are example configuration files which can be used and customized for your environment.

```
dataSource {
    default () {
        uri = -'jdbc:mysql://localhost/tigasedb?user=tigase_user&password=mypass'
    }
}
userRepository {
    default () {}
}
authRepository {
    default () {}
}
```

This is the basic setup for setting up an SQL repository for Tigase. `dataSource` contains the `uri` for `default` which is the `mysql` database. MySQL connector requires connection string in the following format: `jdbc:mysql://[hostname]/[database name]?user=[user name]&password=[user password]`

Edit the `config.tds1` file for your environment.

Start the server using following command:

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

## Integrating Tigase Server with Drupal

First of all, Tigase can authenticate users against a Drupal database which means you have the same user account for both Drupal website and the XMPP server. Moreover 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 followed by Tigase server too.

## Connecting to Drupal Database

The best way to setup Tigase with Drupal database is via the `config.tds1` file where you can put initial setting for Tigase configuration.

If you look in `etc/` directory of your Tigase installation you should find a the file there.

All you need to connect to Drupal database is set the following:

```
dataSource {
```

```
-'default-auth' () {  
    uri = -'jdbc:mysql://localhost/drupal?user=drupalusr&password=drupalpass'  
-}  
}  
authRepository {  
    default () {  
        cls = -'tigase.db.jdbc.DrupalWPAuth'  
        -'data-source' = -'default-auth'  
    -}  
}
```

Typically, you will need to have drupal for authentication, and another for user repository. In this case, we will use SQL for user DB.

```
dataSource {  
    default () {  
        uri = -'jdbc:mysql://localhost/tigasedb?user=tigase_user&password=myspass'  
    -}  
    -'default-auth' () {  
        uri = -'jdbc:mysql://localhost/drupal?user=drupalusr&password=drupalpass'  
    -}  
}  
userRepository {  
    default () {}  
}  
authRepository {  
    default () {  
        cls = -'tigase.db.jdbc.DrupalWPAuth'  
        -'data-source' = -'default-auth'  
    -}  
}
```

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 MySQL guide.

Now run the Tigase server.

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

Now you can register an account on your Drupal website and connect with an XMPP client using the account details.

## Note

You have to enable plain password authentication in your XMPP client to connect to Tigase server with Drupal database.

## PostgreSQL Database Use

This guide describes how to configure Tigase server to use PostgreSQL [<http://www.postgresql.org/>] database as a user repository.

If you used an 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 the database schema:

```
# psql --U tigase --d tigasedb --f postgresql-schema.sql
```

Now the database is ready for Tigase server to use.

## Server Configuration

Server configuration is almost identical to MySQL database setup. The only difference is the connection string which usually looks like:

```
dataSource {
    default () {
        uri = 'postgresql://localhost/tigasedb?user=tigase'
    }
}
```

## Schema Updates

This is a repository for Schema updates in case you have to upgrade from older installations.

- Tigase Server Schema v7.1 Updates Applies to v7.1.0 and v8.0.0

## Changes to Schema in v8.0.0

For version 8.0.0 of Tigase XMPP Server, we decided to improve authentication and security that was provided. In order to do this, implementation of repository and database schemas needed to be changed to achieve this goal. This document, as well one in the HTTP API, will describe the changes to the schemas in this new version.

### Reasons

Before version 8.0.0, user passwords were stored in plaintext in `user_pw` database field within `tig_users` table, but in plaintext. It was possible to enable storage of the MD5 hash of the password instead, however this limited authentication mechanism SASL PLAIN only. However an MD5 hash of a password is not really a secure method as it is possible to revert this mechanism using rainbow tables.

Therefore, we decided to change this and store only encrypted versions of a password in PBKDF2 form which can be easily used for SCRAM-SHA-1 authentication mechanism or SCRAM-SHA-256. SASL PLAIN mechanism can also use these encrypted passwords. The storage of encrypted passwords is now enabled **by default** in v8.0.0 of Tigase.

## Summary of changes

### Added support for storage of encrypted password

Passwords are no longer stored in plaintext on any database.

### Using same salt for any subsequent authentications

This allows clients to reuse calculated credentials and keep them instead of storing plaintext passwords.

### Disabled usage of stored procedure for authentication

In previous versions, Tigase used stored procedures `TigUserLoginPlainPw` and `TigUserLogin` for SASL PLAIN authentication. From version 8.0.0, those procedures are no longer used, but they are updated to use passwords stored in `tig_user_credentials` table.

It is still possible to use this procedures for authentication, but to do that you need add:

```
'user-login-query' = -'{ call TigUserLoginPlainPw(?, -?) -}'
```

to configuration block of **every** authentication repository.

To enable this for default repository, the `authRepository` configuration block will look like this:

```
authRepository () {  
  default () {  
    -'user-login-query' = -'{ call TigUserLoginPlainPw(?, -?) -}'  
    -}  
  }  
}
```

### Deprecated API

Some methods of `AuthRepository` API were deprecated and should not be used. Most of them were used for authentication using stored procedures, retrieval of password in plaintext or for password change.

For most of these methods, new versions based on `tig_user_credentials` table and user credentials storage are provided where possible.

### Deprecated storage procedures

Stored procedures for authentication and password manipulation were updated to a new form, so that will be possible to use them by older versions of Tigase XMPP Server during rolling updates of a cluster. However, these procedures will not be used any more and will be depreciated and removed in future versions of Tigase XMPP Server.

### Usage of MD5 hashes of passwords

If you have changed `password-encoding` database property in previous versions of Tigase XMPP Server, then you will need to modify your configuration to keep it working. If you wish only to allow access using old passwords and to store changed passwords in the new form, then you need to enable credentials decoder for the correct authentication repository. In this example we will provided changes required for MD5-PASSWORD value of `password-encoding` database property. If you have used a different one, then just replace MD5-PASSWORD with MD5-USERNAME-PASSWORD or MD5-USERID-PASSWORD.

**Usage of MD5 decoder.**

```
authRepository () {  
  default () {  
    credentialDecoders () {  
      - 'MD5-PASSWORD' () {}  
    }  
  }  
}
```

If you wish to store passwords in MD5 form then use following entries in your configuration file:

**Usage of MD5 encoder.**

```
authRepository () {  
  default () {  
    credentialEncoders () {  
      - 'MD5-PASSWORD' () {}  
    }  
  }  
}
```

## Enabling and disabling credentials encoders/decoders

You may enable which encoders and decoders used on your installation. By enabling encoders/decoders you are deciding in what form the password is stored in the database. Those changes may impact which SASL mechanisms may be allowed to use on your installation.

**Enabling PLAIN decoder.**

```
authRepository () {  
  default () {  
    credentialDecoders () {  
      - 'PLAIN' () {}  
    }  
  }  
}
```

**Disabling SCRAM-SHA-1 encoder.**

```
authRepository () {  
  default () {  
    credentialEncoders () {  
      - 'SCRAM-SHA-1' (active: false) {}  
      - 'SCRAM-SHA-256' (active: false) {}  
    }  
  }  
}
```

## Warning

It is strongly recommended not to disable encoders if you have enabled decoder of the same type as it may lead to the authentication issues, if client tries to use a mechanism which that is not available.

## Schema changes

This change resulted in a creation of the new table `tig_user_credentials` with following fields:

uid	id of a user row in <code>tig_users</code> .
username	username used for authentication (if <code>authzid</code> is not provided or <code>authzid</code> localpart is equal to <code>authcid</code> then row with default value will be used).
mechanism	name of mechanism for which this credentials will be used, ie. <code>SCRAM-SHA-1</code> or <code>PLAIN</code> .
value	serialized value required for mechanism to confirm that credentials match.

## Warning

During execution of `upgrade-schema` task, passwords will be removed from `tig_users` table from `user_pw` field and moved to `tig_user_credentials` table.

## Added password reset mechanism

As a part of Tigase HTTP API component and Tigase Extras, we developed a mechanism which allows user to reset their password. To use this mechanism HTTP API component and its REST module **must** to be enabled on Tigase XMPP Server installation.

## Note

Additionally this mechanism need to be enabled in the configuration file. For more information about configuration of this mechanism please check Tigase HTTP API component documentation.

Assuming that HTTP API component is configured to run on port 8080 (*default*), then after accessing address `http://localhost:8080/rest/user/resetPassword` in the web browser it will present a web form. By filling and submitting this form, the user will initiate a password reset process. During this process, Tigase XMPP Server will send an email to the user's email address (provided during registration) with a link to the password change form.

## Upgrading from v7.1.x

When upgrading from previous versions of Tigase, it is recommended that you first backup the database. Refer to the documentation of your database software to find out how to export a copy. Once the backup is made, it will be time to run the schema upgrade. Be sure that your schema is up to date, and should be v7.1.0 Schema.

To upgrade, use the new `upgrade-schema` task of SchemaManager:

- In linux

```
./scripts/tigase.sh install-schema etc/tigase.conf
```

- In Windows

```
java --cp -"jars/*" tigase.db.util.SchemaManager -"install-schema"
```

You will need to configure the following switches:

- `-T` Specifies Database Type Possible values are: `mysql`, `derby`, `sqlserver`, `postgresql`, `mongodb`
- `-D` Specifies Databse Name The explicit name of the database you wish to upgrade.

- -H Specifies Host address By default, this is localhost, but may be set to IP address or FQDNS address.
- -U Specifies Username This is the username that is authorized to make changes to the database defined in -D.
- -P Specifies Password The password for username specified in -U.
- -R Password for Administrator or Root DB account.
- -A Password for Administrator or Root DB account.
- -J Jid of user authorized as admin user from Tigase.
- -N Password for user specified in -J.
- -F Points to the file that will perform the upgrade. Will follow this form database/{dbname}-server-schema-8.0.0.sql

## Tigase Server Schema v7.2 Updates

### FOR ALL USERS UPGRADING TO v8.0.0 FROM A v7.0.2 INSTALLATION

The schema has changed for the main database, and the pubsub repository. In order to upgrade to the new schemas, you will need to do the following:

1. Upgrade the Main database schema to v7.1 using the database/\${DB\_TYPE}-schema-upgrade-to-7-1.sql file
2. Upgrade the Pubsub Schema to v3.1.0 using the database/\${DB\_TYPE}-pubsub-schema-3.1.0.sql file
3. Upgrade the Pubsub Schema to v3.2.0 using the database/\${DB\_TYPE}-pubsub-schema-3.2.0.sql file
4. Upgrade the Pubsub Schema to v3.3.0 using the database/\${DB\_TYPE}-pubsub-schema-3.3.0.sql file

All three commands may be done at the same time in that order, it is suggested you make a backup of your current database to prevent data loss.

## Tigase Schema Change for v7.1

Tigase has made changes to its database to include primary keys in the tig\_pairs table to improve performance of the Tigase server. This is an auto-incremented column for Primary Key items appended to the previous schema.

### Warning

**You MUST update your database to be compliant with the new schema. If you do not, Tigase will not function properly.**

### Note

*This change will affect all users of Tigase using v7.1.0 and newer.*



If you are installing a new version of v8.0.0 on a new database, the schema should automatically be installed.

First, shut down any running instances of Tigase to prevent conflicts with database editing. Then from command line use the DBSchemaLoader class to run the -schema-upgrade-to-7.1.sql file to the database. The command is as follows:

In a linux environment

```
java --cp -"jars/*" tigase.db.util.DBSchemaLoader --dbHostname ${HOSTNAME} --dbType $
```

In a windows environment

```
java --cp jars/* tigase.db.util.DBSchemaLoader --dbHostname ${HOSTNAME} --dbType $
```

All variables will be required, they are as follows:

- \${HOSTNAME} - Hostname of the database you wish to upgrade.
- \${DB\_TYPE} - Type of database [derby, mysql, postgresql, sqlserver].
- \${ROOT\_USER} - Username of root user.
- \${ROOT\_USER\_PASS} - Password of specified root user.
- \${DB\_USER} - Login of user that can edit database.
- \${DB\_USER\_PASS} - Password of the specified user.
- \${DB\_NAME} - Name of the database to be edited.
- \${DB\_VERSION} - In this case, we want this to be 7.1.
- \${ADMIN\_JID} - Bare JID of a database user with admin privileges. Must be contained within quotation marks.
- \${ADMIN\_JID\_PASS} - Password of associated admin JID.

Please note that the SQL file for the update will have an associated database with the filename. i.e. postgresql-update-to-7.1.sql for postgresql database.

A finalized command will look something like this:

```
java --cp -"jars/*" tigase.db.util.DBSchemaLoader --dbHostname localhost --dbType
```

Once this has successfully executed, you may restart your server. Watch logs for any db errors that may indicate an incomplete schema upgrade.

## Changes to Pubsub Schema

Tigase has had a change to the PubSub Schema, to upgrade to PubSub Schema v7.1 without having to reform your databases, use this guide to update your databases to be compatible with the new version of Tigase.

### Note

Current PubSub Schema is v3.3.0, you will need to repeat these instructions for v3.1.0, v3.2.0 and then v3.3.0 before you run Tigase V7.1.0.

The PubSub Schema has been streamlined for better resource use, this change affects all users of Tigase. To prepare your database for the new schema, first be sure to create a backup! Then apply the appropriate PubSub schema to your MySQL and it will add the new storage procedure.

All these files should be in your /database folder within Tigase, however if you are missing the appropriate files, use the links below and place them into that folder.

The MySQL schema can be found Here [<https://projects.tigase.org/projects/tigase-pubsub/repository/revisions/master/entry/database/mysql-pubsub-schema-3.3.0.sql>].

The Derby schema can be found Here [<https://projects.tigase.org/projects/tigase-pubsub/repository/changes/database/derby-pubsub-schema-3.3.0.sql>].

The PostgreSQL schema can be found Here [<https://projects.tigase.org/projects/tigase-pubsub/repository/changes/database/postgresql-pubsub-schema-3.3.0.sql>].

The same files are also included in all distributions of v8.0.0 in [tigaseroot]/database/. All changes to database schema are meant to be backward compatible.

You can use a utility in Tigase to update the schema using the following command from the Tigase root:

- Linux

```
java --cp -"jars/*" tigase.db.util.DBSchemaLoader
```

- Windows:

```
java --cp jars/* tigase.db.util.DBSchemaLoader
```

## Note

**Some variation may be necessary depending on how your java build uses -cp option**

Use the following options to customize. Options in bold are required.:

- `-dbType database_type {derby, mysql, postgresql, sqlserver}` (*required*)
- `-schemaVersion schema version {4, 5, 5-1}`
- `-dbName database name` (*required*)
- `-dbHostname database hostname` (default is localhost)
- `-dbUser tigase username`
- `-dbPass tigase user password`
- `-rootUser database root username` (*required*)
- `-rootPass database root password` (*required*)
- `-file path to sql schema file` (*required*)
- `-query sql query to execute`
- `-logLevel java logger Level`
- `-adminJID comma separated list of admin JIDs`

- -adminJIDpass password (one for all entered JIDs)

## **Note**

Arguments take following precedent: query, file, whole schema

As a result your final command should look something like this:

```
java --cp -"jars/*" tigase.db.util.DBSchemaLoader --dbType mysql --dbName tigasedb
```

---

# Chapter 10. Components

The only step is to tell the server what components to load, how to name them and optionally give some extra parameters. To do so open the `config.tdsl` file you use in your installation.

Let's say you want to just add PubSub for now. All you need to do is add the following to the properties file:

```
pubsub (class: tigase.pubsub.PubSubComponent) {}
```

Normally, this is not necessary since pubsub is loaded by default, however this is just an example of loading a class with the DSL format.

```
'pubsub-priv' (class: tigase.pubsub.PubSubComponent) {}
```

As you can see, we can customize the name of a component in the deceleration, here we are using `pub-sub-priv`.

Although this may be rare, it allows for wide compatibility and platform stability.

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:

```
muc (class: tigase.muc.MUCComponent) {}
pubsub (class: tigase.pubsub.PubSubComponent) {}
```

Changes to the `config.tdsl` file will take effect upon server restart.

## Advanced Message Processing - AMP XEP-0079

Tigase server offers support for XEP-0079: Advanced Message Processing [<http://xmpp.org/extensions/xep-0079.html>] (often abbreviated to AMP).

It is enabled by default but there are several configuration options that you may tweak.

Configuration of AMP is not very complex, but as it 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.

```
'sess-man' {
  amp () {
    -'amp-jid' = -'amp@your-domain.tld'
  }
  message (active: false) {}
  msgoffline (active: false) {}
}
'amp-security-level' = -'STRICT'
```

### 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. Those are 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):

```
'sess-man' {  
  amp () {}  
  message (active: false) {}  
  msgoffline (active: false) {}  
}
```

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`:

```
'sess-man' {  
  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:

```
dataSource {  
  - 'default-amp' () {  
    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 authorized for presence updates. There are a few possible ways to prevent this.

Tigase's implementation offers 3 modes to handle AMP requests to prevent revealing user's status to non-authorized users:

```
'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 efficiently by just returning 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 users are authorized to receive notifications, even if it may reveal user status to unauthorized users. It does not impact the server performance and it offers full AMP compliance.

```
'amp-security-level' = - 'NONE'
```

## Server Monitoring

All the documentation and resources related to the Tigase server monitoring.

- Setting up Remote Monitoring in the Server
- Statistics Logger Configuration
- Retrieving Statistics from the Server
- Event Bus

## Setting Up Remote Monitoring in the Server

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 while the HTTP server doesn't offer any access restriction mechanisms. Therefore HTTP monitoring is recommended to operate 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 on without worrying about performance degradation.

NOTE This works with the Tigase server from version **4.2.0** or build **1418**.

## What You Need

Statistics binaries are built-in `-dist-max` and no extra files are needed. If you have downloaded `-dist` file, you will need `tigase-extras`[<https://tigase.tech/projects/tigase-extras/repository>] built and included in the `jars/` directory.

## Activation

You can either run the Tigase installer and use the configuration wizard to activate the monitoring or edit `etc/config.tdsl` file and add following lines:

```
monitoring() {  
  jmx() {  
    port = 9050  
  }  
  http() {  
    port = 9080  
  }  
  snmp() {  
    port = 9060  
  }  
}
```

As you see there is a separate block for each monitoring server 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 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://projects.tigase.org/projects/tigase-server/repository/changes/src/main/resources/mib/JVM-MANAGEMENT-MIB.mib>] for the server specific statistics. The MIB contains definition for all the server statistics exposed via SNMP.

## HTTP

Access the server at `example.com:9080` and you will be presented with an Agent View.

## Retrieving statistics from the server

By default we can retrieve server statistics using XMPP, no additional setup is necessary.

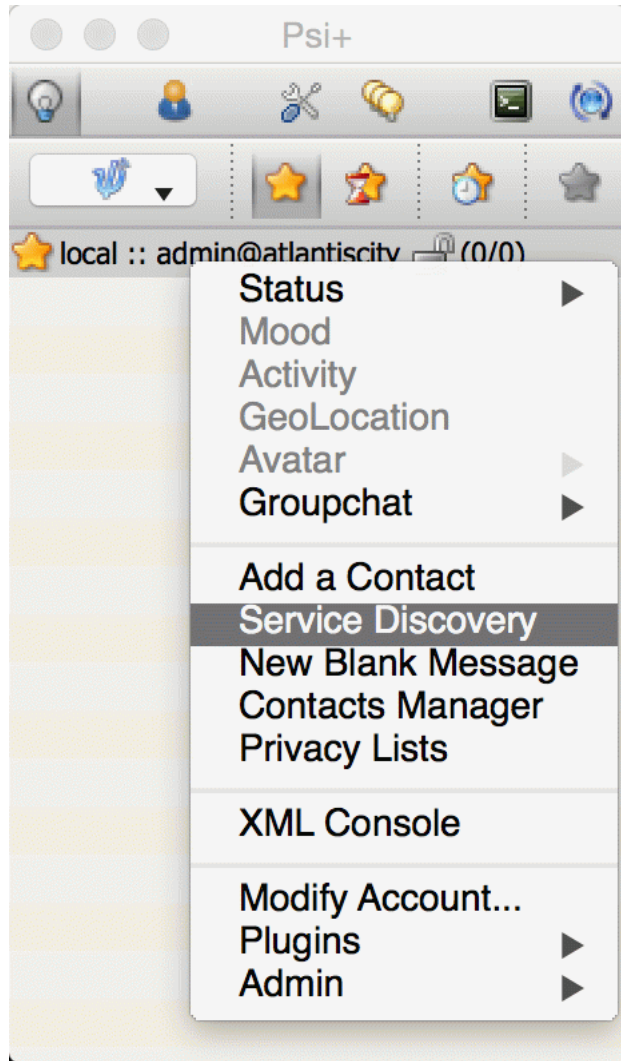
## Retrieving statistics using XMPP

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 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:





In the *Service Discovery* window we need to find *Server Statistics* component:

The screenshot shows the 'Service Discovery' application window. At the top, there is a toolbar with various icons for navigation and actions. Below the toolbar, the 'Address' field is set to 'atlantiscity' and the 'Node' field is empty. The main area displays a table of components and their JIDs.

Name	JID
▶ ★ Cluster connection manager	cl-comp@atlantiscity
▶ ★ Cluster controller	cluster-contr@atlantiscity
▶ ★ Monitor	monitor@atlantiscity
▶ ★ S2S connection manager clustered	s2s@atlantiscity
▶ ★ Session manager clustered, acs-online-cache...	sess-man@atlantiscity
▼ ★ Server statistics	stats@atlantiscity
▶ ★ Component: vhost-man	stats@atlantiscity
▶ ★ Component: message-router	stats@atlantiscity
▶ ★ Component: amp	stats@atlantiscity
▶ ★ Component: bosh	stats@atlantiscity
▶ ★ Component: c2s	stats@atlantiscity
▶ ★ Component: cl-comp	stats@atlantiscity
▶ ★ Component: monitor	stats@atlantiscity
▶ ★ Component: s2s	stats@atlantiscity
▶ ★ Component: sess-man	stats@atlantiscity
▶ ★ Component: ws2s	stats@atlantiscity
▶ ★ VHost Manager	vhost-man@atlantiscity
▶ ★ Websocket connection manager clustered	ws2s@atlantiscity

Below the table, there is a 'Filter by JID:' field and two checkboxes: 'Auto-browse into objects' and 'Automatically get item information'. At the bottom, there is a button labeled 'PSI'.

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:

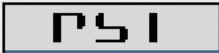
sess-man/Open user  
connections:

sess-man/Maximum user  
connections:

sess-man/Open user  
sessions:

presence/Users status  
changes:

Stats level:  

 Previous Next Cancel Finish

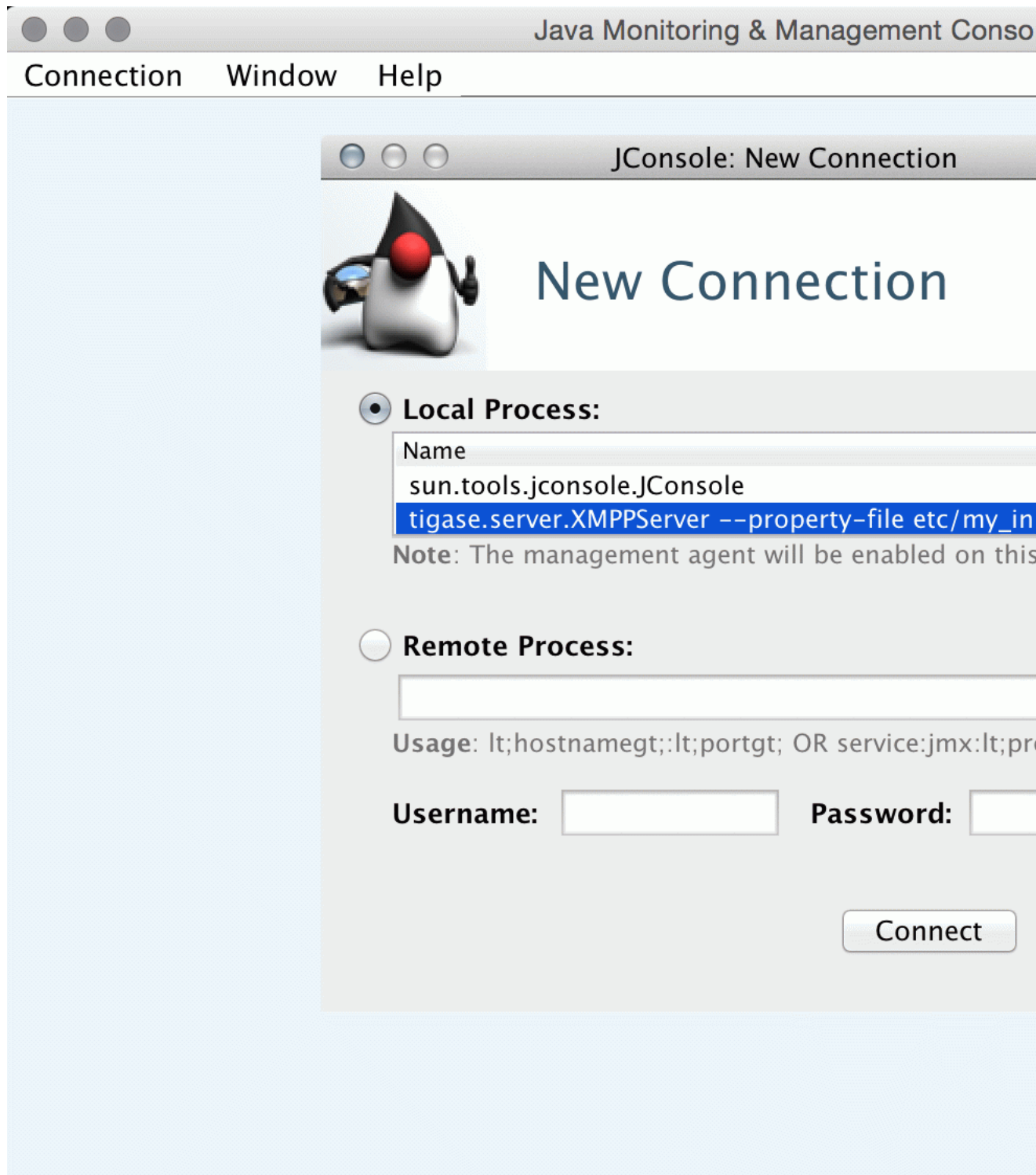
In this window, in addition to see the statistics, we can adjust *Stats level* by selecting desired level from the list and confirm by clicking *Finish*.

## Retrieving statistics using JMX

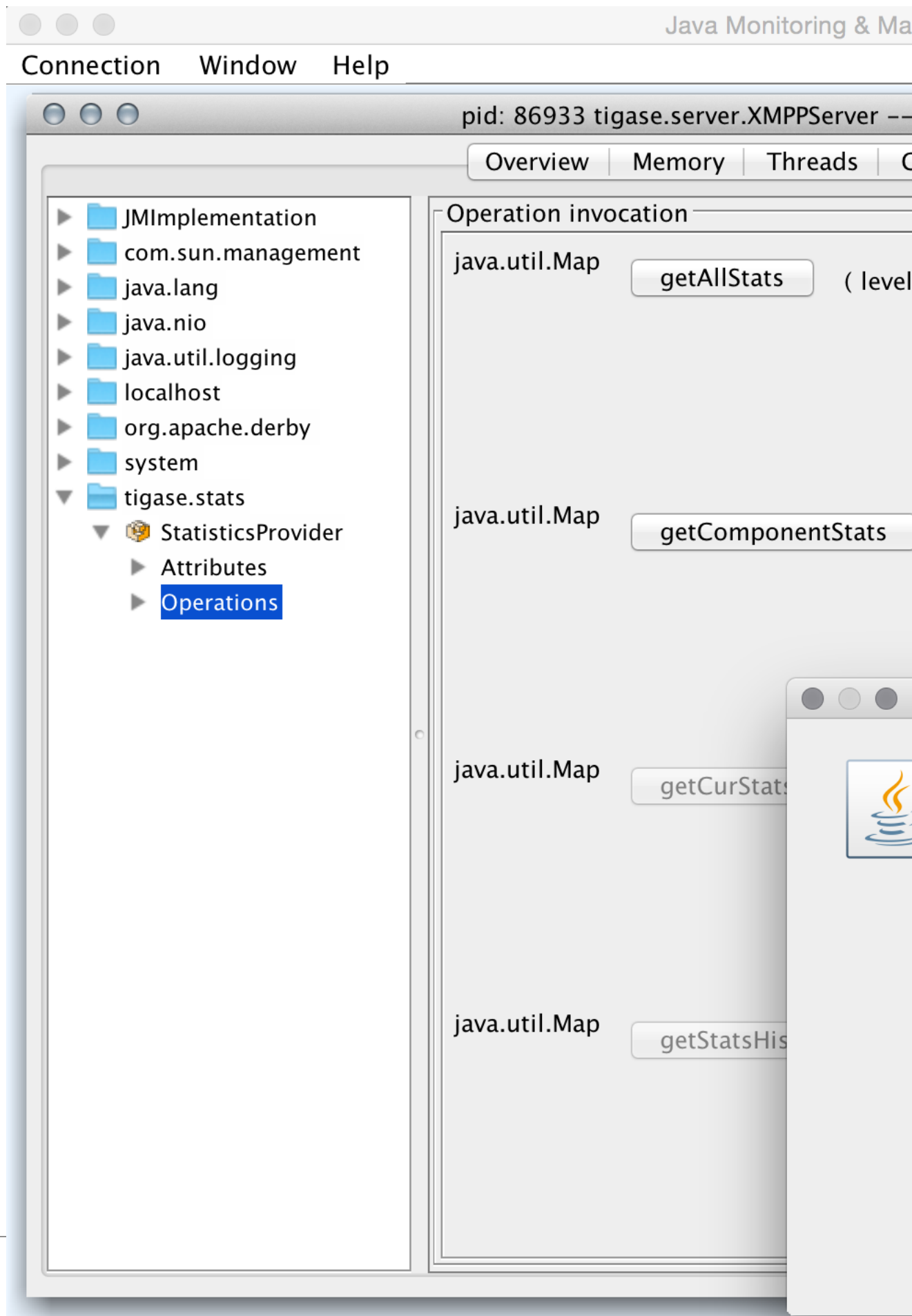
In order to access statistics over JMX we need to enable support for it in Tigase - Monitoring Activation. Afterwards we can use a number of tools to get to the statistics, for example the following:

### 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 the MBeans tab from where we can access the `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:



## 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 saves 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)

## Eventbus

Tigase includes an **eventbus** component to help with monitoring has been implemented. This allows you to set thresholds for certain predefined tasks and you or other JIDs can be sent a message when those thresholds are passed. You can even configure a mailer extension to have an E-mail sent to system administrators to let them know an event has occurred! Lets begin with setup and requirements.

Eventbus is based on a limited PubSub [<http://www.xmpp.org/extensions/xep-0060.html>] specification. Events are delivered to subscribers as a normal PubSub notification.

Each component or client may subscribe for specific types of events. Only components on cluster nodes are allowed to publish events.

## Setup

Eventbus is enabled by default on v7.1.0 b4001 and later, so no setup needed!

## How it Works

Events in Eventbus are identified by two elements: name of event and its namespace:

```
<EventName xmlns="tigase:demo">  
  <sample_value>1</sample_value>  
</EventName>
```

Where event name is `EventName` and namespace is `tigase:demo`.

Listeners may subscribe for a specific event or for all events with specific a namespace. Because in pubsub, only one node name exists, so we have to add a way to convert the event name and namespace to a node name:



`nodename = eventname + "-" + namespace`

So for example, to subscribe to `<EventName xmlns="tigase:demo">`, node must be: `EventName|tigase:demo`. If you wish to subscribe to all events with a specific namespace, use an asterisk (\*) instead of the event name: `*|tigase:demo`.

## Note

If client is subscribed to `*|tigase:demo` node, then events will not be sent from node `*|tigase:demo`, but from the **real** node (in this case: `EventName|tigase:demo`).

## Available Tasks

Eventbus monitoring has several pre-defined tasks that can be monitored and set to trigger. What follows is the list of tasks with the options attributed to each task.

- **disk-task** - Used to check disk usage. Available Options
  1. `enabled` - Enable or disable task, Boolean value.
  2. `period` - Period of running check, Integer value.
  3. `threshold` - Percentage of used space on disk, Float value.
- **cpu-temp-task** - Used to check CPU temperature. Available Options
  1. `enabled` - Enable or disable task, Boolean value.
  2. `period` - Period of running check, Integer value.
  3. `cpuTempThreshold` - Temperature threshold of CPU in °C.
- **load-checker-task** - Used to check system load. Available Options
  1. `enabled` - Enable or disable task, Boolean value.
  2. `period` - Period of running check, Integer value.
  3. `averageLoadThreshold` - Average percent load threshold, Long value.
- **memory-checker-task** - Used to check memory usage. Available Options
  1. `enabled` - Enable or disable task, Boolean value.
  2. `period` - Period of running check, Integer value.
  3. `maxHeapMemUsagePercentThreshold` - Alarm when percent of used Heap memory is larger than, Integer value.
  4. `maxNonHeapMemUsagePercentThreshold` - Alarm when percent of used Non Heap memory is larger than, Integer value.
- **logger-task** - Used to transmit log entries depending on level entered.
  1. `enabled` - Enable or disable task, Boolean value.
  2. `levelThreshold` - Minimal log level that will be the threshold. Possible values are SEVERE, WARNING, INFO, CONFIG, FINE, FINER, FINEST, and ALL.



- **connections-task** - Used to check users disconnections. **NOTE: The event will be generated only if both thresholds (amount and percentage) will be fulfilled.**

1. `enabled` - Enable or disable task, Boolean value.
2. `period` - Period of running check in ms, Integer value.
3. `thresholdMinimal` - Minimal amount of disconnected users required to generate alarm.
4. `threshold` - Minimal percent of disconnected users required to generate alarm.

## Configuration

Configuration of the eventbus monitor can be done one of two ways; either by lines in `config.tdsl` file, or by sending XMPP stanzas to the server. You may also send XMPP stanzas VIA HTTP REST. XMPP stanza configurations will override ones in `config.tdsl`, but they will only last until the server restarts.

### config.tdsl

Tasks can be configured in the `config.tdsl` file. See available tasks for the tasks that can be setup.

To enable a specific monitor task, use the following line:

```
monitor {
  - '$TASKNAME' {
    setting = value
  }
}
```

Where `monitor` is the component name for `MonitorComponent`, and `$TASKNAME` is one of the available task names.

This format will be the same for other settings for tasks, and it's best to group settings under one heading. For example:

```
monitor {
  - 'connections-task' {
    enabled = true
    period = 1000
  }
}
```

sets the check period to 1000 milliseconds and enables `connections-task`.

### Note

Once triggers have been activated, they will become dormant. Think of these as one-shot settings.

### Subscription Limitations

To define list of JIDs allowed to subscribe for events:

```
eventbus {
  affiliations {
    allowedSubscribers = - 'francisco@denmark.lit,bernardo@denmark.lit'
  }
}
```

If this is not specified, all users can subscribe.

## Configuration via XMPP

We can also configure the eventbus monitor component using XMPP stanzas. This allows us to set and change configurations during server runtime. This is done using a series of iq stanzas send to the monitor component.

We can query each component for its current settings using the following stanza.

```
<iq type="set" to="monitor@$DOMAIN/disk-task" id="aad0a">
  <command xmlns="http://jabber.org/protocol/commands" node="x-config"/>
</iq>
```

The server will return the component current settings which will make things easier if you wish to edit them. In this case, the server has returned the following to us

```
<iq from="monitor@$DOMAIN/disk-task" type="result" id="aad0a" to="alice@coffeebean">
  <command xmlns="http://jabber.org/protocol/commands" status="executing" node="x-config"
    sessionid="0dad3436-a029-4082-b0e0-04d838c6c0da">
    <x xmlns="jabber:x:data" type="form">
      <title>Task Configuration</title>
      <instructions/>
      <field type="boolean" label="Enabled" var="x-task#enabled">
        <value>0</value>
      </field>
      <field type="text-single" label="Period [ms]" var="x-task#period">
        <value>60000</value>
      </field>
      <field type="text-single" label="Disk usage ratio threshold" var="x-task#threshold">
        <value>0.8</value>
      </field>
    </x>
  </command>
</iq>
```

This tells us that the disk-task setting is not active, has a period of 60000ms, and will trigger when disk usage is over 80%.

To send new settings to the monitor component, we can send a similar stanza back to the monitor component.

```
<iq type="set" to="monitor@$DOMAIN/disk-task" id="aad1a">
  <command xmlns="http://jabber.org/protocol/commands" node="x-config"
    sessionid="0dad3436-a029-4082-b0e0-04d838c6c0da">
    <x xmlns="jabber:x:data" type="submit">
      <field type="boolean" var="x-task#enabled">
        <value>0</value>
      </field>
      <field type="text-single" var="x-task#period">
        <value>60000</value>
      </field>
      <field type="text-single" var="x-task#threshold">
        <value>0.8</value>
      </field>
    </x>
  </command>
</iq>
```

```
        </x>
      </command>
    </iq>
```

To which a successful update will give you an XMPP success stanza to let you know everything is set correctly.

Alternatively, you can update specific settings by editing a single field without adding anything else. For example, if we just wanted to turn the disk-task on we could send the following stanza:

```
<iq type="set" to="monitor@$HOSTNAME/disk-task" id="ab53a">
  <command xmlns="http://jabber.org/protocol/commands" node="x-config">
    <x xmlns="jabber:x:data" type="submit">
      <field type="boolean" var="x-task#enabled">
        <value>1</value>
      </field>
    </x>
  </command>
</iq>
```

To set any other values, do not forget that certain parts may need to be changed, specifically the `<field type="boolean" var=x-task#enabled">` fields:

- Your field type will be defined by the type of variable specified in the Available Tasks section.
- `var=x task#` will be followed by the property value taken directly from the Available Tasks section.

## Getting the Message

Without a place to send messages to, eventbus will just trigger and shut down. There are two different methods that eventbus can deliver alarm messages and relevant data; XMPP messages and using the mailer extension.

### XMPP notification

In order to retrieve notifications, a subscription to the `eventbus@tigase.org` user must be made. Keep in mind that subscriptions are not persistent across server restarts, or triggers. The eventbus schema is very similar to most XMPP subscription requests but with a few tweaks to differentiate it if you wanted to subscribe to a certain task or all of them. Each task is considered a node, and each node has the following pattern: `eventName|eventXMLNS`. Since each monitoring task has the `tigase:monitor:event` event XMLNS, we just need to pick the event name from the list of tasks. So like the above example, our event node for the disk task will be `disk-task|tigase:monitor:event`. Applied to an XMPP stanza, it will look something like this:

```
<iq type='set'
  to='eventbus@tigase.org'
  id='sub1'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <subscribe node='disk-taskEvent|tigase:monitor:event' jid='$USER_JID'/>
  </pubsub>
</iq>
```

Don't forget to replace `$USER_JID` with the bare JID of the user you want to receive those messages. You can even have them sent to a MUC or any component with a JID.

Available events are as follows:

- disk-taskEvent for disk-task
- LoggerMonitorEvent for logger-task
- HeapMemoryMonitorEvent for memory-checker-task
- LoadAverageMonitorEvent for load-checker-task
- CPUTempMonitorEvent for cpu-temp-task
- UsersDisconnected for connections-task

Alternatively, you can also subscribe to all events within the eventbus by using a wildcard \* in place of the event XMLNS like this example:

```
<iq type='set'
  to='eventbus@tigase.org'
  id='sub1'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <subscribe node='*|tigase:monitor:event' jid='$USER_JID'/>
  </pubsub>
</iq>
```

## Sample notification from Eventbus

```
<message from='eventbus.shakespeare.lit' to='francisco@denmark.lit' id='foo'>
  <event xmlns='http://jabber.org/protocol/pubsub#event'>
    <items node='EventName|tigase:demo'>
      <item>
        <EventName xmlns="tigase:demo" eventSource="samplecomponent.shakespeare.li
          <sample_value>1</sample_value>
        </EventName>
      </item>
    </items>
  </event>
</message>
```

## Mailer Extension

*Tigase Server Monitor Mailer Extension (TSMME)* can send messages from the monitor component to a specified E-mail address so system administrators who are not logged into the XMPP server.

For v7.1.0 versions and later, TSMME is already included in your distribution package and no extra installation is needed.

For versions older than 7.1.0 TSMME requires two files to operate:

- A compiled build of tigase mailer from its repository [<https://projects.tigase.org/projects/tigase-server-ext-mailer/repository>]. Place the compiled .jar file into jars/ directory.
- javax.mail.jar file which may be downloaded from this link [<http://java.net/projects/java-mail/downloads/download/javax.mail.jar>]. Also place this file in the jars/ directory.

## Configuration

Tigase Mailer Extension may be configured via the config.tds1 file in the following manner:

```
monitor {
  -'mailer-from-address' = -'sender@tigase.org'
  -'mailer-smtp-host' = -'mail.tigase.org'
  -'mailer-smtp-password' = -'*****'
  -'mailer-smtp-port' = -'587'
  -'mailer-smtp-username' = -'sender'
  -'mailer-to-addresses' = -'receiver@tigase.org,admin@tigase.org'
}
```

Here is an explanation of those variables.

- mailer-smtp-host - SMTP Server hostname.
- mailer-smtp-port - SMTP Server port.
- mailer-smtp-username - name of sender account.
- mailer-smtp-password - password of sender account.
- mailer-from-address - sender email address. It will be set in field from in email.
- mailer-to-addresses - comma separated notification receivers email addresses.

It is recommended to create a specific e-mail address in your mail server for this purpose only, as the account settings are stored in plaintext without encryption.

## Configuration of statistics loggers

It is possible to enable and configure automatic storage of statistics information. To do that you need to configure any of following statistics loggers as a `StatisticsCollector` component sub-beans:

`tigase.stats.CounterDataArchiveLogger` - every execution put current basic server metrics (CPU usage, memory usage, number of user connections, uptime) into database (overwrites previous entry).

`tigase.stats.CounterDataLogger` - every execution insert new row with new set of number of server statistics (CPU usage, memory usage, number of user connections per connector, number of processed packets of different types, uptime, etc) into the database.

`tigase.stats.CounterDataFileLogger` - every execution store all server statistics into separate file.

As an example to configure `tigase.stats.CounterDataFileLogger` to archive statistics data with level `FINE` every 60 seconds to file prefixed with `stat` and located in `logs/server_statistics` following entry is needed:

```
stats() {
  -'stats-file-logger' (class: tigase.stats.CounterDataFileLogger) {
    -'stats-directory' = -'logs/server_statistics'
    -'stats-filename' = -'stat'
    -'stats-unixtime' = false
    -'stats-datetime' = true
    -'stats-datetime-format' = -'HH:mm:ss'
    -'stats-level' = -'FINEST'
  }
}
```

```
}
```

## Server to Server Protocol Settings

Tigase server-to-server communication component facilitates communication with other XMPP servers (federation) and 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 optimal settings chosen. There are however, a set of configuration parameters you can adjust the server behavior to achieve optimal performance on 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.

Tigase server offers 2 different parameters to tweak the number of concurrent, s2s connections:

- `max-out-total-conns` - this property specifies the maximum outgoing connections the Tigase server opens to any remote XMPP server. This is a **per domain** limit, which means that this limit applies to each of the remote domains Tigase connects to. If it is set to 4 then Tigase opens a maximum of 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 to the `config.tdsl` file:

```
s2s {  
  - 'max-out-total-conns' = 2  
}
```

- `max-out-per-ip-conns` - this property specifies the maximum outgoing connections Tigase server opens to any remote XMPP server to its single IP address. This too, is **per domain** limit, which means that this limit applies to each of the remote domains Tigase connects to. If it is set to 1, and the above limit is set to 4, and the remote server is visible behind 1 IP address, then Tigase opens a maximum of 1 connection to `jabber.org` plus a maximum of 1 connection to `muc.jabber.org` and other subdomains.

To adjust the limit you have to add following line to the `config.tdsl` file:

```
s2s {
```

```
    - 'max-out-per-ip-conns' = 2  
  }
```

## Connection Throughput

Of course everybody wants his server to run with maximum throughput. This comes with a cost on resources, usually increased memory usage. This is especially important if you have large 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 `net-buff-high-throughput` or `net-buff-Standard` property descriptions.

## 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. 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' = 420L  
}
```

- `max-inactivity-time` - this parameters specifies the maximum s2s connection inactivity time before it is closed. If a connection is not in use for a long time, it doesn't make sense to keep it open and tie resources up. 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' = 900L  
}
```

## Custom Plugin: Selecting s2s Connection

Sometimes for 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 `config.tdsl` file:

```
s2s {
```

```
    - 's2s-conn-selector' = - 'YourSelectorImplementation'
}
```

The default selector picks connections randomly.

## skip-tls-hostnames

The `s2s-skip-tls-hostnames` property disables TLS handshaking for s2s connections to selected remote domains. Unfortunately some servers (certain versions of Openfire - [1 [http://community.igniterealtime.org/thread/36206]] or [2 [http://community.igniterealtime.org/thread/30578]]) have problems with TLS handshaking over s2s which prevents establishing a usable connection. This completely blocks any communication to these servers. As a workaround you can disable TLS for these domains to get communication back. Enabling this can be done on any vhost, but must be configured under the s2s component.

```
s2s {
    - 'skip-tls-hostnames' = [ - 'domain1', - 'domain2' - ]
}
```

## ejabberd-bug-workaround

This property activates a workaround for a bug in EJabberd in its s2s implementation. EJabberd does not send dialback in stream features after TLS handshaking even if the dialback is expected/needed. This results in unusable connection as EJabberd does not accept any packets on this connection either. The workaround is enabled by default right now until the EJabberd version without the bug is popular enough. A disadvantage of the workaround is that dialback is always performed even if the SSL certificate is fully trusted and in theory this dialback could be avoided. By default, this is not enabled.

```
s2s {
    dialback () {
        - 'ejabberd-bug-workaround' = true
        - }
    }
}
```

This replaces the old `--s2s-ejabberd-bug-workaround-active` property.

## Tigase Load Balancing

Tigase includes 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. The 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 the `SeeOtherHostIfc` interface.

Currently there are three strategies available:

- `SeeOtherHost` - most basic implementation returning either single host configured in `config.tds1` 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 a selection would be made is by default composed and reflects all connected nodes, alternatively hosts list can be configured in the `config.tdsl`;
- `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.
- `SeeOtherHostDualIP` - matches internal Tigase cluster nodes against the lookup table to provide relevant redirection hostname/IP (by default internal Tigase `tig_cluster_nodes` table will be used)

## Configuration Options

The most basic configuration is related to the choice of actual redirection implementation by declaring class for each connector:

```
bosh {
    seeOtherHost (class: <value>) {}
}
c2s {
    seeOtherHost (class: <value>) {}
}
ws2s {
    seeOtherHost (class: <value>) {}
}
```

Possible values are:

- `tigase.server.xmppclient.SeeOtherHost`
- `tigase.server.xmppclient.SeeOtherHostHashed`
- `tigase.server.xmppclient.SeeOtherHostDB`
- `tigase.server.xmppclient.SeeOtherHostDualIP`
- `none` - disables redirection

All options are configured on a per-connection-manager basis, thus all options need to be prefixed with the 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'
    - 'phases' = [ - 'OPEN', - 'LOGIN' - ]
  }
}
```

- `'default-host' = 'host1;host2;host3'` - a semicolon separated list of hosts to be used for redirection.
- `'phases' = [ ]` - an array 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.

## SeeOtherHostDB

For SeeOtherHostDB implementation there are additional options:

```
c2s {
  -'cm-see-other-host' {
    -'db-url' = -'jdbc:mysql://localhost/username?,password?'
    -'get-all-query-timeout' = -'10'
  }
}
```

- db-url - a JDBC connection URI which should be used to query redirect information; if not configured the default dataSource will be used;
- get-host-query - a SQL query which should return redirection hostname;
- get-all-data-query - a SQL helper query which should return all redirection data from database;
- get-all-query-timeout - allows to set timeout for executed queries.

## SeeOtherHostDualIP

This mechanisms matches internal Tigase cluster nodes against the lookup table to provide matching and relevant redirection hostname/IP. By default internal Tigase `tig_cluster_nodes` table is used (and appropriate repository implementation will be used).

To enable this redirection mechanism following configuration / class should be used. Note that for global use, all connection managers must have the same class defined. You can define each connection manager individually.

```
bosh {
  seeOtherHost (class: tigase.server.xmppclient.SeeOtherHostDualIP) {}
}
c2s {
  seeOtherHost (class: tigase.server.xmppclient.SeeOtherHostDualIP) {}
}
ws2s {
  seeOtherHost (class: tigase.server.xmppclient.SeeOtherHostDualIP) {}
}
```

It offers following configuration options:

- data-source - configuration of the source of redirection information - by default internal Tigase `tig_cluster_nodes` table will be used (and appropriate repository implementation will be used); alternatively it's possible to use `eventbus` source;
- db-url - a JDBC connection URI which should be used to query redirect information; if not configured `user-db-uri` will be used;
- get-all-data-query - a SQL helper query which should return all redirection data from database;

- `get-all-query-timeout` - allows to set timeout for executed queries;
- `fallback-redirection-host` - if there is no redirection information present (i.e. secondary host-name is not configured for the particular node) redirection won't be generated; with this it's possible to configure fallback redirection address.

All options are configured on per-component basis:

```
<connector> {
  -'cm-see-other-host' {
    -'data-source' = -'<class implementing tigase.server.xmppclient.SeeOtherHost>'
    -'db-url' = -'jdbc:<database>://<uri>'
    -'fallback-redirection-host' = -'<hostname>'
    -'get-all-data-query' = -'select * from tig_cluster_nodes'
    -'get-all-query-timeout' = 10
  }
}
```

## EventBus as a source of information

It's possible to utilize EventBus and internal Tigase events as a source of redirection data. In order to do that `eventbus-repository-notifications` needs to be enabled in `ClusterConnectionManager`:

```
'cl-comp' {
  -'eventbus-repository-notifications' = true
}
```

## Auxiliary setup options

### Enforcing redirection

It's possible to enforce redirection of connections on the particular port of connection manager with `force-redirect-to` set to Integer with the following general setting option:

```
<connection_manager> {
  connections {
    <listening_port> {
      -'force-redirect-to' = <destination_port>
    }
  }
}
```

for example, enable additional port 5322 for c2s connection manager and enforce all connections to be redirected to port 5222 (it will utilize hostname retrieved from `SeeOtherHost` implementation and will be only used when such value is returned):

```
c2s {
  connections {
    ports = [ 5222, 5322 -]
    5322 {
      -'force-redirect-to' = 5222
      socket = -'plain'
      type = -'accept'
    }
  }
}
```

```
}
```

## Configuring hostnames

To fully utilize SeeOtherHostDualIP setup in automated fashion it's now possible to provide both primary (*internal*) and secondary (*external*) hostname/IP (they need to be correct, `InetAddress.getByName( property );` will be used to verify correctness). It can be done via JVM properties `tigase-primary-address` and `tigase-secondary-address`. You can also utilize different implementation of DNS resolver by providing class implementing `tigase.util.DNSResolverIfc` interface as value to `resolver-class` property. Those properties can be set via `etc/tigase.conf` (uncommenting following lines, or manually exposing in environment):

```
DNS_RESOLVER=" --Dresolver-class=tigase.util.DNSResolverDefault -"
```

```
INTERNAL_IP=" --Dtigase-primary-address=hostname.local -"
```

```
EXTERNAL_IP=" --Dtigase-secondary-address=hostname -"
```

or in the `etc/config.tdsl` (they will be converted to JVM properties):

```
'dns-resolver' {  
  -'tigase-resolver-class' = -'tigase.util.DNSResolverDefault'  
  -'tigase-primary-address' = -'hostname.local'  
  -'tigase-secondary-address' = -'hostname'  
}
```

## External Component Configuration

Tigase can connect to external components, this guide will show you how this can be accomplished.

Configuration follows the same standards as all other components. It is also much more powerful as a single Tigase 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 more protocols can be supported or custom extensions to existing protocols can be added.

The implementation also supports a 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.

Pages in this guide describe in details all the administration aspects of setting up and managing external components.

- External Component Configuration
- Tigase as an External Component
- Load Balancing External Components in Cluster Mode

## External Component Configuration

As for all Tigase components you can load and configure external components via the `config.tdsl` file described in details in the DSL configuration section. This document describes how to enable 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` (class doesn't have to be specified when using default name).

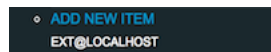
The following line in the `config.tds1` will load the component during the server startup time:

```
ext (class: tigase.server.ext.ComponentProtocol) {}
```

While this would load the component, without any additional configurations provided, the component would be practically useless. It is necessary to configure the virtual host domains of the external component during run-time via ad-hoc commands to make use of this component.

You may additionally configure the `bind-ext-hostnames` property.

To configure external component connections using Admin UI you need to open Admin UI web page (if you are logged in the same computer on which Tigase XMPP Server is running by default it should be available at `http://localhost:8080/admin/`). Then you should click on Configuration on the left side of the Admin UI web page and then select Add new item on `ext` component or by execution corresponding ad-hoc command on `ext` component using ad-hoc capable XMPP client, ie. Psi [`http://psi-im.org`].



You will be presented with a form which you should fill to configure external component connection details:

- *Domain name* - external component domain name (`muc.devel.tigase.org`)
- *Domain password* - password for authentication of the external component connection (`muc-pass`)
- *Connection type* - `accept` to make component wait for connection or `connect` force component to connect to the server (`connect`)

- *Port number* - port on which component should wait for connection or on which it try to connect (5270)
- *Remote host* - host to connect to (`devel.tigase.org`) (*may be left blank if component will only accept connections*)
- *Protocol* - id of protocol used for establishing connection
  - if connection type is `connect`:
    - XEP-0114: Jabber Component Protocol (`accept`) - for XEP-0114: Jabber Component Protocol [<https://xmpp.org/extensions/xep-0114.html>]
    - XEP-0225: Component Connections - for XEP-0225: Component Connections [<https://xmpp.org/extensions/xep-0225.html>]
  - if connection type is `accept`:
    - Autodetect - for automatic detection of protocol used by incoming connection (*recommended*)
    - XEP-0114: Jabber Component Protocol (`accept`) - for XEP-0114: Jabber Component Protocol [<https://xmpp.org/extensions/xep-0114.html>]
    - XEP-0225: Component Connections - for XEP-0225: Component Connections [<https://xmpp.org/extensions/xep-0225.html>]

Additional options may be left with defaults.

Later on if you would like to modify this values, you can do that using Admin UI by clicking on `Configuration` and `Remove an item` or `Update item configuration` at `ext` component or by execution corresponding ad-hoc commands on `ext` component using ad-hoc capable XMPP client, ie. Psi [<http://psi-im.org>].

## Tigase as an External Component

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's 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...

### Usage with shared database (since version 8.0.0)

When you are using Tigase server 8.0.0 or newer in the "external component mode" while using shared default "user repository" and you have main server also running Tigase XMPP Server 8.0.0 or newer, then you can benefit from the remote management of the component connections from the main server. To use that, you need to enable external component and external component manager on the main server by adding following line to the config file:

```
'ext' () {}  
'ext-man' () {}
```

With that in place you can use Admin UI or ad-hoc commands available at ext-man component of the main server to configure connection details of the servers running in the component mode.

In Admin UI you click on Configuration section and select Add new item at the ext-man component, which will present you with a following form to fill in external component connectivity details:

## 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 Tigase as an external component:

```
'config-type' = -'component'
```

Knowing that we can now create simple configuration file for Tigase XMPP Server:

```
admins = [ -'admin@devel.tigase.org' -]
'config-type' = -'component'
debug = [ -'server' -]
'virtual-hosts' = [ -'devel.tigase.org' -]
dataSource {
    default () {
        uri = -'master_server_default_database_url'
    }
}
```

```

}
userRepository {
    default () {}
}
authRepository {
    default () {}
}
}
muc (class: tigase.muc.MUCComponent) {}
ext () {
}

```

where `master_server_default_database_url` is the same URL as the one used on the main server for default data source.

With that in place we can use ad-hoc commands or Admin UI on the main server to configure Tigase XMPP Server to accept external component connections and to connect from the external component to the master server.

### Adding external component connection settings to the manager (ext-man) using Admin

The screenshot shows the Tigase Admin UI. On the left is a sidebar with a 'CONFIGURATION' menu. The main area displays a form for configuring an external component connection. The form fields are as follows:

Field	Value
Domain name	muc.devel.tigase.org
Domain password	muc-pass
Connection type	connect
Port number	5270
Remote host	devel.tigase.org
Protocol	accept
Load balancer class	tigase.server.ext.lb.ReceiverBareJidLB
(Optional) Routings	
(Optional) Socket type	plain
Owner	admin@localhost
Administrators	

At the bottom of the form is a button labeled 'Prześlij' (Submit).

UI.

You need to pass:

- Domain name - external component domain name (`muc.devel.tigase.org`)
- Domain password - password for authentication of the external component connection (`muc-pass`)
- Connection type - `accept` to make component wait for connection or `connect` force component to connect to the server (`connect`)
- Port number - port on which component should wait for connection or on which it try to connect (`5270`)
- Remote host - host to connect to (`devel.tigase.org`)
- Protocol - id of protocol used for establishing connection



- XEP-0114: Jabber Component Protocol (accept) - establish connection using XEP-0114: Jabber Component Protocol [<https://xmpp.org/extensions/xep-0114.html>]
- XEP-0225: Component Connections - establish connection using XEP-0225: Component Connections [<https://xmpp.org/extensions/xep-0225.html>]

Additional options may be left with defaults.

## More Components

Suppose you want to run more than one component as an external components within one Tigase instance. Let's add another - PubSub component to the configuration above and see how to set it up.

The most straightforward way is just to add another component to the server running in the component mode for the component domain

```
admins = [ -'admin@devel.tigase.org' -]
'config-type' = -'component'
debug = [ -'server' -]
'virtual-hosts' = [ -'devel.tigase.org' -]
dataSource {
    default () {
        uri = -'jdbc:derby:/tigasedb'
    }
}
userRepository {
    default () {}
}
authRepository {
    default () {}
}
muc (class: tigase.muc.MUCComponent) {}
pubsub (class: tigase.pubsub.PubSubComponent) {}
ext () {}
```

and then to add new connection domain to the main server external component settings and to the external component manager settings. You basically do the same thing as you did while adding only MUC component as the external component.

Please note however that we are opening two connections to the same server. This can waste resources and over-complicate the system. For example, what if we want to run even more components? Opening a separate connection for each component is a tad overkill.

In fact there is a way to reuse the same 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. Since you are reusing connections (hostname binding is defined in XEP-0225 [<http://xmpp.org/extensions/xep-0225.html>] only), you must use this protocol for the functionality.

Here is an example configuration with a single connection over the XEP-0225 [<http://xmpp.org/extensions/xep-0225.html>] protocol used by both external domains:

```
admins = [ -'admin@devel.tigase.org' -]
```

```
'bind-ext-hostnames' = [ -'pubsub.devel.tigase.org' -]
'config-type' = -'component'
debug = [ -'server' -]
'virtual-hosts' = [ -'devel.tigase.org' -]
dataSource {
    default () {
        uri = -'jdbc:derby:/tigasedb'
    }
}
ext () {
}
userRepository {
    default () {}
}
authRepository {
    default () {}
}
}
muc (class: tigase.muc.MUCComponent) {}
pubsub (class: tigase.pubsub.PubSubComponent) {}
```

With this configuration you do not need to configure entries in `ext-man` for PubSub component, only for MUC component but you need to use `client` as the value for protocol field.

## Usage with a separate database

### 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 Tigase as an external component:

```
'config-type' = -'component'
```

This generates a configuration for Tigase with only one component loaded by default - the component used for external component connection. If you use this configuration type, your `config.tdsl` file may look like this:

```
admins = [ -'admin@devel.tigase.org' -]
'config-type' = -'component'
debug = [ -'server' -]
'virtual-hosts' = [ -'devel.tigase.org' -]
dataSource {
    default () {
        uri = -'jdbc:derby:/tigasedb'
    }
}
```

```
userRepository {
    default () {}
}
authRepository {
    default () {}
}
muc (class: tigase.muc.MUCComponent) {}
ext () {
}
```

To make this new instance connect to the Tigase XMPP Server, you need to create one more file with external connection configuration at `etc/externalComponentItems` which will be loaded to the local database and then removed.

```
muc.devel.tigase.org:muc-pass:connect:5270:devel.tigase.org:accept
```

## Warning

While loading configuration from `etc/externalComponentItems` file is supported, we recommend usage of shared database if possible. In future this method may be deprecated.

## More Components

Suppose you want to run more than one component as an external components within one Tigase instance. Let's add another - PubSub component to the configuration above and see how to set it up.

The most straightforward way is just to add another external component connection to the main server for the component domain using Admin UI or ad-hoc command on the main server.

Then we can use following configuration on the server running in the component mode:

```
admins = [ -'admin@devel.tigase.org' -]
'config-type' = -'component'
debug = [ -'server' -]
'virtual-hosts' = [ -'devel.tigase.org' -]
dataSource {
    default () {
        uri = -'jdbc:derby:/tigasedb'
    }
}
userRepository {
    default () {}
}
authRepository {
    default () {}
}
muc (class: tigase.muc.MUCComponent) {}
pubsub (class: tigase.pubsub.PubSubComponent) {}
ext () {
}
```

and we need to create a file with configuration for external component connection which will be loaded to the internal database:

```
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 can waste resources and over-complicate the system. For example, what if we want to run even more components? Opening a separate connection for each component is a tad overkill.

In fact there is a way to reuse the same 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. Since you are reusing connections (hostname binding is defined in XEP-0225 [<http://xmpp.org/extensions/xep-0225.html>] only), you must use this protocol for the functionality.

Here is an example configuration with a single connection over the XEP-0225 [<http://xmpp.org/extensions/xep-0225.html>] protocol used by both external domains:

```
admins = [ -'admin@devel.tigase.org' -]
'bind-ext-hostnames' = [ -'pubsub.devel.tigase.org' -]
'config-type' = -'component'
debug = [ -'server' -]
'virtual-hosts' = [ -'devel.tigase.org' -]
dataSource {
    default () {
        uri = -'jdbc:derby:/tigasedb'
    }
}
ext () {
}
userRepository {
    default () {}
}
authRepository {
    default () {}
}
}
muc (class: tigase.muc.MUCComponent) {}
pubsub (class: tigase.pubsub.PubSubComponent) {}
```

and example of the external connections configuration file:

```
muc.devel.tigase.org:muc-pass:connect:5270:devel.tigase.org:client
```

## Load Balancing External Components in Cluster Mode

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 8.0.0 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 components work as an external component and make it work in a cluster mode, even if the component itself does not support cluster mode.

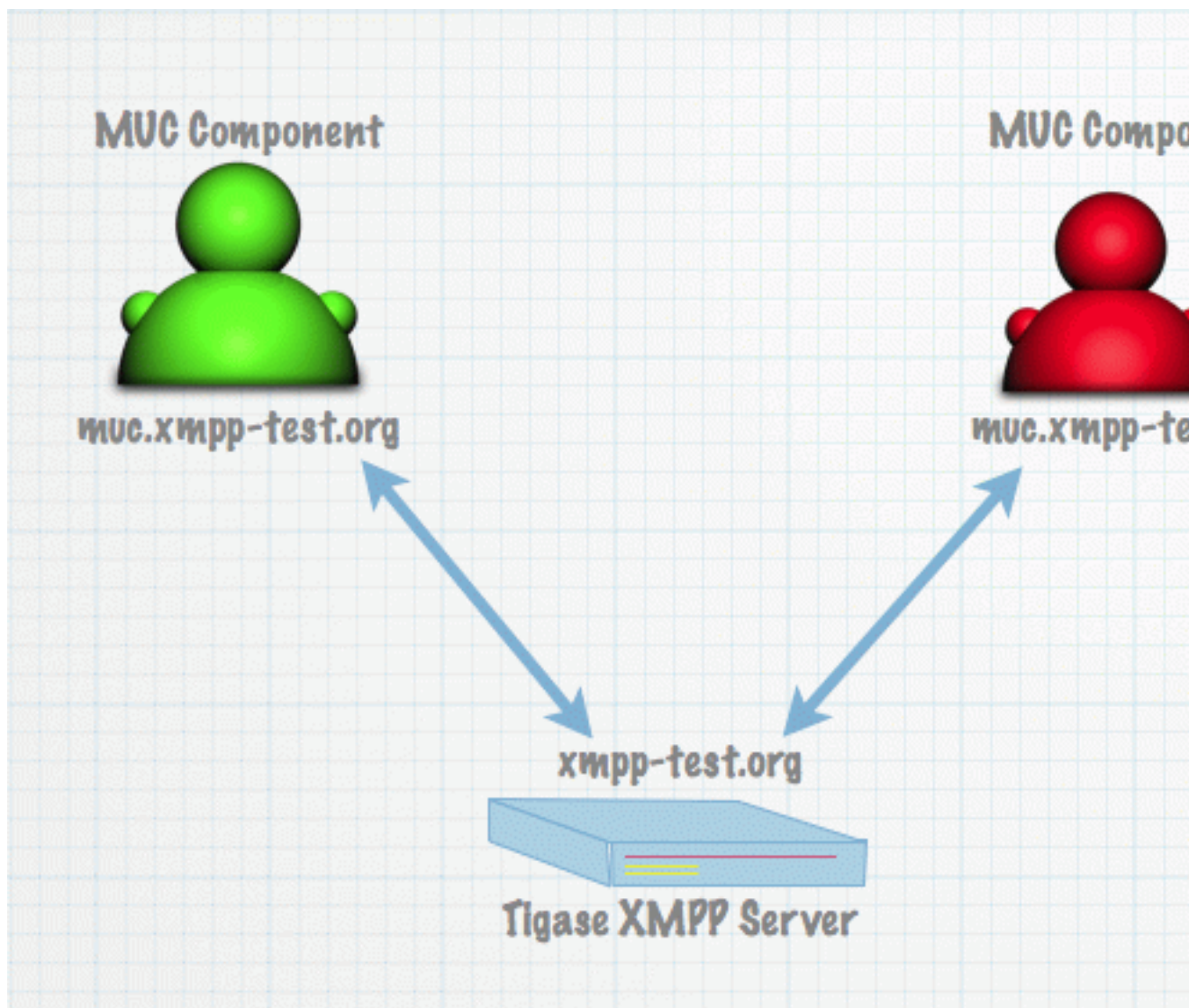
Here are step by step instructions and configuration examples teaching how to achieve both goals.

## Load Balancing External Component

The first, and most simple scenario is to connect multiple instances of an external component to a single Tigase XMPP Server to distribute load.

There are at least 2 reasons why this would be an optimal solution: 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 Tigase.

On the server side we will enable `ComponentProtocol` component as we need to do to enable external component without clustering support.

Then using Admin UI we will add a new external component connection settings using `Add item` position for `ext` component in `Configuration` section of the web page just as it is described in `External Component Configuration` section.

The screenshot shows the Tigase Admin UI configuration page. On the left is a sidebar with a 'CONFIGURATION' header and a list of items. The main area is a form for configuring an external component. The fields are as follows:

- Domain name: `muc.devel.tigase.org`
- Domain password: `muc-pass`
- Connection type: `accept` (dropdown)
- Port number: `5270`
- Remote host: (empty)
- Protocol: (empty)
- Load balancer class: `tigase.server.ext.lb.ReceiverBareJidLB`
- (Optional) Routings: (empty)
- (Optional) Socket type: `plain` (dropdown)
- Owner: `admin@localhost`
- Administrators: (empty)

At the bottom left of the form is a button labeled 'Prześlij'.

The only change here is that we will specify value for field `Load balancer class` and we will use `ReceiverBareJidLB` as a value.

The configuration for both instances of the MUC component (identical for both of them) can be done in the same way as it is done for a single instance of the MUC component. There is nothing to change here.

The difference is one small element in the server configuration. At the value of `Load balancer class` field in `Add item` form is set to **`ReceiverBareJidLB`**.

This is the 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 makes 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 if you use plugin with class name: **`SenderBareJidLB`**.

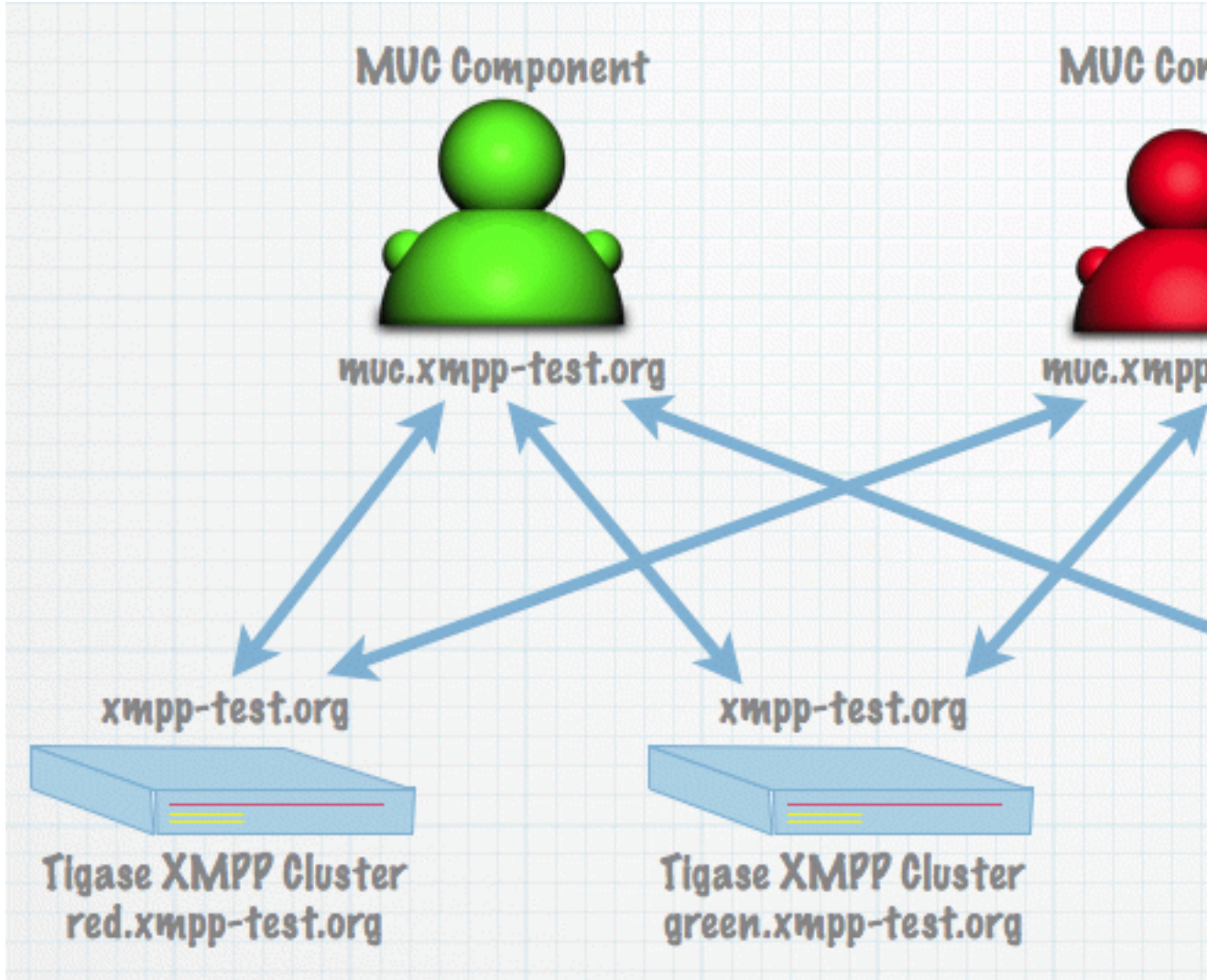
These are two basic load distribution strategies available now. For some use cases none of them is good enough. If you have PubSub, then you probably want to distribute load based on the PubSub node. There is no plugin for that yet but it is easy enough to write one and put the class name in configuration.

## 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. To do so, we are configuring Tigase XMPP Server to run in the cluster mode and on each of cluster nodes we need to enable `ComponentProtocol` component.

This can be simply done by adding following line to the server configuration file:

```
ext () {}
```

After this is done we need to add a new external component connection settings using `Add item` position for `ext` component in `Configuration` section of the web page just as it is described in `External Component Configuration` section.

As you can see there is nothing special here. The most interesting part comes on the MUC side, but it is only a very small change from the configuration of the component to use with single node Tigase XMPP Server installation.

When you are adding/configuring external component settings using Admin UI (Add item or Update item configuration for ext-man component) or using separate configuration file (when you are not using shared database) then you need to pass as a value for Remote host field a semicolon separated list of all of the cluster nodes to which external component should connect.

In our case it would be:

```
red.xmpp-test.org;green.xmpp-test.org;blue.xmpp-test.org
```

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.

## Client to Server Communication

Client to server communication is an integral part of XMPP communication. C2S handles all client communication to the server, and is responsible for filtering and handling remote communications. C2S CAN be disabled, however doing so will only allow communication of internal components, and S2S communications.

## Configuration

To disable C2S, use the following line in config.tdsl folder.

```
c2s (active: false) {}
```

Otherwise, C2S component is activated by default.

## Connections

The connections container houses all configuration related to connections with the component. Each port may be configured individually.

```
c2s {
  connections {
    5222 {
      <configuration>
    }
    5080 {
      <configuration>
    }
  }
}
```

## new-connections-throttling

The property allows you to limit how many new users' connection per second the server accepts on a particular port. Connections established within the limit are processed normally, all others are simply disconnected. This allows you to avoid server overload in case there is a huge number of users trying to connect at the same time. Mostly this happens after a server restart.



```
c2s {
  connections {
    5222 {
      -'new-connections-throttling' = 150L
    }
  }
}
```

Here, this limits the number to 150 connections per second before connection attempts are dropped.

This replaces the old `--new-connections-throttling` property.

## Resumption timeout

It is now possible to set a default stream resumption timeout that the server uses. This allows control of how long a server will wait for a reconnection from a client. This can be particularly helpful to manage mobile clients connecting to your server as they may not have complete coverage, and you do not want to close the stream right away. By default, Tigase sets this value to 60 seconds.

```
c2s {
  -'urn:xmpp:sm:3' {
    -'resumption-timeout' = 90
  }
}
```

This sets the default timeout to 90 seconds. You may, if you choose, specify a maximum timeout time, which will allow the server to wait between the default and maximum before a connection is closed.

```
c2s {
  -'urn:xmpp:sm:3' {
    -'max-resumption-timeout' = 900
  }
}
```

### Note

If the `max-resumption-timeout` is not set, it will always equal the `resumption-timeout` number, or default is none is set.

Available since v7.1.0

## Packet Redelivery

Normally packets are handled by C2S and are typically processed in the first run, however if that fails to send, a retry of sending that packet will occur after 60 seconds. If that second try fails, the delay will increase by a factor of 1.5. This means that the next retry will occur at 90, 135, and so on until the retry count is reached. By default this count is 15, however it can be changed by using the following setting:

```
c2s {
  -'packet-deliver-retry-count' = -'20'
}
```

This setting prevents packet redelivery attempts from continuing into infinity (or when the host machine runs out of memory).

# Tigase External Service Discovery

Welcome to the Tigase External Service Discovery component user guide. Component provides support for XEP-0215: External Service Discovery [<http://xmpp.org/extensions/xep-0215.html>] which allows discovery of external services which are not accessible using XMPP protocol.

## Setup & Configuration

Component (which is implemented in class `tigase.server.extdisco.ExternalServiceDiscoveryComponent`) is by default registered under name `ext-disco` and disabled. To enable it you need to enable it in configuration. Example:

- in DSL format:

```
ext-disco () { -}
```

Additionally you need to activate `urn:xmpp:extdisco:2` XMPP processor in `SessionManager` by:

- in DSL - enable subbean of `sess-man`:

```
sess-man {  
  - 'urn:xmpp:extdisco:2' () {}  
}
```

List of external services returned by server is configurable using ad-hoc commands provided for this component. AdHoc commands are accessible only for server administrator using XMPP client with support for AdHoc commands or using Tigase Admin UI. Usage of AdHoc commands provides easiest and flexible way to add, modify or remove entries for services which will be returned by discovery.

---

# Chapter 11. Using Tigase

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

- Tigase Log Guide
- Debugging Tigase
- Basic System Checks
- Add and Manage Domains
- Presence Forwarding
- Tigase and PyMSN Transport
- Multiple Session Managers
- Watchdog
- Tips And Tricks
  1. Runtime Environment Tip
  2. Checking Cluster Connections
  3. Best Practices for Connecting to Tigase XMPP server From Web Browser
- Command Line Tools
  1. Configuration Management Tool
- Scripting Support in Tigase
  1. Scripting Introduction - Hello World!
  2. Tigase Scripting Version 4.4.x Update for Administrators
  3. Tigase and Python Scripting
- Configuration Wizards

## Offline Messages

Tigase like any XMPP server supports storing of messages for users who are offline so that they may receive messages sent to them while they were not logged in.

By default, Tigase `MessageAmp` processor is responsible for storing offline messages, and will automatically store offline messages. This guide has multiple sections for setting limits globally, per user, and others.

Many of the features listed here require the use of the Advanced Message Processor Plugin which is turned on by default. To ensure AMP is turned on your system, view your `config.tds1` file and be sure the following is there in your plugins line:

```
'sess-man' {  
    amp ( ) { }
```

```
}
```

Messages will be delivered to intended recipients when they first login after roster exchange.

## Offline Message Limits

Support for limiting number of stored offline messages on a per-user basis has now been added to Tigase as of v7.1.0. By default, Tigase comes with a limit of stored offline messages which is set for every user. This limit by default is 100 offline messages for barejid-barejid pair. This value can be changed by the `store-limit` property. To change to 200 messages on barejid-barejid paid, add the following entries to the `config.tds1` file:

```
amp {
  -'store-limit' = 200L
}
'sess-man' {
  amp () {
    -'store-limit' = 200L
  }
}
```

This setting applies to every user.

## User Limit

Each user is able to configure the number of offline messages which should be stored for him. To enable this feature, the following lines need to be entered into the `config.tds1` file:

```
amp {
  -'user-store-limit-enable' = true
}
'sess-man' {
  amp () {
    -'user-store-limit-enable' = true
  }
}
```

Values of user-specific limits will be stored in `UserRepository` under subnode of `offline-msgs` and key `store-limit`. Data storage will be stored in `tig_pairs` key with the value and a proper record from `tig_nodes` points to this record.

## Handling of Offline Messages Exceeding Limits

There are two possible ways to handle offline messages that exceed the limitations: . error sending message with error type back to sender. . drop drop of message without notifications to sender.

By default, Tigase sends a message back to the original sender with an error type of `service-unavailable` with a proper description of error according to XEP-0160 [<http://www.xmpp.org/extensions/xep-0160.html>]. However, it is possible to change this behavior to better suit your needs. This is done by adding the following line to your `config.tds1` file.

```
'sess-man' {
  amp () {
    -'quota-exceeded' = -'drop'
  }
}
```

This will force Tigase to drop packets that exceed the offline message limit.

## Setting the Limits by User

Users wishing to set a custom limit of stored offline messages for barejid-barejid pairs needs to send the following XMPP stanza to the server:

```
<iq type="set" id="{random-id}">
  <msgoffline xmlns="msgoffline" limit="{limit}"/>
</iq>
```

Where: `{random-id}` is a random ID of the stanza (can be any string). `{limit}` is the integer value of the offline message limit. This can be set to `false` to disable offline message limits.

In response, the server will send back an `iq` stanza with a result type:

```
<iq type="result" id="{random-id}">
  <msgoffline xmlns="msgoffline" limit="{limit}"/>
</iq>
```

## Example of Setting Limit of Stored Offline Messages to 10

XMPP client sends the following to the server:

```
<iq type="set" id="aabba">
  <msgoffline xmlns="msgoffline" limit="10"/>
</iq>
```

Server response:

```
<iq type="result" id="aabba">
  <msgoffline xmlns="msgoffline" limit="10"/>
</iq>
```

## Example of Disabling Offline Message Limit

XMPP client sends the following to the server:

```
<iq type="set" id="aabbb">
  <msgoffline xmlns="msgoffline" limit="false"/>
</iq>
```

Server response:

```
<iq type="result" id="aabbb">
  <msgoffline xmlns="msgoffline" limit="false"/>
</iq>
```

## Storing offline messages without body content

Tigase can now store offline messages without `<body/>` content.

See XEP-0334 [<http://xmpp.org/extensions/xep-0334.html>] for protocol details.

This can include message receipts, and messages with specific `do-not-store` tags.

Support has been added to set a list of paths and `xmlns` to trigger and place storage of offline messages using the following settings in `config.tds1`:

```
'sess-man' {  
  amp () {  
    -'msg-store-offline-paths' = [ -'/message/received[urn:xmpp:receipts]', -'  
  -}  
}
```

This example results in two settings:

/message/received[urn:xmpp:receipts]	Results in storage of messages with a received subelement and with the xmlns set to urn:xmpp:receipts
/message/store-offline	Results in storing messages with a store-offline subelement without checking xmlns.

## Filtering of offline storage

It is possible to set storage of other types to save:

```
'sess-man' {  
  amp () {  
    -'msg-store-offline-paths' = [ -'/message/store-offline', -'/message/do-n  
  -}  
}
```

The above setting in the `config.tds1` file will cause that:

- messages with `<store-offline>` subelement will be stored without checking for associated xmlns.
- messages with `<do-not-store>` element **will not** be saved.

Any of these can be adjusted for your installation, remember that a '-' will stop storage of messages with the indicated property. Messages will be checked by these matchers and if any of them result in a positive they will override default settings.

For example, if you wanted to store messages with `<received>` element, but not ones with `<plain>` element, your filter will look like this:

```
'sess-man' {  
  amp () {  
    -'msg-store-offline-paths' = [ -'/message/received', -'/message/plain' -]  
  -}  
}
```

However....

### Note

**THE ABOVE STATEMENT WILL NOT WORK** As it will just store all messages with `<received>` subelement.

The below statement will properly filter your results.

```
'sess-man' {  
  amp () {  
    -'msg-store-offline-paths' = [ -'/message/plain', -'/message/received' -]  
  -}  
}
```

Filtering logic is done in order from left to right. Matches on the first statement will ignore or override matches listed afterwards.

## Disabling Offline Messages

If you wish to disable the storing of offline messages, use the following line in your `config.tdsl` file. This will not disable other features of the AMP plugin.

```
'sess-man' {
  amp () {
    msgoffline (active: false) {}
  }
}
```

## Last Activity

Tigase XMPP Server supports XEP-0012: Last Activity [<https://xmpp.org/extensions/xep-0012.html>] extension, which allows retrieval information when particular contact was active last time. It's not enabled by default.

The functionality itself is split in two plugins:

- `jabber:iq:last-marker` - responsible for updating information about last activity of user
- `jabber:iq:last` - responsible for handling requests to retrieve last activity information (it depends on `jabber:iq:last-marker` plugin).

In order to enable functionality you should add both plugins to your configuration file

```
'sess-man' {
  -'jabber:iq:last-marker' (active: true) {
    'jabber:iq:last' (active: true) {}
  }
}
```

## What updates last activity

By default marker plugin will only update last activity information on presence stanza. It's possible to control whether `<presence/>` and/or `<message/>` should update with respective options:

```
'sess-man' {
  -'jabber:iq:last-marker' (active: true) {
    message = true
    presence = true
  }
}
```

Those settings will cause updating last activity information for both `<message/>` and `<presence/>` stanzas

## Persist everything to repository

To lower impact on performance, by default last activity information is persisted to repository less frequently. This can yield slightly less accurate results on installations with multiple cluster nodes with users having multiple resources connected. To get more accurate results you should set `persistAll-`

ToRepository to true, which will cause all update times to be persisted (please bear in mind that this could cause higher impact on the repository).

```
'sess-man' {  
  -'jabber:iq:last-marker' (active: true) {  
    persistAllToRepository = true  
  -}  
}
```

## Tigase Log Guide

Tigase has multiple levels of logging available to help provide targeted and detailed information on processes, components, or traffic. In these documents we will look at where tigase generates logs, what they contain, and how we can customize them to our needs.

### install.log

This log file is a basic list of files that are made on install of Tigase server. Although you may not need to use it, it can provide a handy list to see if any files were not written to your hard drive upon installation.

### derby.log

If you are using the derby database installed with Tigase, this is the startup log for the database itself. Issues that might be related to the database, can be found in this file. Typically, if everything works okay, it's a very small file with only 10 lines. It is overwritten on startup of the database.

### etc/config-dump.properties

The config-dump.properties is dump file of all your properties listed for every option within Tigase and components. The structure of the log lines is the same as the structure of Tigase XMPP Server config file - TDSL. Lets take the value for admins, listing who is administrator for the server.

```
admins = [ -'admin@jabber.freehost.org', -'administrator@jabber.freehost.org', -'f
```

The admin parameter which is an array of strings and has 3 users listed.

This file is re-written every time tigase starts.

### logs/tigase.log.#

The tigase.log files are where the majority of logging will take place. The rules for writing to these logs can be manipulated by editing files in the int.properties file. To see how, see the Debugging Tigase section of this manual for more details about how to turn on debug logging, and how to manipulate log settings. Entries to these logs are made in the following format:

```
2015-08-10 13:09:41.504 [main]          Sctipr.init()          INFO: Initilized script
```

The format of these logs is below: <timestamp> <thread\_name> <class>.<method>  
<log\_level>: <message> <thread\_name>. This can vary - for components it would be  
<direction>\_<int>\_<component name>, for plugins it will just be the plugin name.

Let's look at another example from the log file.

```
2015-08-10 12:31:40.893 [in_14_muc] InMemoryMucRepository.createNewRoom() FINE:
```



The process ID may sometimes come in a different format such as `[in_14-muc]` which specifies the component (muc) along with the process thread identifier (14). As you can see, the format otherwise is nearly identical.

`tigase.log.#` files are *rotated* - this means that server begins writing to `tigase.log.0` when it is first run, and continues to dump information until the log size limit is hit. At this point, Tigase renames `tigase.log.0` as `tigase.log.1`. A new `tigase.log.0` will be created, and Tigase will begin logging to this file. When this file is full, `tigase.log.1` will be renamed `tigase.log.2` and `tigase.log.0` will be renamed `tigase.log.1`. Using this scheme, `tigase.log.0` will **always** be your most recent log.

By default, Tigase has a limit of 10000000 bytes or 10MB with a file rotation of 10 files. You can edit these values by editing the `config.tds1` file and adding the following lines.

```
logging {
    java.util.logging.FileHandler {
        count = -'15'
        limit = -'20000000'
    }
}
```

This code, if entered into the `config.tds1` file increases the size of the files to 15, and enlarges the maximum size to 20MB. Note the larger the collective log space is, the larger number of sectors on hard disk are active. Large log blocks may impact system performance.

*You may see a `tigase.log.0.lck` file in the directory while the server is running. This is a temporary file only and is deleted once Tigase is cleanly shut down.*

## logs/statistics.log.#

Statistics log will duplicate any information that is related to sending of statistics to Tigase if you are using an unlicensed copy of Tigase XMPP server. Mainly it will consist output of LicenceChecker. The numbering logic will be the same as `tigase.log.#` files.

## logs/tigase.pid

`tigase.pid` is a file that just contains the Process ID or PID for the current run of Tigase. It is only valid for the current or most recent run cycle and is overwritten every time Tigase starts.

## logs/tigase-console.log

### Important

This is the most important log file containing the most essential information related to operation of the Tigase XMPP Server. Any errors or exceptions in this file indicate with high probability serious issues with server operation.

This file contains information related to Tigase's running environment, and is a dump from the server itself on what is being loaded, when, and if any issues are encountered. It will start by loading Java classes (consequently making sure the Java environment is present and functioning). Then it will begin loading the configuration file, and adding default values to settings that have not been customized. You can then see all the components being loaded, and settings added where default values are needed. Lastly you will see a log of any plugins that are loaded, and any parameters therein. You may see tags such as INFO or WARNING in the logs. Although they may contain important information, the program will continue to operate as normal are not of too great concern.

ERROR flags are issues you will want to pay attention as they may list problems that prevent Tigase or components from properly functioning.

### Note

Windows does not create this file, rather the output is shown in the command line and is not dumped to a file.

If Tigase is gracefully shut down, `tigase-console.log` will add statistics from the server's operation life in the following format.

```
component/statistic = value
```

*Any component that may have a statistic, whether used or not, will place a value here*

This file can be handy if you are tracking issues in the server.

`tigase-console.log` is appended during each run session of the server.

## Log File Location

You can also change the location of log files if you have a specific directory you wish to use. The configuration may be made by the following lines in your `config.tdsl` file:

```
logging {
    java.util.logging.FileHandler {
        pattern = -'/var/log/tigase/tigase.log'
    -}
}
```

This setting changes the log file location to `/var/log/tigase/` where all log files will be made. Files in the original location will be left.

## Debugging Tigase

If something goes wrong and you can't find out why it is not working as expected, you might want more detailed debugging options switched on.

Tigase is a Java application and it uses Java logging library, this gives you the flexibility to switch logging on for selected Java packages or even for a single Java class.

Logs files are stored in `logs/` directory. `tigase-console.log` stores basic log data, but is the main log file. `tigase.log.N` files keep all the detailed logging entries. So this is the place where you should look in case of problems.

## Configuration

By default, Tigase has the old `debug = [ 'server' ]` setting is turned on and does not need to be added.

However, 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' -]
```

You can also have debugging switched on for many packages/classes at the same time:

```
debug = [ -'cluster' -, -'xmpp.XMPPIService' -]
```

Other packages you might be interested in are:

- `io` can print out what is going on a very low level network level including TLS/SSL stuff.
- `xml` would print the XML parser debugging data.
- `cluster` would print all the clustering related stuff.
- `xmpp.impl` would print logs from all plugins loaded to Tigase server.

## Non-Tigase packages

To enable logging for your own packages from those different than Tigase, you have to use another option which has been made available for this:

```
debug-packages = [ your.com.package -]
```

## Basic System Checks

Previously, a configuration article is available about Linux settings for high load systems. This has a description of basic settings which are essential to successfully run XMPP service for hundreds or thousands of 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 DDOS 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.*

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 time consuming and it is very easy to forget about.

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.

Inside the Tigase server scripts/ [<https://tigase.tech/projects/tigase-server/repository/revisions/master/show/scripts>] repository find 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://tigase.tech/projects/tigase-server/repository/revisions/master/show/scripts/machine-check.sh>] and run for yourself.

Any comments or suggestions, as usual, are very much appreciated.

## Add and Manage Domains (VHosts)

Tigase XMPP Server offers an easy to use and very flexible way to add and manage domains hosted on installation (vhosts).

There are two ways of managing domains you host on your server:

- using web-based admin management console - Admin UI
- using XMPP ad-hoc commands by XMPP client, ie. Psi [<http://psi-im.org/>]

## Note

To use any of those ways, you need to be an administrator of the server, which means that you have a XMPP account created on this XMPP server and your account JID is added to the list of the administrators in the Tigase XMPP Server configuration file.

## Using Admin UI

First, you need to open Admin UI web page. By default Admin UI is enabled and available at the port 8080 at path `/admin/` on the XMPP server. Assuming that you are logged on the same machine which hosts Tigase XMPP Server, it will be available at `http://localhost:8080/admin/`.

When you will be prompted for username and password to login to the Admin UI please fill username with full JID of your XMPP admin account and fill password field with password for this account. When you submit correct credentials you will get access to the Admin UI and Tigase XMPP Server configuration and management web-based interface.

## Adding a new domain

To add a new domain you need to open Configuration section of the Admin UI (by clicking on Configuration label and then selecting Add new item position which mentions `vhost-man`).



After doing that, you will be presented with a form which you need to fill in. This form allows you to pass Domain name to add and other options (some of the are advanced options).

Domain name	<input type="text" value="test.example.com"/>
<input checked="" type="checkbox"/> Enabled	
<input checked="" type="checkbox"/> Anonymous enabled	
<input checked="" type="checkbox"/> In-band registration	
<input type="checkbox"/> TLS required	
S2S secret	<input type="text" value="1696d4cb-da5c-4db1-93ab-854c38d0a5b7"/>
Domain filter policy	<input type="text" value="ALL"/>
Domain filter domains (only LIST and BLACKLIST)	<input type="text"/>
Max users	<input type="text" value="0"/>
Allowed C2S,BOSH,WebSocket ports	<input type="text"/>
Presence forward address	<input type="text"/>
Message forward address	<input type="text"/>
Other parameters	<input type="text"/>
Allowed SASL mechanisms	<input type="text"/>
Owner	<input type="text" value="admin@localhost"/>

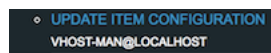
## Tip

All options with exception of Domain name may be changed later on by modifying vhost settings.

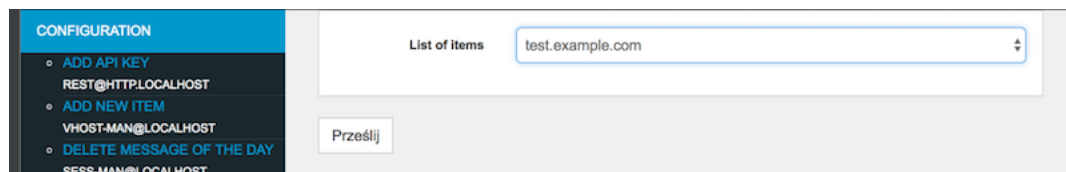
When you will be ready, please submit the form using button below the form. As a result you will be presented with a result of this operation. If it was successful it show `Operation successful` message and if something was not OK, it will display an error to help you fix this issue which you encountered.

## Modifying domain settings

Modifying a domain settings is very similar to adding a new domain. You need to open Configuration section of the Admin UI and then select `Update item configuration` position which mentions `vhost-man`.



You will be presented with a list of domains hosted on this Tigase XMPP Server installation. From them you should choose the one for which you wish to modify settings.



After submitting this selection, you will be presented with a the same form as the one used during adding a new domain. It presents configuration options for this domain and currently used values.

**Tigase XMPP Server - Admin console: Update i...**

**CONFIGURATION**

- ADD API KEY
- REST@HTTPLOCALHOST
- ADD NEW ITEM
- VHOST-MAN@LOCALHOST
- DELETE MESSAGE OF THE DAY
- SESS-MAN@LOCALHOST
- DELETE WELCOME MESSAGE
- SESS-MAN@LOCALHOST
- EDIT MESSAGE OF THE DAY
- SESS-MAN@LOCALHOST
- RELOAD REST HTTP SCRIPT HANDLERS
- REST@HTTPLOCALHOST
- REMOVE API KEY
- REST@HTTPLOCALHOST
- REMOVE AN ITEM
- VHOST-MAN@LOCALHOST
- SET MESSAGE OF THE DAY
- SESS-MAN@LOCALHOST
- SET WELCOME MESSAGE
- SESS-MAN@LOCALHOST
- SHUTDOWN
- MESSAGE-ROUTER@LOCALHOST
- UPDATE API KEY
- REST@HTTPLOCALHOST
- UPDATE ITEM CONFIGURATION
- VHOST-MAN@LOCALHOST

**EXAMPLE SCRIPTS**

**LICENCE**

**MONITOR**

Domain name: test.example.com

☒ Enabled

☒ Anonymous enabled

☒ In-band registration

☐ TLS required

S2S secret: 1696d4cb-da5c-4db1-93ab-854c38d0a5b7

Domain filter policy: ALL

Domain filter domains (only LIST and BLACKLIST):

Max users: 0

Allowed C2S,BOSH,WebSocket ports:

Presence forward address:

Message forward address:

Other parameters:

Now you should adjust them as you wish and submit this form using the button below the form.

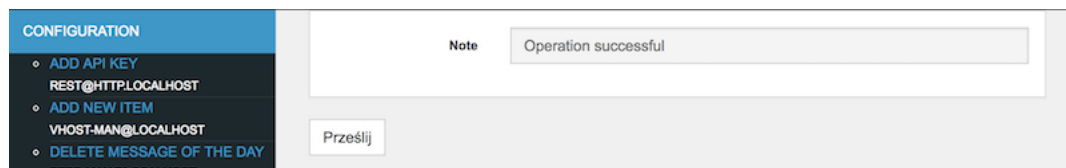
As a result you will be presented with a result of this operation. If it was successful it show `Operation successful` message and if something was not OK, it will display an error to help you fix this issue which you encountered.

## Removing a domain

Removing a hosted domain from the Tigase XMPP Server installation is quite simple as well. You need to open `Configuration` section of the Admin UI and then select `Remove an item` position which mentions `vhost-man`.



You will be presented with a list of domains hosted on this Tigase XMPP Server installation. From them you should select the one which should be removed.



After submitting your selection, Tigase XMPP Server will try to remove this domain from the list of hosted domains and will present you with the result. If it was successful it show `Operation successful` message and if something was not OK, it will display an error to help you fix this issue which you encountered.

## Using ad-hoc commands

For everybody interested in using our service to host their own XMPP domain we have good news! You do not have to ask an 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 the new functions and describes how to add a new domain and new users within your domain.

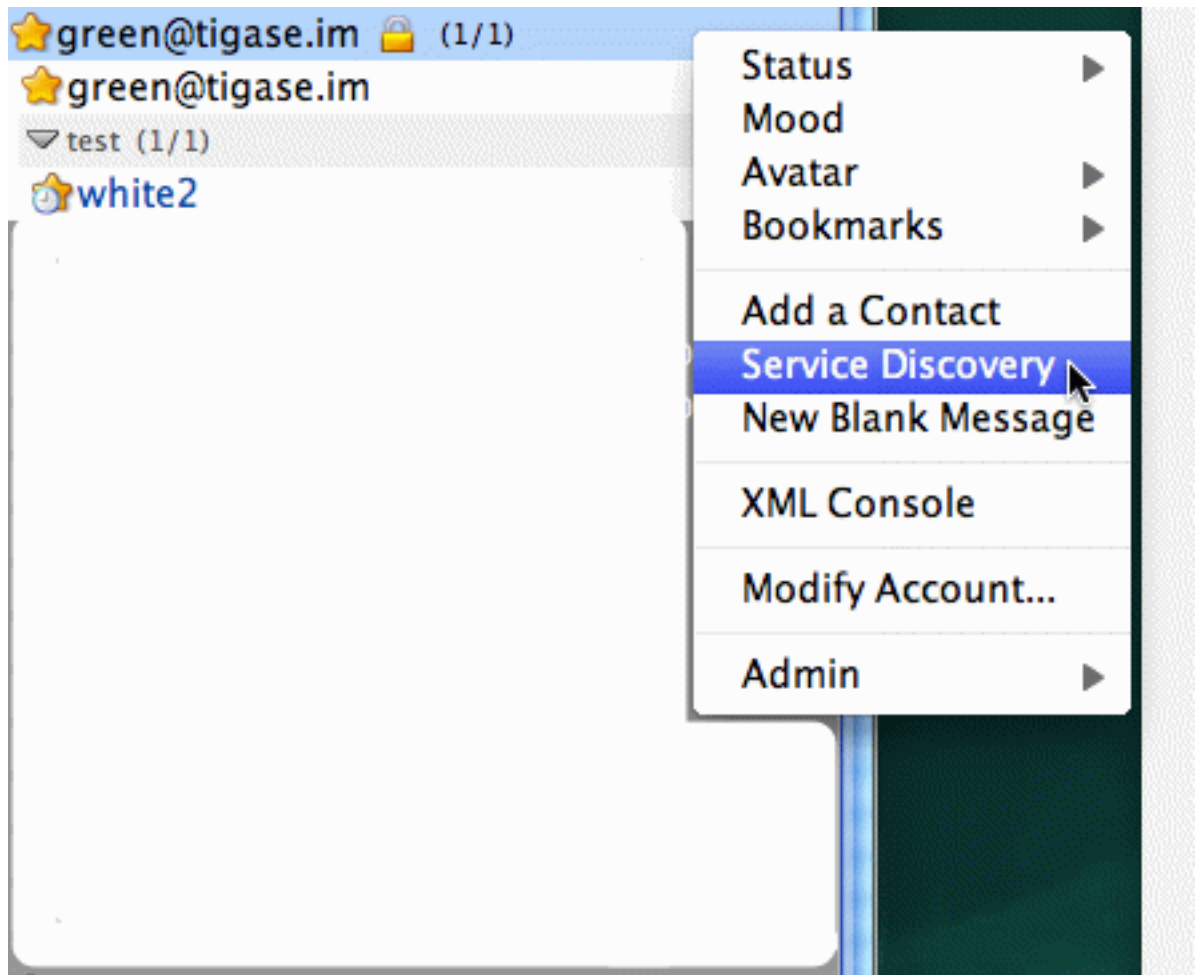
You can do everything from your XMPP client or you can use our web application that allows you to connect to the service and execute admin commands. I recommend Psi [<http://psi-im.org/>] because of its excellent support for parts of the XMPP protocol which are used for domains and user management. You may use other clients as well, but we 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/>].

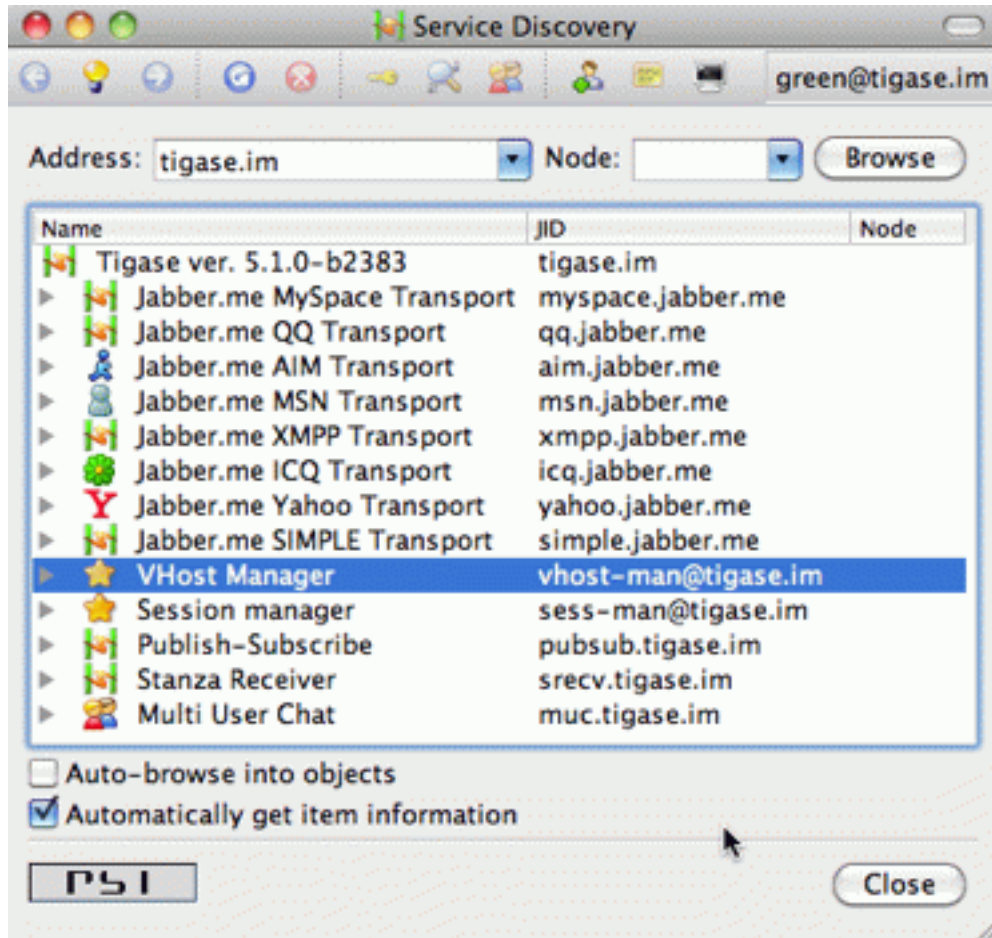
## Adding a New Domain

Once you register an account on one of the websites, connect to the XMPP server using the account on the Psi client. We will be using the following account: `green@tigase.im` [<mailto:green@tigase.im>] which is 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.



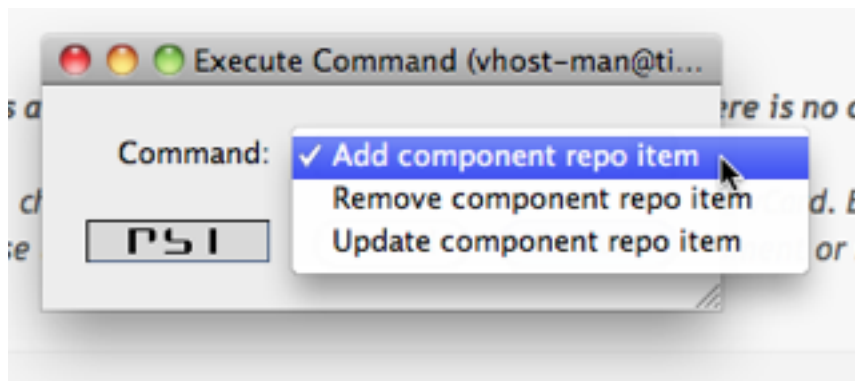
A new window pops up as in the example on the right. The service discovery window is where all the stuff installed on XMPP service should show up. Most of elements on the list are well known transports, MUC and PubSub components. The new stuff on the list, which we are interested in, are 2 elements: **VHost Manager** and **Session Manager**.



**VHost Manager** component in 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.

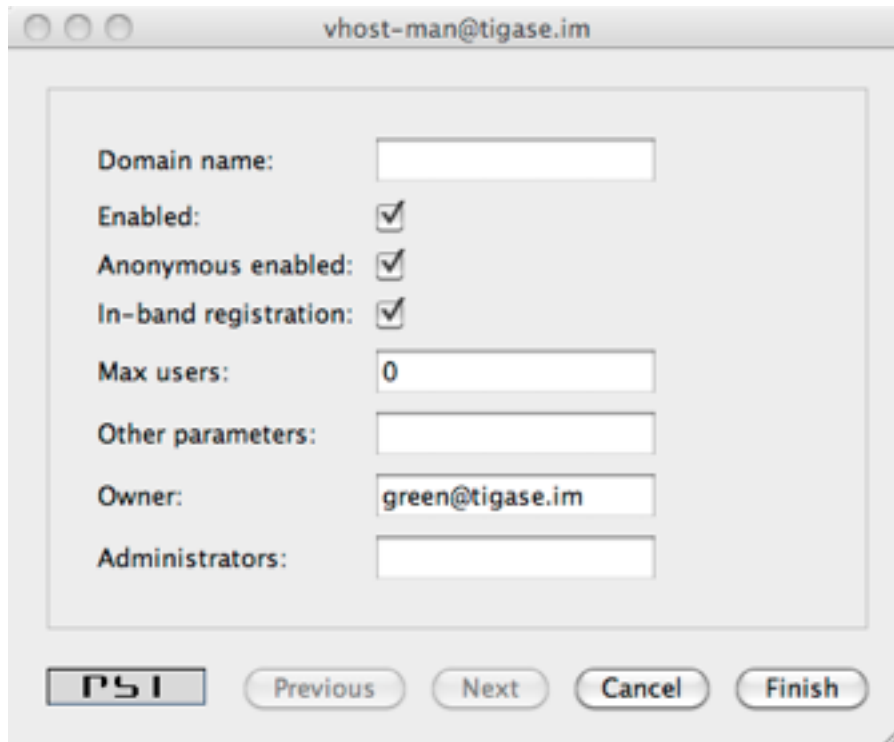
**Session Manager** component in Tigase is responsible for managing users. In most cases online users but it can also perform some actions on user repository where all user data is stored.

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...**





Click **Execute** and you get a new window where you can enter all of your VHost configuration details. All fields should be self explanatory. Leave a 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. You do not need to add your user name to the list as Owners can always manage users for the domain.

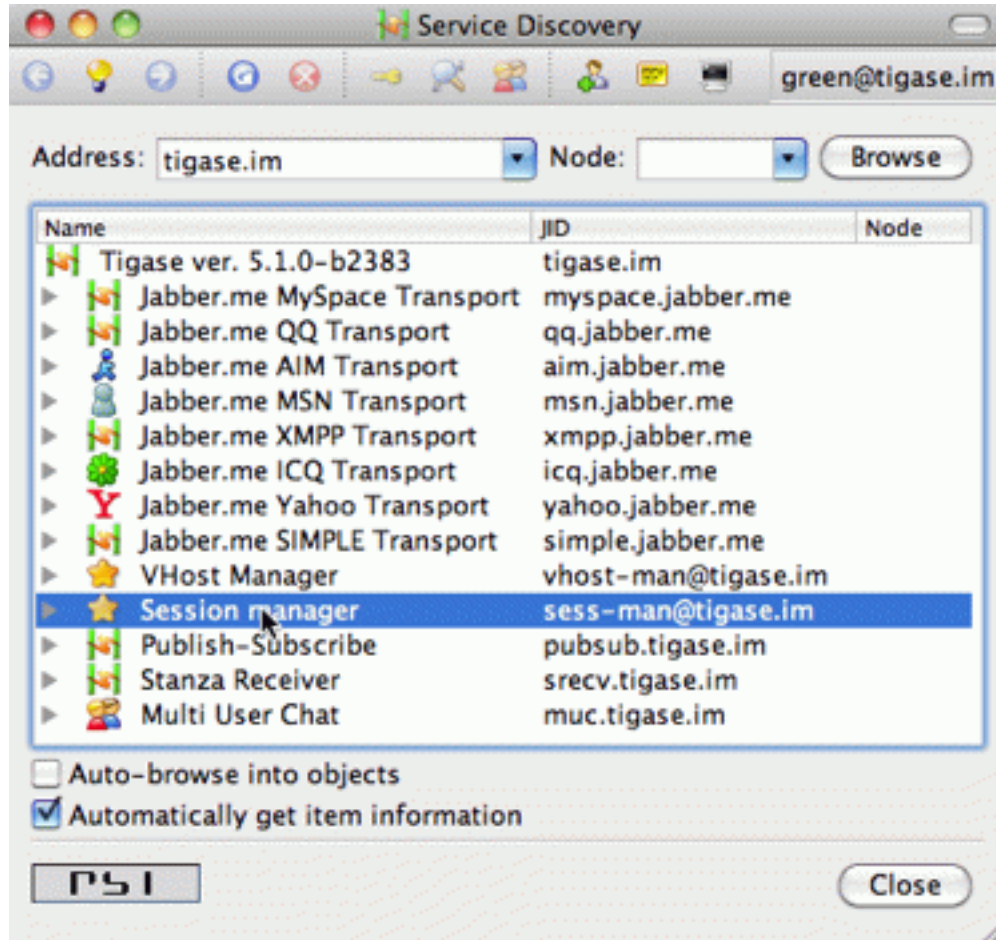


The screenshot shows a window titled "vhost-man@tigase.im". Inside, there are several configuration fields: "Domain name:" with an empty text box; "Enabled:" with a checked checkbox; "Anonymous enabled:" with a checked checkbox; "In-band registration:" with a checked checkbox; "Max users:" with a text box containing "0"; "Other parameters:" with an empty text box; "Owner:" with a text box containing "green@tigase.im"; and "Administrators:" with an empty text box. At the bottom, there is a "PSI" button and four navigation buttons: "Previous", "Next", "Cancel", and "Finish".

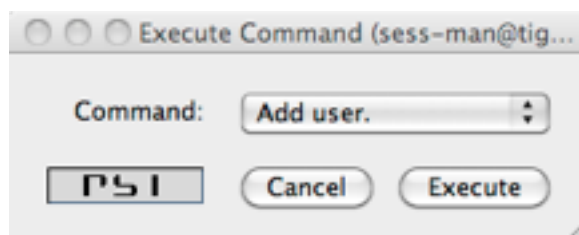
When you are ready click the **Finish** button. All done, hopefully. You can get either a window confirming everything went well or a window printing an error message if something went wrong. What can be wrong? There are some restrictions I decided to put on the service to prevent abuse. One of the restrictions 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 XMPP guide describes all the details about DNS settings. Please refer to these instructions if you need more details.

## Adding a New User

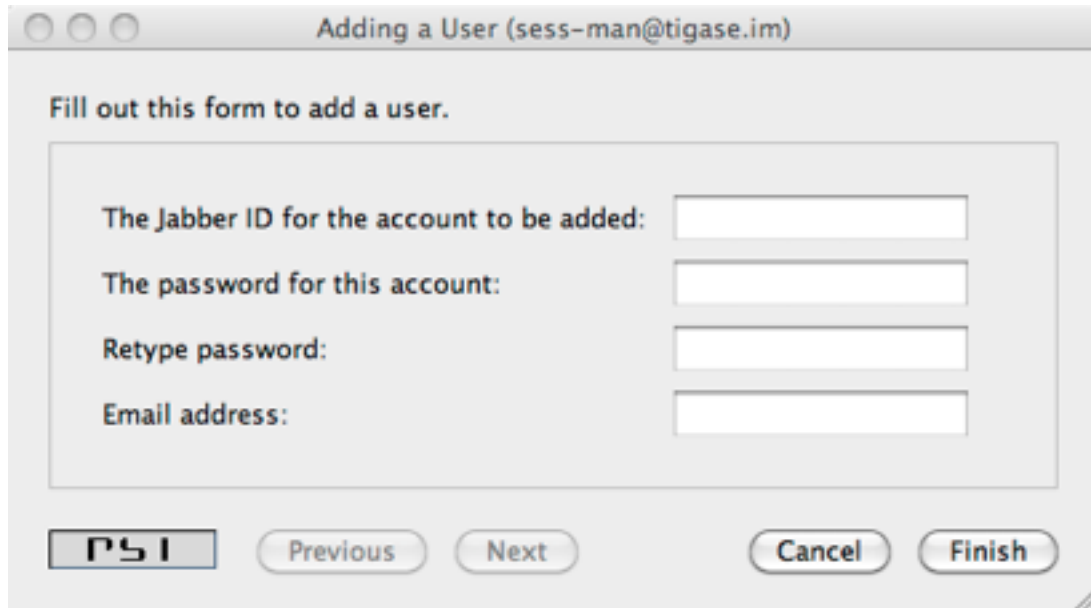
Adding a new user process is quite similar, almost identical to adding a new domain. This time, however we have to select **Session Manager** in the service discovery window.



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 click **Execute** a window presented on the left shows up. Fill all fields accordingly and press **Finish**.



If everything went well you have just added a new user and you should get a window confirming successful operation. If something went wrong, a window with an error message should show up. Possible errors may be you tried to add a user which is already present, or you may have tried to add a user for a domain to which you do not have permission or to non-existent domain.

## SSL Certificate Management

SSL Certificate Management has been implemented, and certificates can be manipulated when in a .pem form. For more details, see Creating and Loading the Server Certificate in pem Files section of documentation for more information.

## Presence Forwarding

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?

Not only is it 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 a bot address which can collect your users' presence information. Once this option is set to a valid XMPP address 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 window titled "vhost-man@devel.tigase.org". Inside, there is a configuration form for a domain named "test.tigase.org". The form includes the following fields and options:

- 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" (highlighted with a blue border), "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 included in current builds of Tigase XMPP Server.

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.getId() + "- is now -" + be.getShow()
                -" (" + msg + "-)" -)
        }
    }
)
println( "-Logging in..." -)
jaxmpp.login()
println( "-Waiting for the presence for 10 minutes" -)
Thread.sleep( 10 * 60 * 1000 -)
disconnect()
```

## Watchdog

Tigase's Watchdog was implemented to help Tigase close connections that have become stale or inactive. Sometimes the connection is delayed, maybe dropped packets, or a service interruption. After a time, if that connection is re-established, both server and client (or server and server) will continue on as if nothing happened. However, these gaps in connection can last longer, and some installations will rely on the operating system to detect and close stale connections. Some operating systems or environments can take up to 2 hours or more to determine whether a connection is bad and wait for a response from a foreign entity and may not be configured. This can not only slow down performance, but can lead to security issues as well. To solve this problem, we have introduced Watchdog to monitor connections independent of operating system and environments to keep those broken connections from becoming a problem.

## Setup

No extra setup is necessary, Watchdog is already included with your build of Tigase (as long as it's 7.1.0 or newer). Follow the steps in the configuration section.

## Watchdog Configuration

To configure watchdog, the following lines need to be present or edited in `config.tdsl` file:

```
'watchdog-timeout' = 70000
'watchdog-delay' = 60000
'watchdog-ping-type' = - 'xmpp'
```

The three settings are as follows:

- `'watchdog-timeout' = 70000` This setting sets the amount of time that watchdog will consider before it determines a connection may be stale. This setting sets the timeout at 70000ms or 70 seconds.
- `'watchdog-delay' = 60000` This setting sets how often the watchdog should conduct the check, the default delay at 60000ms or 60 seconds.
- `'watchdog-ping-type'` This setting determines the type of ping sent to components when watchdog is testing for activity.

You may, if you choose, to specify individual watchdog settings for specific components by adding them to the component settings, for example if we wanted to change the Client2Server settings to include watchdog, use the following lines in `config.tdsl`:

```
c2s {
  watchdog-delay = - '1500'
  watchdog-timeout = - '3000'
}
```

If any settings are not set, the global or settings will be used. `watchdog-delay` default is set to 10 min `watchdog-ping-type` default is set to XMPP

## Logic

Watchdog compares it's own pings, and records the time it takes for a round trip to different components, clustered connections, and if one variable is larger than the other, watchdog will commence closing that stale connection. Here is a breakdown:

1. A check is performed of a connection(s) on every `watchdog-delay` interval.
2. During this check two things occur
  - If the last transfer time exceeds `max-inactivity-time` a stop service command is given to terminate and broadcast unavailable presence.
  - If the last transfer time is lower than `max-inactivity-time` but exceeds `watchdog-timeout` watchdog will try to send a ping (of `watchdog-ping-type`). This ping may be one of two varieties (set in `config.tdsl`)
    - `WHITESPACE` ping which will yield the time of the last data transfer in any direction.
    - `XMPP` ping which will yield the time of the last received xmpp stanza.
3. If the 2nd option is true, the connection will remain open, and another check will begin after the `watchdog-delay` time has expired.

For example, lets draw this out and get a visual representation

This line is client activity, here the client sent a message at 40 seconds (marked by +) and has gone idle.

```
}
'cl-comp' {
    -'max-inactivity-time' = 180L
}
s2s {
    -'max-inactivity-time' = 900L
}
ws2s {
    -'max-inactivity-time' = 86400L
}
```

## Important

Again remember, for Watchdog to properly work, the `max-inactivity-time` **MUST** be longer than the `watchdog-timeout` setting

## Testing

The `tigase.log.0` file can reveal some information about watchdog and how it is working (or how it might be fighting your settings). To do so, enter the following line into your `config.tdsl` file:

```
debug = [ -'server', -'xmpp.init' -]
```

This will set debug mode for your log, and enable some more information about what Tigase is doing. These logs are truncated for simplicity. Lets look at the above scenario in terms of the logs:

### Stage Two.

```
2015-10-16 08:00:00.000 [Watchdog -- c2s]    ConnectionManager$Watchdog$1.check()
```

### Stage Three.

```
2015-10-16 08:01:00.000 [Watchdog -- c2s]    ConnectionManager$Watchdog$1.check()
```

### Stage Four.

```
2015-10-16 08:02:00.000 [Watchdog -- c2s]    ConnectionManager$Watchdog$1.check()
2015-10-16 08:02:00.697 [Watchdog -- c2s]    ConnectionManager$Watchdog$1.check()
```

### Stage Five.

```
2015-10-16 08:03:00.000 [Watchdog -- c2s]    ConnectionManager$Watchdog$1.check()
2015-10-16 08:03:00.248 [pool-20-thread-6]    ConnectionManager.serviceStopped() FI
2015-10-16 08:03:00.248 [pool-20-thread-6]    ClientConnectionManager.xmppStreamClos
```

## Tips and Tricks

The section contains some short tricks and tips helping in different kinds of issues related to the server administration and maintenance.

- Runtime Environment Tip
- Best Practices for Connecting to Tigase XMPP server From Web Browser



## 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, the problem still existed after uploading a new version and restarting the server. It worked fine in our development environment and it just didn't on the target system.

It turned out that due to specific environment settings on the target system, an old version of Tigase server was always started regardless of what updates uploaded. We finally located the problem by noticing that the logs were not being generated in the proper locations. This led us to finding the issue: improper environment settings.

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.

## Best Practices for Connecting to Tigase XMPP server From Web Browser

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 information 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 this protocol being widely supported and used. 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 enabled. In this case the 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, the XMPP client should randomly pick other node from list of rest of available nodes.

*Solution:*

Tigase XMPP Server by default provides server side solution for this issue by sending additional `host` attribute in body element of BOSH response. As value of this attribute Tigase XMPP Server sends domain name of server cluster node to which client connected and to which next connections of this session should be opened. It is possible to disable this custom feature by addition of following line to `etc/config.tds1` config file:

```
bosh {  
    - 'send-node-hostname' = false  
}
```

*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 the 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 the server it tries to connect to (`example.com` should be value of `to` attribute of XMPP stream).

## WebSocket

WebSocket protocol is newly standardized protocol which is supported 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 servers. 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/config.tds1` config file:

```
ws2s {
  connections {
    ports = [ 5290, 5291 -]
    5290 {
      socket = -'ssl'
      type = -'accept'
    }
    5291 {
      socket = -'plain'
      type = -'accept'
    }
  }
}
```

In this example we enabled WebSocket endpoint on port 5290 allowing unencrypted connections, and encrypted WebSocket endpoint on port 5291. 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`.

### Note

There is no default configuration for non-default ports. All ports outside 443 MUST be configured.

## 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. That 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 URL, if our default certificate is for domain `example2.com`. In this situation it is recommended to use the default certificate for the domain under which the server is accessible from the internet. This domain should identify this server, so this domain would not point to 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 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 in hosting domain `example2.com`. Each of our nodes contains default SSL certificate with domain names matching the 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 the following URL: `wss://t2.example1.com:5291/`. Upon connection the 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.

# Licensing

With the release of v7.1.0, users and commercial clients alike may now be able to register and request a license file from our servers on their own. This process makes it easier for everyone to obtain valid license file when needed. Users who do not wish to register will not be required to register. However, If you are using Tigase ACS or other commercial pieces of software, you will be required to register.

## Warning

Tigase XMPP Server will shut down during license check if no installation-id or license is received within a given period of time.

**Again, Tigase XMPP Server will still be available free under AGPLv3, and free users will not need to register.**

## Note

COMMERCIAL COMPONENTS REQUIRE THE USE OF A LICENSE.

# Registering for a License

There are currently two ways for registering for a license with Tigase commercial products. **The easiest and recommended method is using the built in automatic registration function.** However, you may also register via a web portal if your installation has limitations on network connectivity.

```
<AutomaticLicenceRegistration>
<title>Automatic Registration (recommended)</title>
```

Once a commercial component is activated on Tigase XMPP Server, the program will then retrieve an *Installation ID* from our servers, and make a file called `installation-id` in your `etc/` directory including the *Installation ID* for your instance. An installation ID is generated using the complete cluster map and all machines within the same cluster should have the same *Installation ID*. This *Installation ID* will then be sent along with server details to a license server, and appropriate license files will be made in your `tigasedir/etc` directory. When the license is due to be expired, this mechanism will update your license file automatically.

```
</AutomaticLicenceRegistration>
```

## Manual

### Caution

This method should be used only in extreme cases when ??? can't be used.

If you do not wish to use the automatic method, you may decide to generate a license file using our web portal. Offline installation may obtain *Installation IDs* from our web portal in a three-step process: registration, generating hash, and obtaining license file.

## Generating Installation ID

For offline installations, you may obtain an *Installation ID* from this address: <https://license.tigase.software/register>.

Data Fields:

- `Customer name`: Company or user name used to identify machines. Multiple clusters or servers can have the same customer name.
- `VHosts`: Comma separated list of VHosts you will be using on this node. NOTE: these fields are case sensitive!
- `Legacy license hashes`: Copy the digest hash generated for all legacy licenses - it's available in the `etc/tigase-console.log` after startup (if such licenses are present).
- `Captcha question`: Enter the basic math answer for this form to prove you are not a robot.

The next page will provide you with an installation ID like the following:

```
1TCICGG7K8AS2JSSEVMDA9QOLR4NVLJSR
```

Edit your `config.tdsl` file and add your installation-id

```
'installation-id' = - '1TCICGG7K8AS2JSSEVMDA9QOLR4NVLJSR'
```

Note that the `installation-id` file will be made automatically once the license file is installed and verified by the server.

## Obtaining a Server Code

Once you have the *Installation ID*, you will need to generate a server code. This can be done by accessing the admin UI page and navigating to the License section. Once there, click on Retrieve code for license. Select the component you wish to generate a code for and click Submit. You will see a fields with installation-id, module, VHosts filled out based on your server's configuration. Copy the contents of the Code field and proceed to the next section.

## Obtaining license file

Open a new browser and navigate to this address: <https://license.tigase.software/retrieve> once there, paste the generated code from the last step in the field and click submit. Afterwards you will be prompted to download a license file, place this file in your `etc/` folder and restart your server, your license is now activated and installed on your server.

**If you are provided a manually produced license, you will need to place it in the same `etc/` directory with the name `<component_name>.license` (e.g.: `etc/acs.license`)**

## What happens if I do not use a license file or it is expired?

Tigase permits commercial products to be used without a license, but a validation process must complete otherwise the server will shutdown. Within the first hour of runtime, Tigase will check for the presence and validity of the license file. If none is found, or it is invalid or expired the server will then contact Tigase master server in order to obtain a valid one.

Communications will be made to [license.tigase.software](https://license.tigase.software) over https (port 443) to verify the license or download a new one.

## Demo mode

If no valid license can be found, Tigase will revert to a demonstration mode. Most functions will be available and usable, but with a caveat. Statistics from that server will be sent to <https://stats.tigase.software>

about your server and it's usage. Details are in the next section. If this information cannot be sent, the server will assume unauthorized use and will shut down.

## Statistics Sent

Statistics of your server may be sent to Tigase server's if the all of following happens:

- You are using commercial Tigase components.
- You have registered an `installation-id`.
- You do not have a current license to run Tigase commercial components.

If these conditions exist, statistics will be sent to our servers and a warning will be posted in your logs. The following is an example of what information will be sent.

### Note

The text below has been better formatted for readability, but does not reflect the actual text being sent to Tigase.

```
<statistics version="1">
  <domain>xmppserver</domain>
  <timestamp>2016-06-23T17:16:24.777-0700</timestamp>
  <vhosts>
    <item>vhost1.xmppserver.com</item>
  </vhosts>
  <uptime>308833</uptime>
  <heap>
    <used>30924376</used>
    <max>1426063360</max>
  </heap>
  <cluster>
    <nodes_count>1</nodes_count>
  </cluster>
  <users>
    <online>0</online>
    <active>0</active>
    <max_today>1</max_today>
    <max_yesterday>0</max_yesterday>
  </users>
  <additional_data>
    <components>
      <cmpInfo>
        <name>amp</name>
        <title>Tigase XMPP Server</title>
        <version>7.1.0-SNAPSHOT-b4226/5e7210f6 (2016-06-01/23:15:52)</version>
        <class>tigase.cluster.AmpComponentClustered</class>
      </cmpInfo>

      <cmpInfo>
        <name>bosh</name>
        <title>Tigase XMPP Server</title>
        <version>7.1.0-SNAPSHOT-b4226/5e7210f6 (2016-06-01/23:15:52)</version>
```

```
<class>tigase.cluster.BoshConnectionClustered</class>
</cmpInfo>

<cmpInfo>
  <name>c2s</name>
  <title>Tigase XMPP Server</title>
  <version>7.1.0-SNAPSHOT-b4226/5e7210f6 (2016-06-01/23:15:52)</version>
  <class>tigase.cluster.ClientConnectionClustered</class>
</cmpInfo>

<cmpInfo>
  <name>cl-comp</name>
  <title>Tigase XMPP Server</title>
  <version>7.1.0-SNAPSHOT-b4226/5e7210f6 (2016-06-01/23:15:52)</version>
  <class>tigase.cluster.ClusterConnectionManager</class>
</cmpInfo>

<cmpInfo>
  <name>eventbus</name>
  <title>Tigase XMPP Server</title>
  <version>7.1.0-SNAPSHOT-b4226/5e7210f6 (2016-06-01/23:15:52)</version>
  <class>tigase.disteventbus.component.EventBusComponent</class>
</cmpInfo>

<cmpInfo>
  <name>http</name>
  <title>Tigase HTTP API component: Tigase HTTP API component</title>
  <version>1.2.0-SNAPSHOT-b135/27310f9b-7.1.0-SNAPSHOT-b4226/5e7210f6 (2016-06-01/23:15:52)</version>
  <class>tigase.http.HttpMessageReceiver</class>
</cmpInfo>

<cmpInfo>
  <name>monitor</name>
  <title>Tigase XMPP Server</title>
  <version>7.1.0-SNAPSHOT-b4226/5e7210f6 (2016-06-01/23:15:52)</version>
  <class>tigase.monitor.MonitorComponent</class>
</cmpInfo>

<cmpInfo>
  <name>muc</name>
  <title>Tigase ACS -- MUC Component</title>
  <version>1.2.0-SNAPSHOT-b62/74afbb91-2.4.0-SNAPSHOT-b425/d2e26014</version>
  <class>tigase.muc.cluster.MUCComponentClustered</class>
  <cmpData>
    <MUCClusteringStrategy>class tigase.muc.cluster.ShardingStrategy</class>
  </cmpData>
</cmpInfo>

<cmpInfo>
  <name>pubsub</name>
  <title>Tigase ACS -- PubSub Component</title>
  <version>1.2.0-SNAPSHOT-b65/1c802a4c-3.2.0-SNAPSHOT-b524/892f867f</version>
  <class>tigase.pubsub.cluster.PubSubComponentClustered</class>
  <cmpData>
```

```
<PubSubClusteringStrategy>class tigase.pubsub.cluster.Partitio
</cmpData>
</cmpInfo>

<cmpInfo>
  <name>s2s</name>
  <title>Tigase XMPP Server</title>
  <version>7.1.0-SNAPSHOT-b4226/5e7210f6 (2016-06-01/23:15:52)</vers
  <class>tigase.server.xmppserver.S2SConnectionManager</class>
</cmpInfo>

<cmpInfo>
  <name>sess-man</name>
  <title>Tigase XMPP Server</title>
  <version>7.1.0-SNAPSHOT-b4226/5e7210f6 (2016-06-01/23:15:52)</vers
  <class>tigase.cluster.SessionManagerClustered</class>
  <cmpData>
    <ClusteringStrategy>class tigase.server.cluster.strategy.Onlin
    </ClusteringStrategy>
  </cmpData>
</cmpInfo>

<cmpInfo>
  <name>ws2s</name>
  <title>Tigase XMPP Server</title>
  <version>7.1.0-SNAPSHOT-b4226/5e7210f6 (2016-06-01/23:15:52)</vers
  <class>tigase.cluster.WebSocketClientConnectionClustered</class>
</cmpInfo>

<cmpInfo>
  <name>vhost-man</name>
  <title>Tigase XMPP Server</title>
  <version>7.1.0-SNAPSHOT-b4226/5e7210f6 (2016-06-01/23:15:52)</vers
  <class>tigase.vhosts.VHostManager</class>
</cmpInfo>

<cmpInfo>
  <name>stats</name>
  <title>Tigase XMPP Server</title>
  <version>7.1.0-SNAPSHOT-b4226/5e7210f6 (2016-06-01/23:15:52)</vers
  <class>tigase.stats.StatisticsCollector</class>
</cmpInfo>

<cmpInfo>
  <name>cluster-contr</name>
  <title>Tigase XMPP Server</title>
  <version>7.1.0-SNAPSHOT-b4226/5e7210f6 (2016-06-01/23:15:52)</vers
  <class>tigase.cluster.ClusterController</class>
</cmpInfo>
</components>

<unlicencedComponenents>
  <ComponentAdditionalInfo name="acs"/>
</unlicencedComponenents>
```



```
</additional_data>
</statistics>
```

## Unauthorized use

Tigase will consider itself unauthorized if the following conditions are met:

- if Tigase XMPP Server does not have a valid license file and
- cannot contact the licensing server to obtain installation id and attached licenses.

Then the program will then attempt to send statistics.

- if unable to sent statistics the server after a random number of retries.
- if these retries are not successful within 10 attempts, the server will then shutdown.

If you are experiencing this condition, please contact Tigase.

## Manual mode

If you cannot open communication to `stats.tigase.software` or `license.tigase.software` over the required ports (https over port 443), you may request to use manual mode. Manual mode requires Tigase to create a license file to be used on your machine locally. This must be placed in the same folder as the above information, and the license check system will not seek communication unless the license is invalid or expired.

## Tigase Clustering

Tigase Clustering allows the use of a number of servers to be unified in delivering, from what a client or user sees, a single unified platform. There are two typical reasons why clustering should be employed:

- High Availability

By using clustering, services can be provided with a high reliability and redundancy.

- Load Balancing

This type of cluster helps to distribute a workload over a number of servers to improve performance.

With Tigase, you don't have to choose between either/or!

**Tigase Clustering** offers **Full Redundancy** and **Automatic Load Balancing** allowing addition of new nodes at runtime with a simple configuration. All without a severe tax on resource consumption.

All basic components support clustering configuration, and some may be turned on or off.

## Configuration

To enable Clustering on Tigase servers, use the following line in your `config.tds1` file:

```
'cluster-mode' = true
```

That's it!

## Custom Ports

You can customize ports for the cluster component, just be sure that each clustered server also has the same settings so they can communicate.

```
cl-comp {
  connections {
    4250 {}
    3540 {}
  }
}
```

You can fine tune each port configuration, however this is not typically needed.

## Custom Port Configuration

Each port has it's own details that can be manipulated via the following ports. Again **THIS IS OPTIONAL**

```
'cl-comp' {
  connections {
    4250 {
      ifc = [ '*' -]
      -'remote-host' = -'localhost'
      socket = -'plain'
      type = -'accept'
      connections {
        tls {
          required = false
        }
      }
    }
  }
}
```

## Multi-node configuration

Each node should have 'cluster-mode' = true enabled that you wish to connect to the cluster. They will automatically discover other nodes to connect to VIA Server to Server traffic. Nodes that are added or removed will be periodically updated.

## Traffic Control

You can customize the traffic going between clustered servers with a few options.

### cm-ht-traffic-throttling

This setting will control the number of bytes sent over non-user connections. Namely, Server to Server or S2S connections.

```
'cm-ht-traffic-throttling' = -'xmpp:25k:0:disc,bin:200m:0:disc'
```

The format is as follows: {traffic-type}:{maximum-traffic}:{max-lifespan-traffic}:{action}

traffic-type	Specifies the type of traffic controlled. This can either be <b>XMPP</b> or <b>bin</b> . XMPP limits the number of packets transferred, whereas bin limits the number of bytes transferred.
maximum-traffic	Specifies how many bytes or packets may be sent within one minute.
max-lifespan-traffic	Specifies how many bytes or packets may be sent within the lifetime of the connection. 0 means unlimited.
action	Specifies the action to be taken which can be <b>disc</b> which disconnects the connection, or <b>drop</b> which will drop any data exceeding the thresholds.

### cm-see-other-host

This allows the specific use of a load balancing mechanism by selecting `SeeOtherHostIfc` implementation. For more details, see Tigase Load Balancing documentation.

## Old configuration method

While these options are still available these settings CAN be less reliable. **Use ONLY if you need specific setups that cannot be accommodated by the automatic cluster mode.**

### Specifying Specific nodes

You can still use the old method of specifying every node on each server. Server 3 needs the following set

```
'cluster-nodes' = [ -'serv1.xmpp-test.org' -, -'serv2.xmpp-test.org' -]
```

Server 2 needs

```
'cluster-nodes' = [ -'serv1.xmpp-test.org' -, -'serv3.xmpp-test.org' -]
```

and so on...

However, we do not recommend this.

### Password and Port configuration

You may specify a password and port to specific cluster servers if that is required. To do so, you will need to add `{password}:{port}` to the domain, like this example:

```
'cluster-nodes' = [ -'serv1.xmpp-test.org:domainpass:5600' -]
```

## Checking Cluster Connections

After setting up clustering you may want to verify that the clusters are operational. Right now it can be done in two manners - first by checking that there are actual network connections established between cluster nodes. The other 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

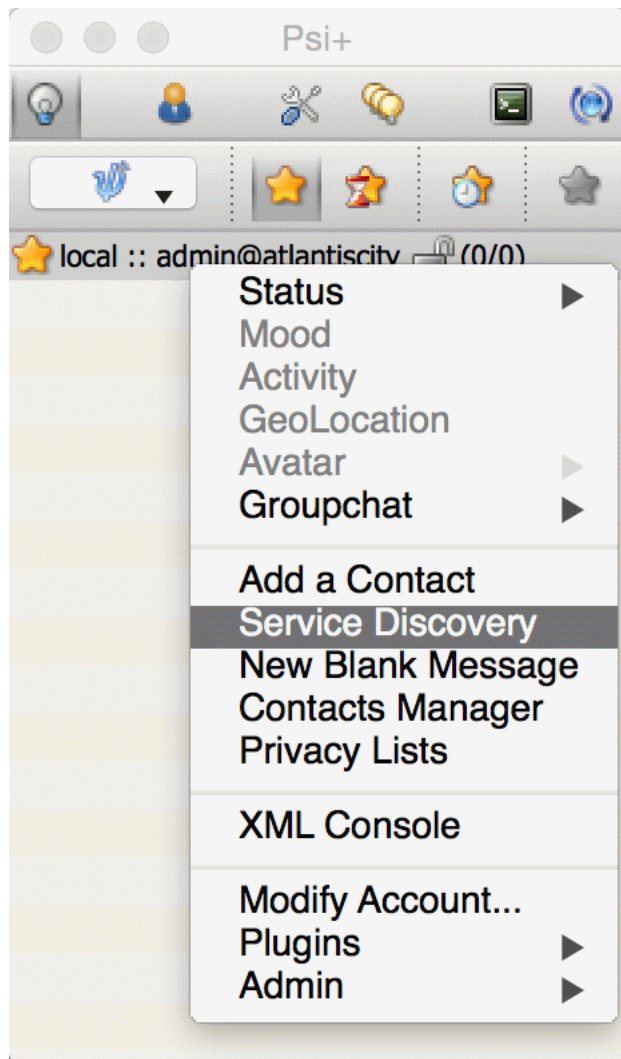
```
C:\WINNT>netstat --anp tcp -| find -":5277 -"
```

## Cluster nodes connected (using XMPP)

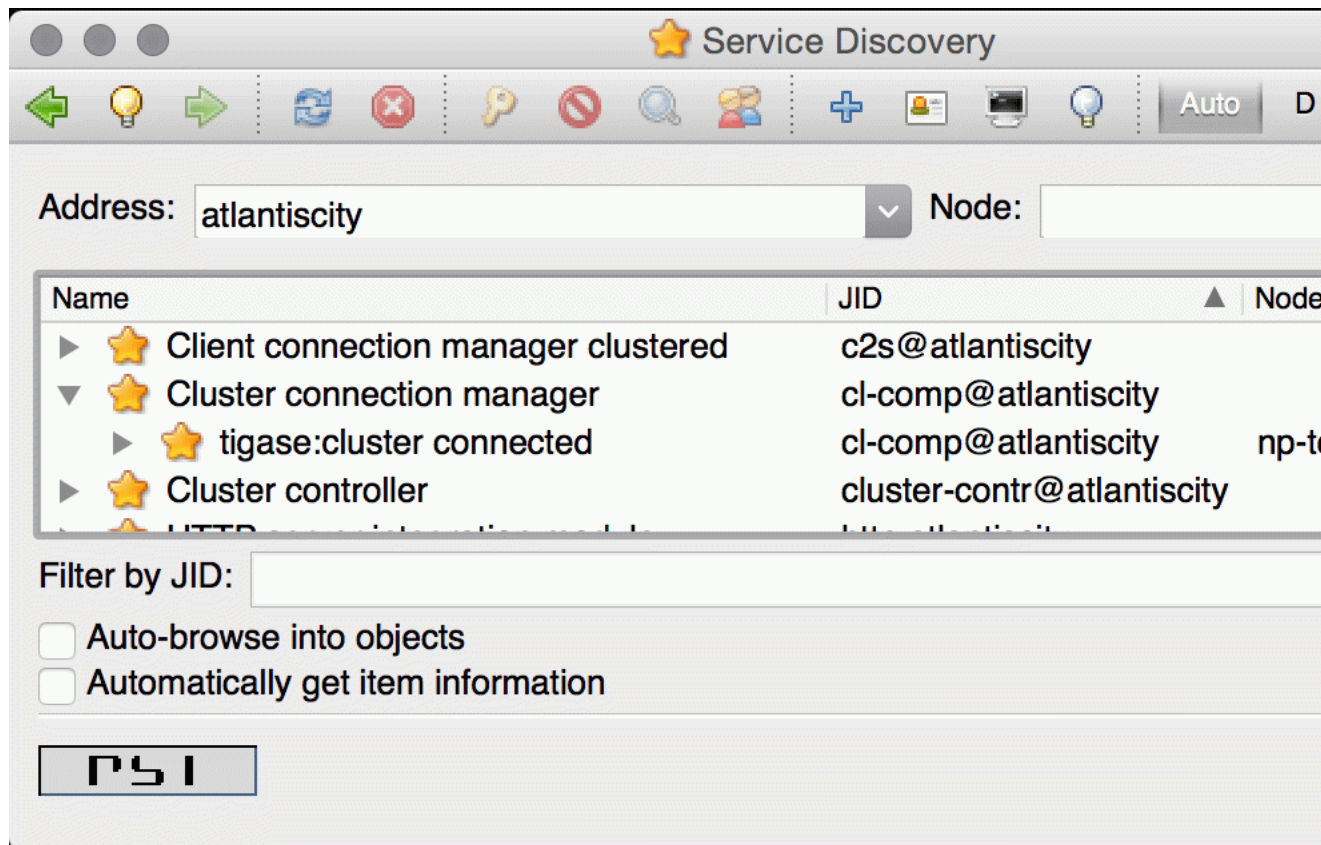
Verifying clustering connectivity over XMPP protocol requires any XMPP client capable of XEP-0030: Service Discovery [<http://xmpp.org/extensions/xep-0030.html>]. It's essential to remember that only an administrator (a user whose JID is configured as administrative) has access.

### 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:



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:



## Anonymous Users & Authentication

To support anonymous users, you must first enable anonymous authentication on your server.

### Anonymous Authentication

Tigase Server can support anonymous logins via SASL-ANONYMOUS in certain scenarios. This can be enabled on per-VHost basis by adjusting *Anonymous enabled* option as described in the section called “Add and Manage Domains (VHosts)” This setting is false by default as SASL-ANONYMOUS may not be totally secure as users can connect without prior permission (username and password).

### Anonymous User Features

To connect to your server anonymously, you must use a client that supports anonymous authentication and users. Connect to the server with the name of the server as the username, and no password. For example, to connect anonymously to `xmpp.example.com` use the following credentials,

Username: `xmpp.example.com` Password:

In this mode all login information is stored in memory, and cannot be retrieved at a later date.

Other features of Anonymous Authentication

- Temporary Jid is assigned and destroyed the moment of login/logout.
- Anonymous users cannot access the database

- Anonymous users cannot communicate outside the server (use s2s connections)
- Anonymous users have a default limit on traffic generated per user.

## Reconnection on Anonymous

On products such as our JaXMPP Server, users connected using SASL-ANONYMOUS can reconnect to existing sessions using cookie management. However, reconnection can be improved and extended using Bosh Session Cache [[http://docs.tigase.org/tigase-server/snapshot/Development\\_Guide/html/#bosh-sessioncache](http://docs.tigase.org/tigase-server/snapshot/Development_Guide/html/#bosh-sessioncache)] which allows for session storage in memory rather than using client-side data for reconnection.

## Scripting support in Tigase

Tigase server supports scripting languages in versions 4.3.1 and higher. 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.

### Note

Tigase server is known for its very low memory consumption and successfully runs with less than 10MB of RAM memory. However adding scripting support for any non-standard (default) language to Tigase server significantly increases memory requirements for the installation. You cannot expect Tigase server to run on 10MB RAM system if you enabled Python, Scala or any other non-standard language.

## Scripting Introduction - Hello World!

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

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

In theory many different languages can be used to write scripts and the only requirement is that support JSR-223 [<http://www.jcp.org/en/jsr/detail?id=223>] for the language is installed. 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 we recommend Groovy [<http://groovy.codehaus.org/>]. However the 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 [<https://projects.tigase.org/projects/tigase-server/repository/revisions/master/show/src/main>] contains a few examples for these languages.

### 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` [<https://projects.tigase.org/projects/tigase-server/repository/revisions/master/show/src/main/groovy/tigase/admin>].

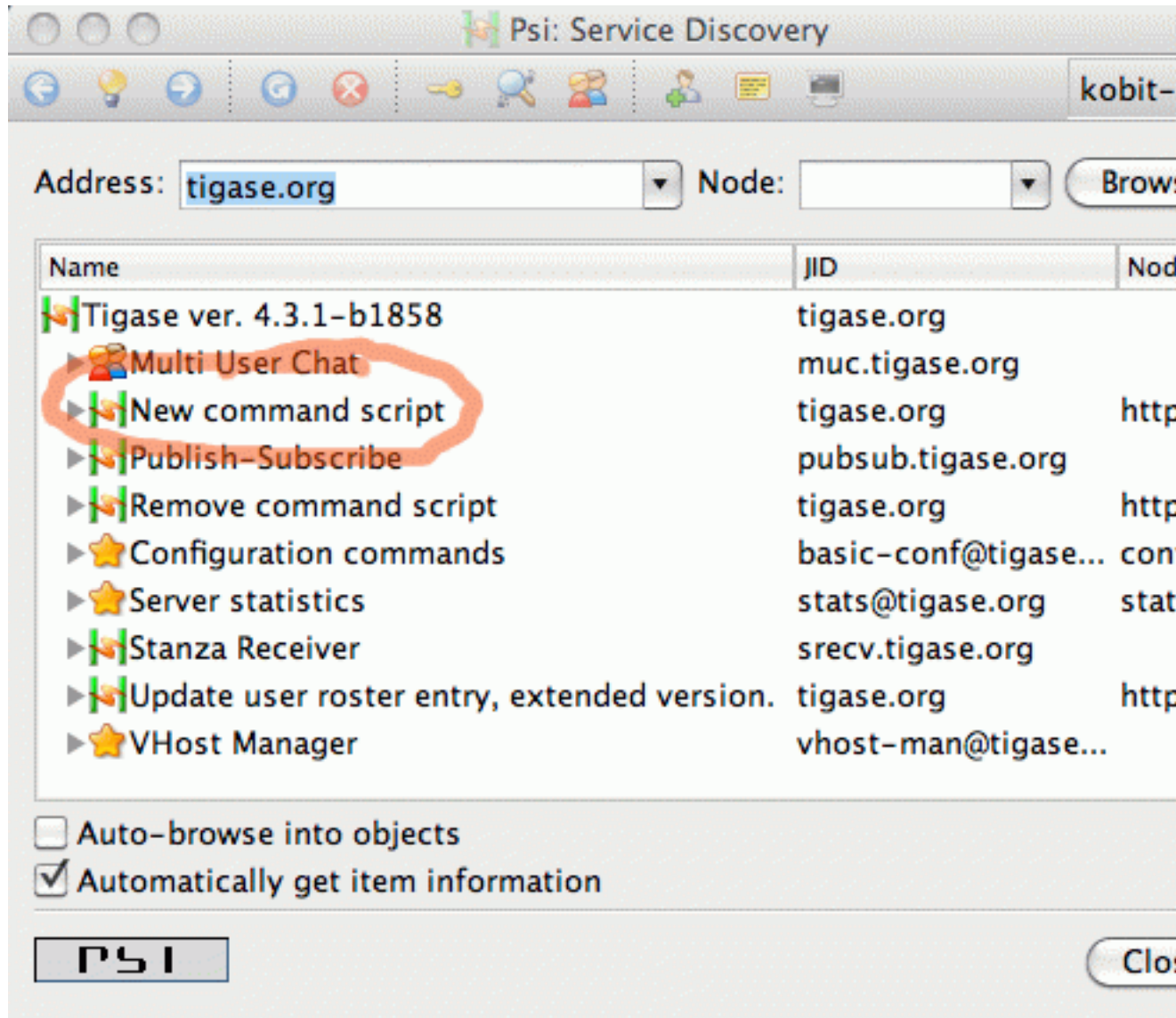
The scripting utilizes only standard XMPP extensions and is by no means specific to any particular solution. We use and prefer Psi client. The whole guide and all the screen-shots are created using Psi client.

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 a XMPP client with a good support for both features.

To follow the guide and run all the examples you need will need to have installed Tigase server version 4.3.1 or newer and you have to connect to the server as administrator.

## Loading Script at Run Time

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

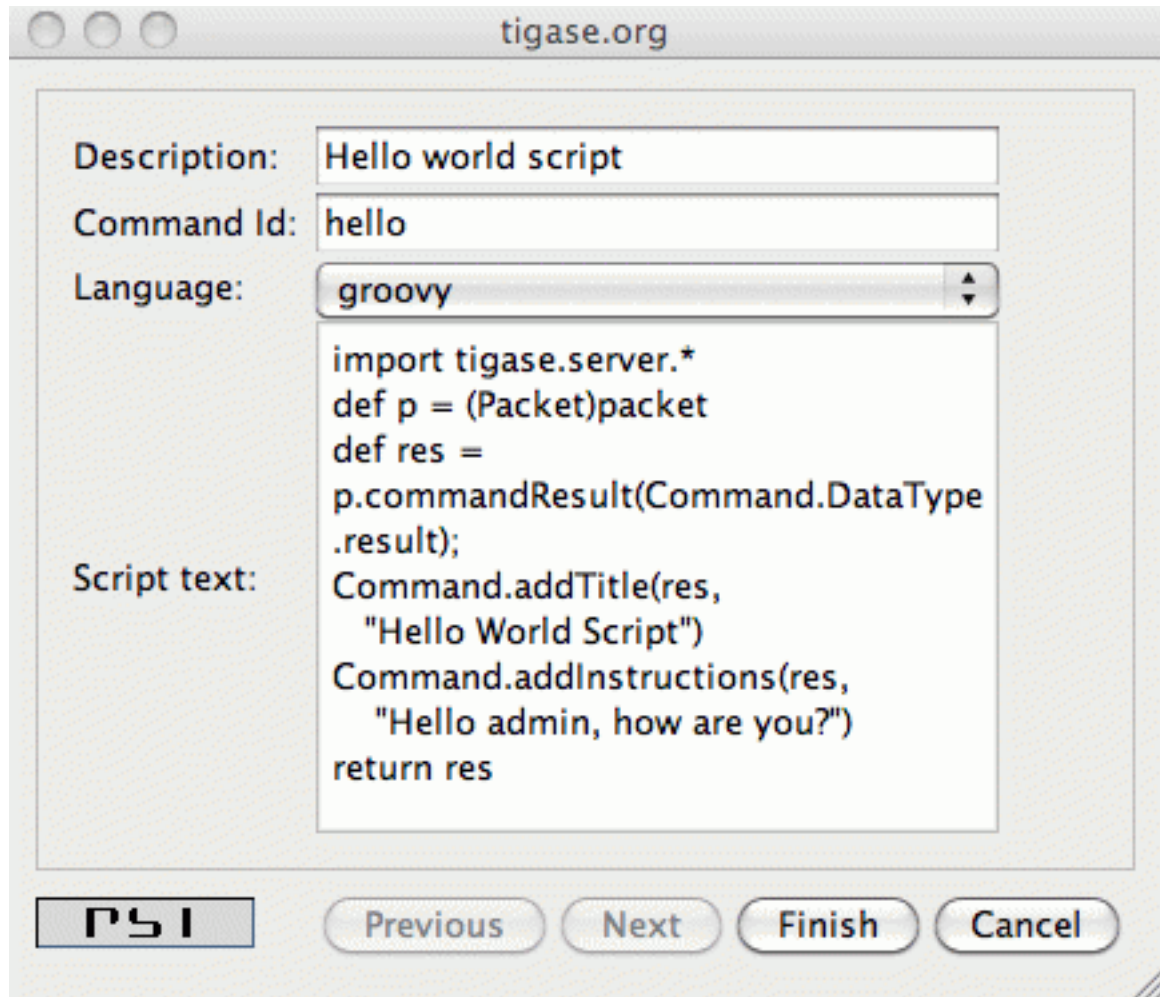


The first thing to do, therefore, is to browse service discovery on the running server. The result you receive will depend on your installation and installed components.

The most interesting things 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: 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.

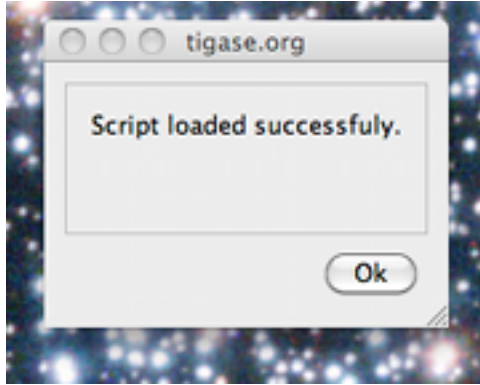


The screenshot above shows a couple of options to set for the loaded script:

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. Tigase automatically detects all libraries for scripting languages and lists them here. So all you need is to select the 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 "Hello world" script written in Groovy. What it does is displays 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:

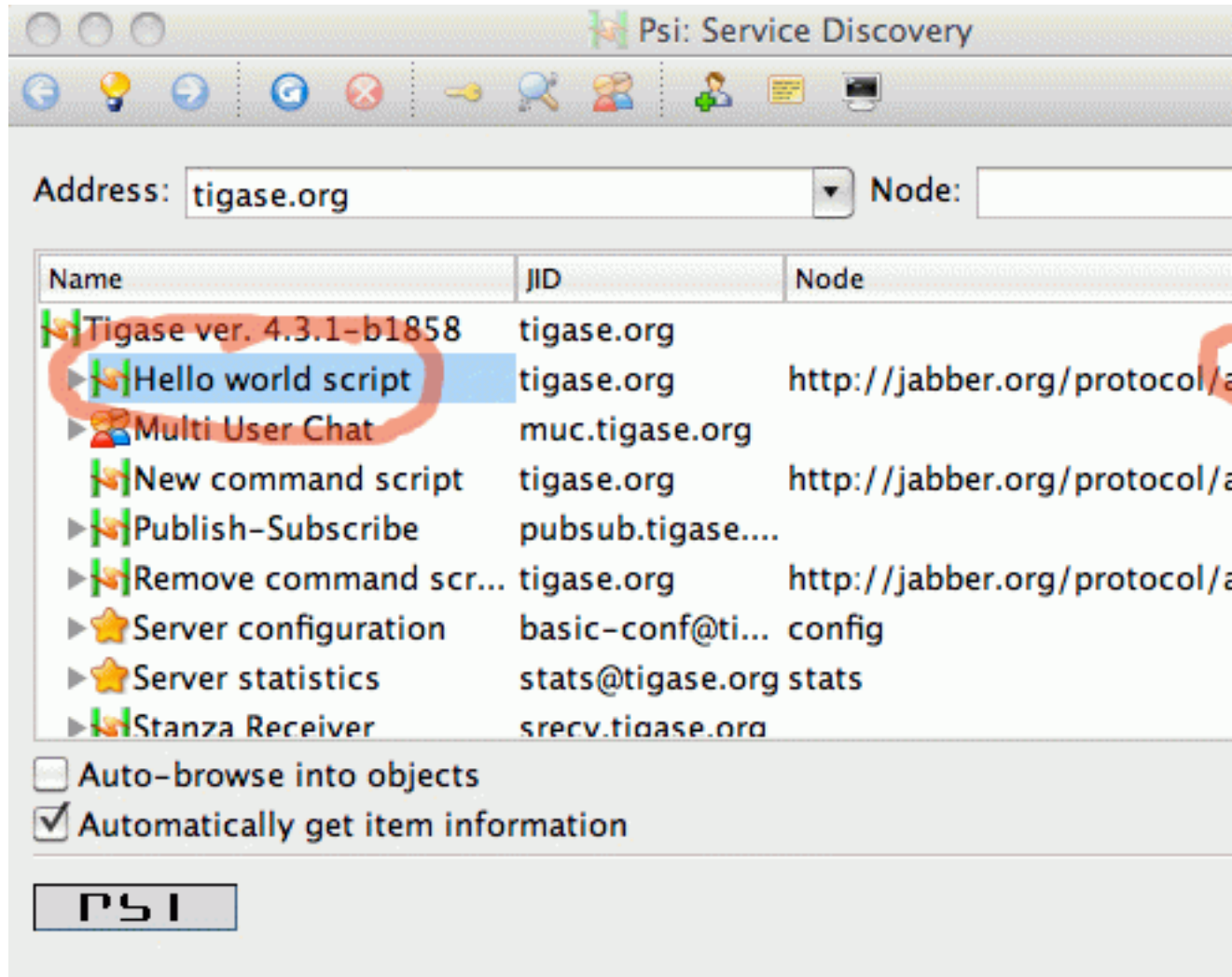
1. It imports basic Tigase classes.
2. Sets a local variable `p` which points to a `packet` variable with data received from the client.
3. Creates a `res` variable which is response sent back to the client (administrator). The response to the client is of type `result`. Other possible types will be 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

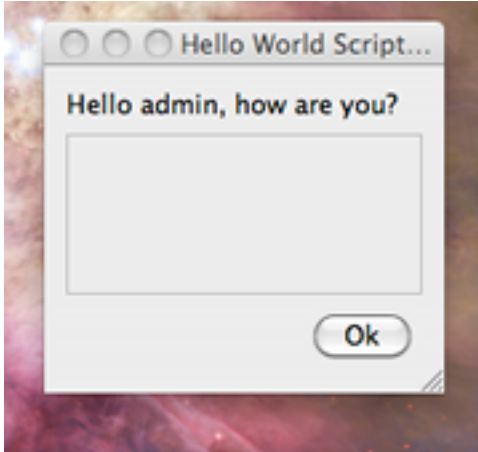
Once the script is successfully loaded you will have to reload/refresh the service discovery window which now should display one more element on the list.



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).

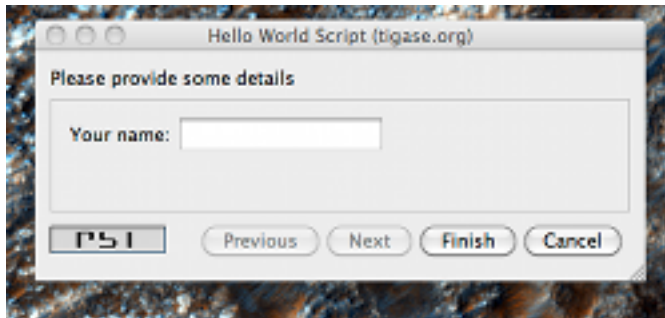
As a result you should see a simple window similar to the screenshot below displaying our message.



## Interaction in Scripts

Displaying just a message is very nice but is not very useful in most cases. 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 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 a greeting.



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")

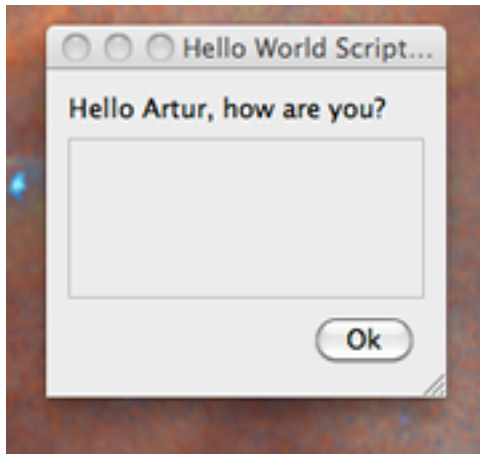
if (name == null) {
    def res = p.commandResult(Command.DataType.form);
    Command.addTitle(res, -"Hello World Script")
    Command.addInstructions(res, -"Please provide some details")
    Command.addFieldValue(res, -"name", name -?: -"", -"text-single",
        -"Your name")
    return res
}

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.



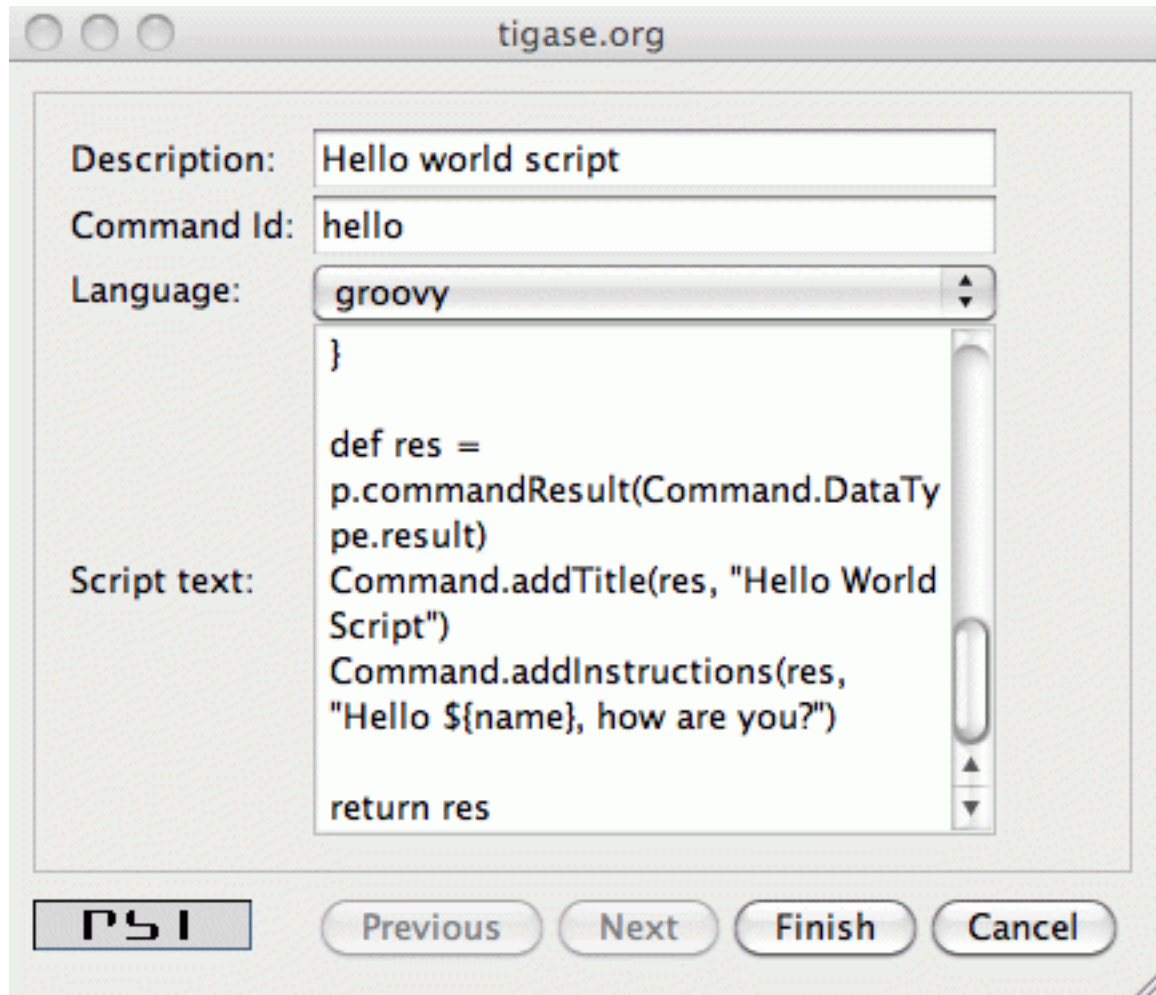
Please note, in this case the packet sent back to the user is of type `form` instead of `result`. The practical difference is that the type `result` displays only **OK** button which when pressed doesn't send any data to the server. The form packet displays more buttons - **Finish** and **Cancel**. Whichever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

This 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 to 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>] article.
- 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 have entered your name and clicked the "Finish" button you will see another window with a greeting message along 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 and remove remove them 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.

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.

## Tigase Scripting Version 4.4.x Update for Administrators

Scripting functionality is quite useful in 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.

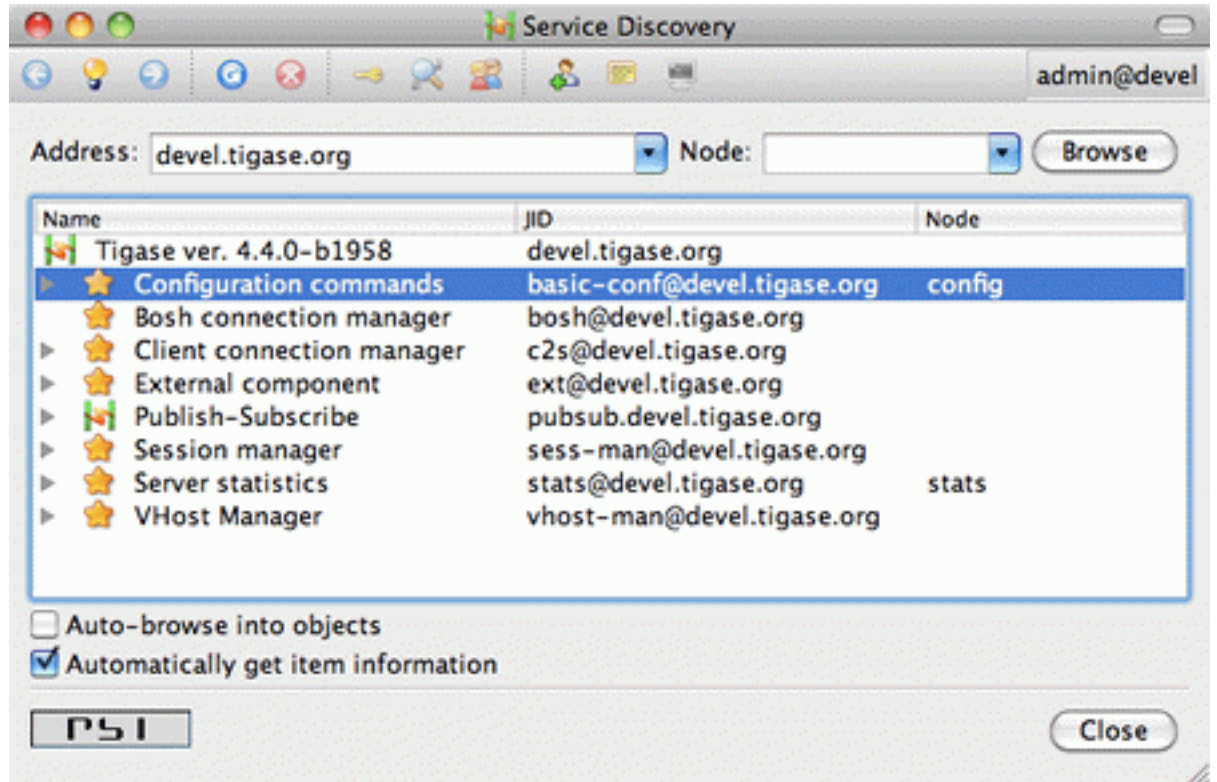
In earlier versions of the Tigase server scripting capabilities was available only in the session manager component while it might be very useful in many other places - connection managers, MUC, PubSub, VHostManager and what even more important in completely new, custom components created for specific 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.

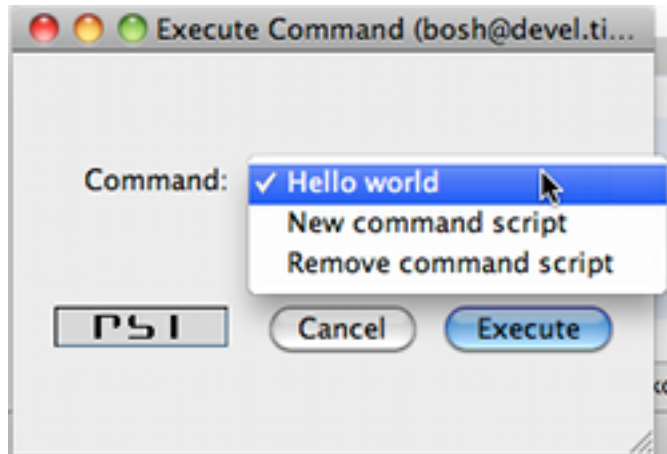


There are no command on the list. They are hidden from the main service discovery list. You can see on the list only the server main components.

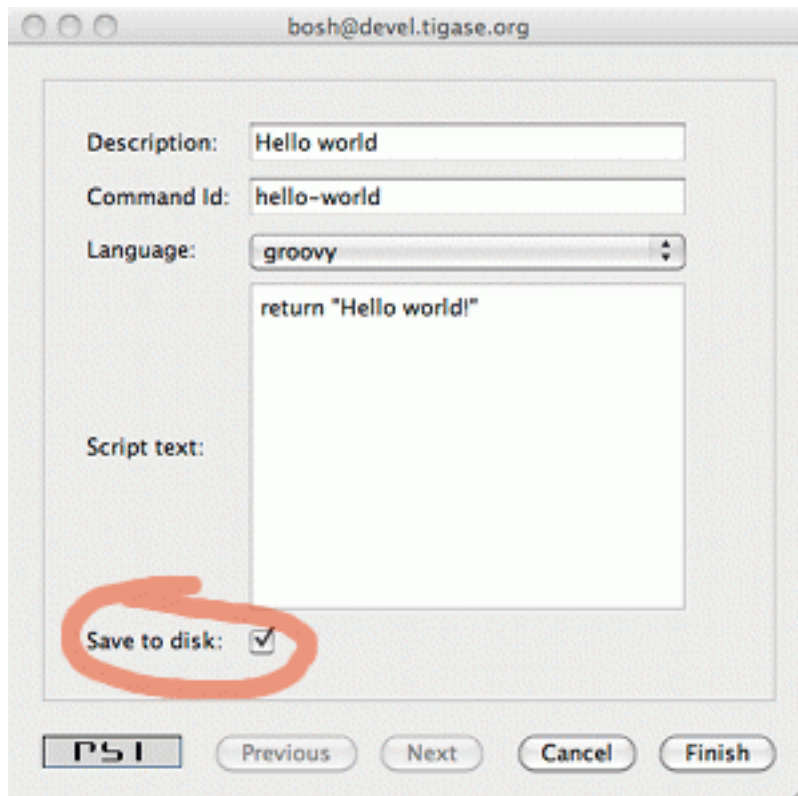
This had to be done for many reasons. One of them is, obviously, the cleaner access to the main server stuff. Another, probably more important, is to avoid a long list of commands for different components mixed together. Commands for different components can have the same name/description and they can even do similar things but they are executed on a different server component. To avoid any confusion and minimize 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.

## Tigase and Python

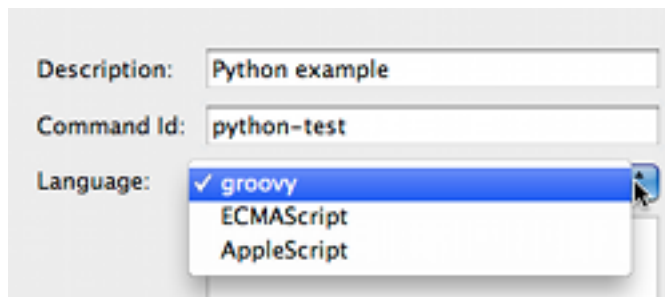
This article describes how to get Python working as a scripting language for ad-hoc commands in Tigase server. The first part is installation, and the second shows a few code examples with explanation of the differences between Python usage and some other languages.

*Please note, we are not a Python developer, and by no means this is Python development guide. 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, installation is extremely simple: just copy the file attached to this article to your Tigase installation, to the `libs/` directory. Restart the server and you are ready to start scripting and executing Python.

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.



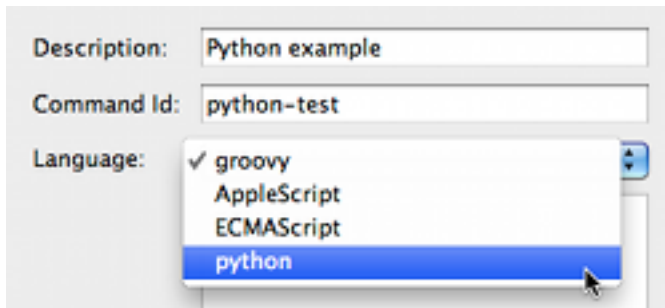
The best way to go is to use Jython implementation. It works very well within JVM and more importantly, perfectly integrates with Tigase server. Tigase server is tested with **Jython-2.2.1** and is confirmed to work fine. Version **Jython-2.5.1** is recommended however, and all the examples are executed 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, find `jython.jar` file in the directory where you installed Jython. 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 which can be downloaded from the 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 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 the introductory article for scripting in Tigase server. There were some minor changes in Tigase 4.4.0 and later versions, so please have a look at the article describing new elements as well.

An example of a 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 minor difference, the rest part of scripting in Python for the Tigase administrator commands is the same as 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 another article [[http://docs.tigase.org/tigase-server/snapshot/Development\\_Guide/html\\_chunk/cil6.html](http://docs.tigase.org/tigase-server/snapshot/Development_Guide/html_chunk/cil6.html)], 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.

---

# Chapter 12. Appendix I - Statistics description

Statistics are divided between data sources, components and processors. You may see the same statistics collected for multiple components which are defined in common components section. Note that statistics are defined by `{component}/statistic` so if you wanted Max queue size on pubsub, you would look for `pubsub/Max queue size`. Statistics will not be provided by components that are not enabled.

## Data source statistics

Data sources used to access data storages such as JDBC (databases) or MongoDB provide statistics related to stability of connections to data storage and number of open connections.

Statistics Name	Description	Statistics Level	Format	List of Possible Statistics
Number of active data sources	Number of defined and active data sources (i.e. connection pools). This is not a number of connections to data sources as it varies and is listed separately for every defined data source.	FINE	Integer	<code>dataSource/Number of data sources</code>
Number of connections for <code>{dataSource-Name}</code>	Number of connections for defined data source.	FINE	String	<code>dataSource/{dataSource-Name}/uri</code>
Number of failed reconnections for <code>{dataSource-Name}</code>	Number of reconnections that has failed since start to the defined data source.	FINE	Integer	<code>dataSource/{dataSource-Name}/failed reconnections</code>
Number of reconnections for <code>{dataSource-Name}</code>	Number of reconnections for defined data source.	FINE	Integer	<code>dataSource/{dataSource-Name}/reconnections</code>
URI of <code>{dataSource-Name}</code>	Returns URI of defined data source.	FINE	String	<code>dataSource/{dataSource-Name}/uri</code>

## User repository statistics of {repo}

For every `{method}` declared in `UserRepository` we gather execution statistics. This statistics are collected separately for every data source for which user repository is defined.

Statistics Name	Description	Statistics Level	Format	List of Possible Statistics
Average processing time of {method}	Average time taken by call of {method} for this data source since creation of data source (most likely from server start-up). It includes time taken by calls which thrown exception, etc.	FINE	Integer	userRepository/{repo}/{method}/Average processing time
Number of exceptions of a {method}	Number of exceptions the specified method has caused	FINE	Integer	userRepository/{repo}/{method}/Exceptions during execution
Number of exceptions of a {method} in last {interval}	Number of exceptions the specified method has caused within the specified interval	FINEST	Integer	userRepository/{repo}/{method}/Executions last {interval}
Number of executions of a {method}	Number of times specified method has been executed	FINE	Integer	userRepository/{repo}/{method}/Executions

## Auth repository statistics of {repo}

For every {method} declared in AuthRepository we gather execution statistics. This statistics are collected separately for every data source for which authentication repository is defined.

Statistics Name	Description	Statistics Level	Format	List of Possible Statistics
Average processing time of {method}	Average time it takes to process {method}.	FINE	Integer	authRepository/{repo}/{method}/Average processing time
Number of exceptions of {method}	Number of times {method} has caused an exception.	FINE	Integer	authRepository/{repo}/{method}/Exceptions during execution

Statistics Name	Description	Statistics Level	Format	List of Possible Statistics
Number of exceptions of {method} in last {interval}	Number of times {method} has caused an exception within the specified interval.	FINEST	Integer	authRepository/{repo}/{method}/Executions last {interval}
Number of executions of {method}	Number of times {method} has been executed.	FINE	Integer	authRepository/{repo}/{method}/Executions

## Statistics common to custom {compname} component repositories

These statistics may be found in many components which are using repository implementations created just for them. An example of such components may be:

amp	with msgBroadcastRepository as {repo} name,
message-archive	with repositoryPool as a {repo} name,
muc	with muc-dao as a {repo} name,
pubsub	with dao as a {repo} name,
sess-man	with msgRepository as a {repo} name

For custom component repositories we gather statistics in a same way as we do for user and authorization repositories. Statistics are collected on per {method} basis separately for every data source ({dataSourceName}) for which repository is defined.

Statistics Name	Description	Statistics Level	Format	List of Possible Statistics
Average processing time of {method}	Average time it takes to process {method}.	FINE	Integer	{compname}/{repo}/{dataSourceName}/{method}/Average processing time
Number of exceptions of a {method}	Number of exceptions {method} has caused.	FINE	Integer	{compname}/{repo}/{dataSourceName}/{method}/Exceptions during execution
Number of executions of a {method}	Number of times {method} has been executed.	FINE	Integer	{compname}/{repo}/{dataSourceName}

Statistics Name	Description	Statistics Level	Format	List of Possible Statistics
				Name} / {method} / Executions

## Statistics common to components

These statistics may be found in multiple components and may be seen multiple times. For example both s2s and c2s will have Bytes received statistic, so each can be found the following way:

```
s2s/Bytes received
c2s/Bytes received
```

Statistics Name	Description	Statistics Level	Format	Available {field}	List of Possible Statistics
add-script last {interval}	The number of times that add-script adhoc command has been run within the last interval.	FINEST	Integer	hour minute second	{compname} / adhoc-command {compname} / adhoc-command {compname} / adhoc-command
add-script/Average processing time	The average processing time add-script takes to complete.	FINEST	Integer		add-script/Average processing time
Average processing time on last 100 runs [ms]	The average processing time in milliseconds for all commands and scripts for this component over the last 100 times component is called. This number will populate with less than 100 runs, and will continue averaging until 100 runs happens, at that point, it's the most recent 100 instances. This statistic will reset every time the server	FINEST	Integer		{compname} / Average processing time

Statistics Name	Description	Statistics Level	Format	Available {field}	List of Possible Statistics
	shuts down or restarts.				
Bytes received	The total number of bytes that the component has received during the current server instance. This statistic resets at server shutdown or restart.	FINE or FINEST	Integer		{compname}/Bytes received
Bytes sent	The total number of bytes that the component has sent during the current server instance. This statistic resets at server shutdown or restart.	FINE or FINEST	Integer		{compname}/Bytes sent
del-script last {interval}	The number of times that del-script adhoc command has been run within the last interval.	FINEST	Integer	hour minute second	{compname}/adhoc-command {compname}/adhoc-command {compname}/adhoc-command
del-script Average processing time	The average time in ms, returned as an integer, it takes for del-script to execute.	FINEST	Integer		{compname}/adhoc-command
Last {interval} packets	The number of packets that have been handled by this component in the last interval.	FINEST	Integer	hour minute second	{compname}/last hour packets {compname}/last minute packets {compname}/last second packets
List-commands last {interval}	The number of list-commands requests sent to the component in the last interval.	FINEST	Integer	hour minute second	{compname}/list-commands {compname}/list-commands {compname}/list-commands



Statistics Name	Description	Statistics Level	Format	Available {field}	List of Possible Statistics
List-commands Average processing time	The average time in ms, returned as an integer, it takes for list-commands to execute on this component.	FINEST	Integer		{compname}/list-commands
{IN/OUT/Total} queue overflow	The number of times the in or out queue has overflowed for this component. That is there are more packets queues than the max queue size. A total statistic is also available that combines both results.	FINEST	Integer		{compname}/IN queue overflow {compname}/OUT queue overflow {compname}/Total queue overflow
{in/out} queue wait: {priority}	The number of packets with {priority} priority currently in the incoming or outgoing queue.	FINEST	Integer	SYSTEM CLUSTER HIGH NORMAL LOW PRESENCE LOWEST	{compname}/In queue wait: SYSTEM {compname}/In queue wait: CLUSTER {compname}/In queue wait: HIGH {compname}/In queue wait: NORMAL {compname}/In queue wait: LOW {compname}/In queue wait: PRESENCE {compname}/In queue wait: LOWEST {compname}/Out queue wait: SYSTEM {compname}/Out queue wait: CLUSTER {compname}/Out queue wait: HIGH {compname}/Out queue wait: NORMAL {compname}/Out queue wait: LOW {compname}/Out queue wait: PRESENCE {compname}/Out queue wait: LOWEST
{IN/OUT}_QUEUE processed {type}	The number of stanzas of different types that have been processed VIA the In or Out Queue of this component. This number will reset at the end of the server instance. Each component will have a list of the dif-	FINER	Integer	# messages presences cluster other IQ no XMLNS IQ http://jabber.org/protocol/discovery IQ bind IQ jabber:iq:roster IQ session IQ vCard IQ command IQ jabber:iq:private IQ http://jabber.org/protocol/discovery total IQ	{compname}/IN_QUEUE processed # {compname}/IN_QUEUE processed messages {compname}/IN_QUEUE processed presences {compname}/IN_QUEUE processed cluster {compname}/IN_QUEUE processed other {compname}/IN_QUEUE processed IQ no XMLNS {compname}/IN_QUEUE processed IQ http://jabber.org/protocol/discovery {compname}/IN_QUEUE processed IQ bind {compname}/IN_QUEUE processed IQ jabber:iq:roster {compname}/IN_QUEUE processed IQ session {compname}/IN_QUEUE processed IQ vCard {compname}/IN_QUEUE processed IQ command {compname}/IN_QUEUE processed IQ jabber:iq:private {compname}/IN_QUEUE processed IQ http://jabber.org/protocol/discovery {compname}/IN_QUEUE processed total IQ

Statistics Name	Description	Statistics Level	Format	Available {field}	List of Possible Statistics
	ferent types of stanzas it can process.				{compname}/OUT_QUEUE pro {compname}/OUT_QUEUE pro {compname}/OUT_QUEUE pro {compname}/OUT_QUEUE pro {compname}/OUT_QUEUE pro {compname}/OUT_QUEUE pro {compname}/OUT_QUEUE pro {compname}/OUT_QUEUE pro {compname}/OUT_QUEUE pro {compname}/OUT_QUEUE pro {compname}/OUT_QUEUE pro {compname}/OUT_QUEUE pro
	NOTE: Several statistics are only available from statistics component, shutdown thread will ONLY print the following: messages, presences, cluster, other, IQ no XLMNS, total IQ.				
max queue size	The maximum number of items allowed in the packet queue for this component.	FINEST	Integer		{compname}/max queue si
Open Connections	The number of open connections to the component.	INFO/FINEST	Integer		{compname}/Open connect.
Packets received	The total number of packets received by the component from external sources in the current instance. This number resets at server shutdown or restart.	FINE	Integer		{compname}/Packets rece
Packets sent	The total number of packets sent by the component in the current instance. This number resets at server shutdown or restart.	FINE	Integer		{compname}/Packets sent

Statistics Name	Description	Statistics Level	Format	Available {field}	List of Possible Statistics
Processed packets thread: {in/out}	How many packets have been processed in and out by each processing thread.	FINEST	Integer		{compname}/Processed packets thread: {in/out}
	Statistics will provide an array for each processor, listed from 0, 1, 2, 3 etc.. Let's say that we have 4 threads set for ws2s, a list will be seen like this: ws2s/Processed packets thread: IN=[2, 6, 4, 2] ws2s/Processed packets thread: OUT=[8, 0, 1, 3] ws2s/Processed packets thread (outliers) IN=mean: 79.0, deviation: 441, outliers: [in_10-ws2s: 2359] ws2s/Processed packets thread (outliers) OUT=mean: 16.5, deviation: 23.2058941, outliers: [out_ws2s: 80] Note that the processor array will only have as many threads as the component has as defined in {compname}/Processing threads.				
processing threads	The number of threads provided for the particular component.	FINER	Integer		{compname}/processing threads
stream-error-counter	The number of errors counted during the operation of the server for this component. Will only be available if stream-error-counter is enabled in config.tdsl, otherwise will be 0.	FINE	Integer		{compname}/processors/stream-error-counter
Socket overflow	The number of times that this component has experienced socket overflow and had to drop packets. This does not include the number of dropped packets.	FINEST	Integer		{compname}/Socket overflow
Total {in/out} queues wait	The number of packets in the inbound or outbound queue	FINEST	Integer		{compname}/Total in queue {compname}/Total out queue

Statistics Name	Description	Statistics Level	Format	Available {field}	List of Possible Statistics
	that are currently waiting to be sent. This includes packets of all types. This is an instant statistics, in that the number in queue is only as many in the queue the moment statistics are gathered.				
Total queue wait	A combined total of Total in queue wait and Total out queue wait statistics for this component.	FINEST	Integer		{compname}/Total queue v
Total queues wait	A combined total of all component queue wait statistics.	FINEST	Integer		Total queues wait
Total queues overflow	The number of times the component packet wait queue has overflowed and had to drop packets. This statistic does not keep track of the number of dropped packets.	FINEST	Integer		{compname}/Total queues
Total/Total queues overflow	The combined total of all queue overflow statistics for all components.	FINEST	Integer		total/Total queues overf
Waiting to send	The number of packets in the component's queue that are waiting to be sent. This num-	FINEST	Integer		{compname}/Waiting to s

Statistics Name	Description	Statistics Level	Format	Available {field}	List of Possible Statistics
	ber will usually be 0 however it will grow if a large number of packets are jamming up your system, or your queue sizes are set too low.				
Watchdog runs	The number of times watchdog has been run on this component to check for stale connections.	FINER	Integer		{compname}/Watchdog runs
Watchdog stopped	The number of times watchdog identified and closed a connection it has found to be stale according to the settings in <code>config.tdsl</code> or by the defaults defined in this section.	FINER	Integer		{compname}/Watchdog stopped
Watchdog tests	The number of times watchdog has found a potential stale connection and has conducted a test to determine whether or not to close the connection. This is per component in the current server instance.	FINER	Integer		{compname}/Watchdog tests

# Component statistics

## AMP

No exclusive amp specific statistics

## bosh

Statistics Name	Description	Statistics Level	Format	Available {field}	List of Possible Statistics
Bosh sessions	The number of currently open and running BOSH sessions to the server.	FINEST	Integer		bosh/Bosh sessions
pre-bind session last {interval}	The number of times the pre-bind-session command has been executed within the last specified interval.	FINEST	Integer	hour minute second	bosh/adhoc-command/pre-l bosh/adhoc-command/pre-l bosh/adhoc-command/pre-l
pre-bind-sessions/Average processing time	The average time in ms, returned as an integer, it takes for pre-bind-session to execute.	FINEST	Integer		bosh/adhoc-command/pre-l

## c2s

No exclusive c2s specific statistics.

## cl-comp

Statistics Name	Description	Statistics Level	Format	Available {field}	List of Possible Statistics
adhoc-command/cluster-nodes-list last {interval}	The number of times per interval that the cluster-nodes-list command has been executed.	FINEST	Integer	hour minute second	cl-comp/adhoc-command/c: cl-comp/adhoc-command/c: cl-comp/adhoc-command/c:
adhoc-command/cluster-nodes-list/Average processing time	The average time in ms, returned as an integer, it takes for cluster-	FINEST	Integer		cl-comp/adhoc-command/c:

	ter-nodes-list to execute.					
adhoc-command/force-stop-service last {interval}	The number of times per interval that the force-stop-service command has been executed.	FINEST	Integer	hour minute second	cl-comp/adhoc-command/f cl-comp/adhoc-command/f cl-comp/adhoc-command/f	
Adhoc-command/force-stop-ser-vice/Average processing time	The average time in ms, returned as an integer, it takes for force-stop-ser-vice to execute.	FINEST	Integer		cl-comp/adhoc-command/f	
adhoc-command/ser-vice-keys last {interval}	The number of times per interval that the ser-vice-keys command has been executed.	FINEST	Integer	hour minute second	cl-comp/adhoc-command/s cl-comp/adhoc-command/s cl-comp/adhoc-command/s	
Adhoc-command/ser-vice-keys/Average processing time	The average time in ms, returned as an integer, it takes for ser-vice-keys to execute.	FINEST	Integer		cl-comp/adhoc-command/s	
adhoc-command/sim-serv-stopped {interval}	The number of times per interval that the sim-serv-stopped command has been executed.	FINEST	Integer	hour minute second	cl-comp/adhoc-command/s cl-comp/adhoc-command/s cl-comp/adhoc-command/s	
Adhoc-command/sim-serv-stopped/Average processing time	The average time in ms, returned as an integer, it takes for sim-serv-stopped to execute.	FINEST	Integer		cl-comp/adhoc-command/s	
Average compression ratio	The average compression ratio of data sent to other clusters	FINE	Float		cl-comp/Average compres	

	during the session.					
Average de-compression ratio	The average compression ratio of data received from other clusters during the session.	FINE	Float		cl-comp/Average decompre	
Known cluster nodes	The number of cluster nodes currently connected to the server.	INFO	Integer		cl-comp/Known cluster n	
Last {interval} disconnects	The number of cluster disconnections within the specified interval.	FINE	Comma Separated Array	day hour	cl-comp/Last day disconn cl-comp/Last hour disconn	
For day, each array is the number of disconnections each hour, most recent first. For hour each array is the number of disconnections each minute, most recent first.						
Service connected time-outs	The number of time-outs during connection initialization of cluster nodes.	FINEST	Integer		cl-comp/Service connecte	
Total disconnects	The number of clusters that have disconnected during the current session.	FINEST	Integer		cl-comp/Total disconnect	

## eventbus

No exclusive eventbus specific statistics.

## message-archive

Statistics Name	Description	Statistics Level	Format	Available {field}	List of Possible Statistics
Removal time of expired messages (avg)	The average amount of time in milliseconds it takes to remove expired messages from the repository. This includes manual and au-	FINE	Integer		message-archive/Removal



	automatic removal of messages.				
--	--------------------------------	--	--	--	--

## message-router

Statistics Name	Description	Statistics Level	Format	Available {field}	List of Possible Statistics
CPUs no	The number of CPUs available on the host machine.	FINEST	Integer		message-router/CPUs no
CPU Usage	% of available CPU power used by Tigase Server at the moment statistics are taken.	FINE	Float/String		message-router/CPU usage message-router/CPU usage
	Two formats are available for CPU usage: A float integer which expresses a long decimal available from CPU Usage [%], and a string which provides a rounded number with a % sign from CPU usage.				
Free Heap	The amount of heap memory that is available for use, expressed in KB.	FINE	String		message-router/Free Heap
Free NonHeap	The amount of non-heap memory that is available for use, expressed in KB.	FINE	String		message-router/Free NonH
HEAP usage [%]	Total percent of HEAP memory in use by Tigase.	FINE	Float		message-router/HEAP usage
Local hostname	The local hostname of the physical server.	INFO	String		message-router/Local hos
Load average	The average system load for the previous minute. The way in which the load average is calculated is operating system specific but is typically a damped time-dependent average.	FINE	Float		message-router/Load aver

Max Heap mem	Maximum amount of heap memory available as defined by JAVA_OPTIONS in tigas.conf, in Kb.	INFO	String		message-router/Max Heap
Max NonHeap mem	Maximum amount of non-heap memory available as defined by JAVA_OPTIONS in tigas.conf, in Kb.	FINE	String		message-router/Max NonHeap
NONHEAP Usage [%]	Total amount of NONHEAP memory in use expressed as a percentage.	FINE	Float		message-router/NONHEAP u
Threads count	The total number of processing threads available across all components.	FINEST	Integer		message-router/Threads c
Uptime	The total amount of time the server has been online for this session.	INFO	String		message-router/Uptime
Used Heap	The amount of heap memory in use in KB.	INFO	String		message-router/Used Heap
Used NonHeap	The amount of non-heap memory in use shown in KB.	FINE	String		message-router/Used NonH

## monitor

Statistics Name	Description	Statistics Level	Format	Available {field}	List of Possible Statistics
adhoc-command/load-errors last {interval}	The number of times per interval that the load-errors command has been executed.	FINEST	Integer	hour minute second	monitor/adhoc-command/1 monitor/adhoc-command/1 monitor/adhoc-command/1
Adhoc-command/load-er	The average time in ms, re-	FINEST	Integer		monitor/adhoc-command/1

rors/Average processing time	turned as an integer, it takes for load-er-rors to execute.				
------------------------------	---	--	--	--	--

## muc

Statistics Name	Description	Statistics Level	Format	Available {field}	List of Possible Statistics
adhoc-command/re-move-room last {interval}	The number of times per interval that the remove-room command has been executed.	FINEST	Integer	hour minute second	monitor/adhoc-command/re monitor/adhoc-command/re monitor/adhoc-command/re
Adhoc-command/re-move-room/Average processing time	The average time in ms, returned as an integer, it takes for re-move-room to execute.	FINEST	Integer		monitor/adhoc-command/re
adhoc-command/default-room-config last {interval}	The number of times per interval that the default-room-command command has been executed.	FINEST	Integer	hour minute second	muc/adhoc-command/default muc/adhoc-command/default muc/adhoc-command/default
Adhoc-command/default-room-config/Average processing time	The average time in ms, returned as an integer, it takes for default-room-config to execute.	FINEST	Integer		muc/adhoc-command/default

## proxy

Statistics Name	Description	Statistics Level	Format	List of Possible Statistics
Average transfer size in KB	Average size of packets sent through the proxy component during the current session.	FINEST	Integer	proxy/Average transfer size
KBytes transferred	Total number of Kb transferred through the proxy component.	FINEST	Integer	proxy/KBytes transferred

Open streams	Number of currently open proxy streams.	FINEST	Integer	proxy/Open streams
Transfers completed	Number of specific transfers completed through proxy component.	FINEST	Integer	proxy/Transfers completed

## pubsub

Statistics Name	Description	Statistics Level	Format	Available {field}	List of Possible Statistics
Added new nodes	The total number of new nodes that has been added in the current server instance. This statistic is reset when the server resets.	FINEST	Integer		pubsub/Added new nodes
adhoc-command/delete-item last {interval}	The number of times per interval that the delete-item command has been executed.	FINEST	Integer	hour minute second	pubsub/adhoc-command/delete-item last {interval}
adhoc-command/delete-item/Average processing time	The average time in ms, returned as an integer, it takes for delete-item to execute.	FINEST	Integer		pubsub/adhoc-command/delete-item/Average processing time
adhoc-command/delete-node last {interval}	The number of times per interval that the delete-node command has been executed.	FINEST	Integer	hour minute second	pubsub/adhoc-command/delete-node last {interval}
adhoc-command/delete-node/Average processing time	The average time in ms, returned as an integer, it takes for delete-node to execute.	FINEST	Integer		pubsub/adhoc-command/delete-node/Average processing time
adhoc-command/list-items last {interval}	The number of times per interval that the	FINEST	Integer		pubsub/adhoc-command/list-items last {interval}

	list-items command has been executed.					
ad hoc-command/list-items/ Average processing time	The average time in ms, re- turned as an in- teger, it takes for list- items to exe- cute.	FINEST	Integer		pubsub/ad hoc-command/li	
ad hoc-command/list-nodes last {interval}	The number of times per in- terval that the list-nodes command has been executed.	FINEST	Integer		pubsub/ad hoc-command/li pubsub/ad hoc-command/li pubsub/ad hoc-command/li	
ad hoc-command/list- nodes/Average processing time	The average time in ms, re- turned as an in- teger, it takes for list- nodes to exe- cute.	FINEST	Integer		pubsub/ad hoc-command/li	
ad hoc-command/pub- lish-item last {interval}	The num- ber of times per interval that the pub- lish-item command has been executed.	FINEST	Integer		pubsub/ad hoc-command/pub pubsub/ad hoc-command/pub pubsub/ad hoc-command/pub	
ad hoc-command/pub- lish-item/Aver- age processing time	The average time in ms, returned as an integer, it takes for pub- lish-item to execute.	FINEST	Integer		pubsub/ad hoc-command/pub	
ad hoc-command/re- trieve-item last {interval}	The number of times per inter- val that the re- trieve-item command has been executed.	FINEST	Integer	hour minute second	pubsub/ad hoc-command/re pubsub/ad hoc-command/re pubsub/ad hoc-command/re	
ad hoc-command/re- trieve-item/Av- erage process- ing time	The average time in ms, returned as an integer, it takes for re- trieve-item to execute.	FINEST	Integer		pubsub/ad hoc-command/re	

AdHocConfig- CommandMod- ule last {inter- val}	The number of times per inter- val that the Ad- HocConfig- Command- Module com- mand has been executed.	FINEST	Integer	hour minute second	pubsub/AdHocConfigComman pubsub/AdHocConfigComman pubsub/AdHocConfigComman
AdHocConfig- CommandMod- ule/Average processing time	The average time in ms, re- turned as an in- teger, it takes for AdHoc- ConfigCom- mandModule to execute.	FINEST	Integer		pubsub/AdHocConfigComman
Affiliations count (in cache)	The total num- ber of pubsub affiliations that are resident in cache mem- ory. Affilia- tions include JIDs that are one of the fol- lowing; Own- er, Publisher, Publish-Only, Member, None, Outcast. This may not reflect total pubsub affiliations in repository.	FINEST	Integer		pubsub/Affiliations coun
Average DB write time [ms]	The average time of all DB writes from PubSub com- ponent. Aver- age is calculat- ed using two other statistics: (Total writing time / Database writes)	FINEST	Integer		pubsub/Average DB write
cache/hits last {interval}	The number of times the cache has achieved a hit within the last interval. A hit is when a request for	FINEST	Integer	hour minute second	pubsub/cache/hits last h pubsub/cache/hits last r pubsub/cache/hits last s

	information is matched to data that is inside the cache memory.					
cache/hit-miss ratio per {interval}	The ratio of cache hits to cache misses over the specified period. A cache hit is when a request for information from the cache is matched with information in the cache. A miss is when that information request cannot find a match in cache. A miss only indicates that that information was not found in the cache, not that it is not in the repository.	FINE	Float	hour minute	pubsub/cache/hit-miss ratio pubsub/cache/hit-miss ratio	
cache/requests last {interval}	The number of memory cache requests made within the last interval.	FINEST	Integer	hour minute second	pubsub/cache/Requests last pubsub/cache/Requests last pubsub/cache/Requests last	
Cached nodes	The number of nodes that is currently in memory cache.	FINEST	Integer		pubsub/Cached nodes	
CapsModule	The number of times per interval that the CapsModule command has been executed.	FINEST	Integer	hour minute second	pubsub/CapsModule last pubsub/CapsModule last pubsub/CapsModule last	
CapsModule/Average processing time	The average time in ms, returned as an integer, it takes for CapsModule to execute.	FINEST	Integer		pubsub/CapsModule/Average	
db/GetNodeItems re-	The number of times GetN-	FINEST	Integer	hour minute	pubsub/db/GetNodeItems pubsub/db/GetNodeItems	

quests last {interval}	odeItems command has been run within the specified interval.			second	pubsub/db/GetNodeItems	
db/GetNodeItems/Average processing time	The average time in ms, returned as an integer, it takes for GetNodeItems to execute.	FINEST	Integer		pubsub/db/GetNodeItems	
DefaultConfigModule last {interval}	The number of times per interval that the DefaultConfigModule command has been executed.	FINEST	Integer	hour minute second	pubsub/DefaultConfigModule pubsub/DefaultConfigModule pubsub/DefaultConfigModule	
DefaultConfigModule/Average processing time	The average time in ms, returned as an integer, it takes for DefaultConfigModule to execute.	FINEST	Integer		pubsub/DefaultConfigModule	
DiscoverInfoModule last {interval}	The number of times per interval that the DiscoverInfoModule command has been executed.	FINEST	Integer		pubsub/DiscoverInfoModule pubsub/DiscoverInfoModule pubsub/DiscoverInfoModule	
DiscoverInfoModule/Average processing time	The average time in ms, returned as an integer, it takes for DiscoverInfoModule to execute.	FINEST	Integer		pubsub/DiscoverInfoModule	
DiscoverItemsModule last {interval}	The number of times per interval that the DiscoverItemsModule command has been executed.	FINEST	Integer		pubsub/DiscoverItemsModule pubsub/DiscoverItemsModule pubsub/DiscoverItemsModule	
DiscoverItemsModule/Average processing time	The average time in ms, returned as an integer, it takes	FINEST	Integer		pubsub/DiscoverItemsModule	



	for DiscoverItemsModule to execute.					
JabberVersionModule last {interval}	The number of times per interval that the JabberVersionModule command has been executed.	FINEST	Integer	hour minute second	pubsub/JabberVersionModule pubsub/JabberVersionModule pubsub/JabberVersionModule	
JabberVersionModule/Average processing time	The average time in ms, returned as an integer, it takes for JabberVersionModule to execute.	FINEST	Integer		pubsub/JabberVersionModule	
ManageAffiliationsModule last {interval}	The number of times per interval that the ManageAffiliationsModule command has been executed.	FINEST	Integer	hour minute second	pubsub/ManageAffiliationsModule pubsub/ManageAffiliationsModule pubsub/ManageAffiliationsModule	
ManageAffiliationsModule/Average processing time	The average time in ms, returned as an integer, it takes for ManageAffiliationsModule to execute.	FINEST	Integer		pubsub/ManageAffiliationsModule	
ManageSubscriptionModule last {interval}	The number of times per interval that the ManageSubscriptionModule command has been executed.	FINEST	Integer	hour minute second	pubsub/ManageSubscriptionModule pubsub/ManageSubscriptionModule pubsub/ManageSubscriptionModule	
ManageSubscriptionModule/Average processing time	The average time in ms, returned as an integer, it takes for ManageSubscriptionModule to execute.	FINEST	Integer		pubsub/ManageSubscriptionModule	

NodeConfig-Module last {interval}	The number of times per interval that the NodeConfigModule command has been executed.	FINEST	Integer	hour minute second	pubsub/NodeConfigModule pubsub/NodeConfigModule pubsub/NodeConfigModule	
NodeConfig-Module/Average processing time	The average time in ms, returned as an integer, it takes for NodeConfigModule to execute.	FINEST	Integer		pubsub/NodeConfigModule	
NodeCreate-Module last {interval}	The number of times per interval that the NodeCreateModule command has been executed.	FINEST	Integer	hour minute second	pubsub/NodeCreateModule pubsub/NodeCreateModule pubsub/NodeCreateModule	
NodeCreate-Module/Average processing time	The average time in ms, returned as an integer, it takes for NodeCreateModule to execute.	FINEST	Integer		pubsub/NodeCreateModule	
NodeDelete-Module last {interval}	The number of times per interval that the NodeDeleteModule command has been executed.	FINEST	Integer	hour minute second	pubsub/NodeDeleteModule pubsub/NodeDeleteModule pubsub/NodeDeleteModule	
NodeDelete-Module/Average processing time	The average time in ms, returned as an integer, it takes for NodeDeleteModule to execute.	FINEST	Integer		pubsub/NodeDeleteModule	
PresenceCollectorModule last {interval}	The number of times per interval that the PresenceCollectorModule command	FINEST	Integer	hour minute second	pubsub/PresenceCollector pubsub/PresenceCollector pubsub/PresenceCollector	

	has been executed.					
PresenceCollectorModule/Average processing time	The average time in ms, returned as an integer, it takes for PresenceCollectorModule to execute.	FINEST	Integer		pubsub/PresenceCollectorModule	
PendingSubscriptionModule last {interval}	The number of times per interval that the PendingSubscriptionModule command has been executed.	FINEST	Integer	hour minute second	pubsub/PendingSubscriptionModule pubsub/PendingSubscriptionModule pubsub/PendingSubscriptionModule	
PendingSubscriptionModule/Average processing time	The average time in ms, returned as an integer, it takes for PendingSubscriptionModule to execute.	FINEST	Integer		pubsub/PendingSubscriptionModule	
PresenceNotifierModule last {interval}	The number of times per interval that the PresenceNotifierModule command has been executed.	FINEST	Integer	hour minute second	pubsub/PresenceNotifierModule pubsub/PresenceNotifierModule pubsub/PresenceNotifierModule	
PresenceNotifierModule/Average processing time	The average time in ms, returned as an integer, it takes for PresenceNotifierModule to execute.	FINEST	Integer		pubsub/PresenceNotifierModule	
PublishItemModule last {interval}	The number of times per interval that the PublishItemModule command has been executed.	FINEST	Integer	hour minute second	pubsub/PublishItemModule pubsub/PublishItemModule pubsub/PublishItemModule	

	mand has been executed.					
PublishItem-Module/Average processing time	The average time in ms, returned as an integer, it takes for PublishItemModule to execute.	FINEST	Integer		pubsub/PublishItemModule	
PurgeItemsModule last {interval}	The number of times per interval that the PurgeItemsModule command has been executed.	FINEST	Integer	hour minute second	pubsub/PurgeItemsModule pubsub/PurgeItemsModule pubsub/PurgeItemsModule	
PurgeItemsModule/Average processing time	The average time in ms, returned as an integer, it takes for PurgeItemsModule to execute.	FINEST	Integer		pubsub/PurgeItemsModule	
Repository writes	Number of individual writes to Repository from the pubsub component since startup.	FINEST	Integer		pubsub/Repository writes	
RetractItemModule last {interval}	The number of times per interval that the RetractItemModule command has been executed.	FINEST	Integer	hour minute second	pubsub/RetractItemModule pubsub/RetractItemModule pubsub/RetractItemModule	
RetractItemModule/Average processing time	The average time in ms, returned as an integer, it takes for RetractItemModule to execute.	FINEST	Integer		pubsub/RetractItemModule	
RetrieveAffiliationsModule last {interval}	The number of times per interval that the RetrieveAffiliationsModule com-	FINEST	Integer	hour minute second	pubsub/RetrieveAffiliat. pubsub/RetrieveAffiliat. pubsub/RetrieveAffiliat.	

	mand has been executed.					
RetrieveAffiliationsModule/Average processing time	The average time in ms, returned as an integer, it takes for RetrieveAffiliationsModule to execute.	FINEST	Integer		pubsub/RetrieveAffiliationsModule	
RetrieveItemsModule last {interval}	The number of times per interval that the RetrieveItemsModule command has been executed.	FINEST	Integer	hour minute second	pubsub/RetrieveItemsModule pubsub/RetrieveItemsModule pubsub/RetrieveItemsModule	
RetrieveItemsModule/Average processing time	The average time in ms, returned as an integer, it takes for RetrieveItemsModule to execute.	FINEST	Integer		pubsub/RetrieveItemsModule	
RetrieveSubscriptionsModule last {interval}	The number of times per interval that the RetrieveSubscriptionsModule command has been executed.	FINEST	Integer	hour minute second	pubsub/RetrieveSubscriptionsModule pubsub/RetrieveSubscriptionsModule pubsub/RetrieveSubscriptionsModule	
RetrieveSubscriptionsModule/Average processing time	The average time in ms, returned as an integer, it takes for RetrieveSubscriptionsModule to execute.	FINEST	Integer		pubsub/RetrieveSubscriptionsModule	
SubscribeNodeModule last {interval}	The number of times per interval that the SubscribeNodeModule command has been executed.	FINEST	Integer	hour minute second	pubsub/SubscribeNodeModule pubsub/SubscribeNodeModule pubsub/SubscribeNodeModule	
SubscribeNodeModule/Average processing time	The average time in ms,	FINEST	Integer		pubsub/SubscribeNodeModule	

age processing time	returned as an integer, it takes for SubscribeNodeModule to execute.					
Subscription count (in cache)	The total number of pubsub subscriptions that are resident in cache memory. This may not reflect total pubsub subscriptions in repository.	FINEST	Integer		pubsub/Subscription count	
Total writing time	The cumulative total of time pubsub component has written to the database expressed in milliseconds.	FINEST	String (###ms)		pubsub/Total writing time	
UnsubscribeNodeModule last {interval}	The number of times per interval that the UnsubscribeNodeModule command has been executed.	FINEST	Integer	hour minute second	pubsub/UnsubscribeNodeModule last {interval}	
UnsubscribeNodeModule/Average processing time	The average time in ms, returned as an integer, it takes for UnsubscribeNodeModule to execute.	FINEST	Integer		pubsub/UnsubscribeNodeModule/Average processing time	
Update subscription calls	Number of times Subscriptions have been updated (this includes new, deleted, and edited).	FINEST	Integer		pubsub/Update subscription calls	
XmppPingModule last {interval}	The number of times per interval that the XmppPingModule command has been executed.	FINEST	Integer	hour minute second	pubsub/XmppPingModule last {interval}	

	mand has been executed.				
XmppPing-Module/Average processing time	The average time in ms, returned as an integer, it takes for XmppPingModule to execute.	FINEST	Integer		pubsub/XmppPingModule/A

## repo-factory

Statistics Name	Description	Statistics Level	Format	List of Possible Statistics
Number of data repositories	The number of data repositories set-up for this XMPP server.	FINE	Integer	repo-factory/Number of data
Repository {jdbclocation} connections count	The number of connections made to this database.	FINE	Integer	repo-factory/repository {
repository {jdbclocation} reconnections	The number of reconnections made to this database.	FINEST	Integer	repo-factory/repository {
repository {jdbclocation} failed reconnections	The number of reconnections that have failed to connect to this database.	FINEST	Integer	repo-factory/repository {

## rest

No exclusive rest specific statistics

## s2s

Statistics Name	Description	Statistics Level	Format	Available {field}	List of Possible Statistics
CIDs number	ConnectionID for the server. This may include multiple CIDs if server is running multiple vhosts.	FINEST	String		s2s/CIDs number
get-cid-connection last {interval}	The number of times get-cid-connection command has been executed	FINEST	Integer	hour minute second	s2s/adhoc-command/get-c s2s/adhoc-command/get-c s2s/adhoc-command/get-c

Statistics Name	Description	Statistics Level	Format	Available {field}	List of Possible Statistics
	within the specified interval.				
get-cid-connection/Average processing time	The average time in ms, returned as an integer, it takes for get-cid-connection to execute.	FINEST	Integer		s2s/adhoc-command/get-c-
s2s-bad-state-conns last {interval}	The number of times s2s-bad-state-conns command has been executed within the specified interval.	FINEST	Integer	hour minute second	s2s/adhoc-command/s2s-ba s2s/adhoc-command/s2s-ba s2s/adhoc-command/s2s-ba
s2s-bad-state-conns/Average processing time	The average time in ms, returned as an integer, it takes for s2s-bad-state-conns to execute.	FINEST	Integer		s2s/adhoc-command/s2s-ba
reset-bad-state-conns last {interval}	The number of times reset-bad-state-conns command has been executed within the specified interval.	FINEST	Integer	hour minute second	s2s/adhoc-command/reset- s2s/adhoc-command/reset- s2s/adhoc-command/reset-
reset-bad-state-conns/Average processing time	The average time in ms, returned as an integer, it takes for reset-bad-state-conns to execute.	FINEST	Integer		s2s/adhoc-command/reset-
Total DB keys	Total number of database keys.	FINEST	Integer		s2s/Total DB keys
Total {incoming/outgoing}	The total number of server-to-server connections, outgoing is local server connecting to	FINEST	Integer		s2s/Total incoming s2s/Total outgoing



Statistics Name	Description	Statistics Level	Format	Available {field}	List of Possible Statistics
	other servers, and incoming is connections from other servers. The results may or may not be the same.				
Total {incoming/outgoing} TLS	The total number of server-to-server connections using TLS, outgoing is local server connecting to other servers, and incoming is connections from other servers. The results may or may not be the same.	FINEST	Integer		s2s/Total incoming TLS s2s/Total outgoing TLS
Total outgoing handshaking	Total number of outgoing connections that are currently handshaking to other servers.	FINEST	Integer		s2s/Total outgoing hands
Total control waiting	Total number of connections that were manually told to wait.	FINEST	Integer		s2s/Total control waitin
Total waiting	Total number of connections that are currently waiting for response from other server.	FINEST	Integer		s2s/Total waiting

## sess-man

Statistics Name	Description	Statistics Level	Format	Available {field}	List of Possible Statistics
Active user connections	Number of user connections that are considered active. An active user	FINER	Integer		sess-man/Active user con

Statistics Name	Description	Statistics Level	Format	Available {field}	List of Possible Statistics
	is a user that has sent stanzas to the server or through the server within the last 5 minutes.				
adhoc-command/connection-time last {interval}	The number of times connection-time command has been executed within the specified interval.	FINEST	Integer	hour minute second	sess-man/adhoc-command/ sess-man/adhoc-command/ sess-man/adhoc-command/
adhoc-command/connection-time/Average processing time	The average time in ms, returned as an integer, it takes for connection-time to execute.	FINEST	Integer		sess-man/adhoc-command/
adhoc-command/http://jabber.org/protocol/admin#add-user last {interval}	The number of times admin#add-user command has been executed within the specified interval.	FINEST	Integer	hour minute second	sess-man/adhoc-command/ sess-man/adhoc-command/ sess-man/adhoc-command/
adhoc-command/http://jabber.org/protocol/admin#add-user/Average processing time	The average time in ms, returned as an integer, it takes for admin#add-user to execute.	FINEST	Integer		sess-man/adhoc-command/
adhoc-command/http://jabber.org/protocol/admin#add-user-tracker last {interval}	The number of times admin#add-user-tracker command has been executed within the specified interval.	FINEST	Integer	hour minute second	sess-man/adhoc-command/ sess-man/adhoc-command/ sess-man/adhoc-command/
adhoc-command/http://	The average time in ms,	FINEST	Integer		sess-man/adhoc-command/

Statistics Name	Description	Statistics Level	Format	Available {field}	List of Possible Statistics
jabber.org/ proto-col/admin#add-user-tracker/Average processing time	returned as an integer, it takes for admin#add-user-tracker to execute.				
adhoc-command/http://jabber.org/ proto-col/admin#announce last {interval}	The number of times admin#announce command has been executed within the specified interval.	FINEST	Integer	hour minute second	sess-man/adhoc-command/l sess-man/adhoc-command/l sess-man/adhoc-command/l
adhoc-command/http://jabber.org/ proto-col/admin#announce/Average processing time	The average time in ms, returned as an integer, it takes for admin#announce to execute.	FINEST	Integer		sess-man/adhoc-command/l
adhoc-command/http://jabber.org/ proto-col/admin#change-user-password last {interval}	The number of times admin#change-user-password command has been executed within the specified interval.	FINEST	Integer	hour minute second	sess-man/adhoc-command/l sess-man/adhoc-command/l sess-man/adhoc-command/l
adhoc-command/http://jabber.org/ proto-col/admin#change-user-password/Average processing time	The average time in ms, returned as an integer, it takes for admin#change-user-password to execute.	FINEST	Integer		sess-man/adhoc-command/l
adhoc-command/http://jabber.org/ proto-col/admin#delete-user last {interval}	The number of times admin#delete-user command has been executed within the specified interval.	FINEST	Integer	hour minute second	sess-man/adhoc-command/l sess-man/adhoc-command/l sess-man/adhoc-command/l
adhoc-command/http://	The average time in ms,	FINEST	Integer		sess-man/adhoc-command/l

Statistics Name	Description	Statistics Level	Format	Available {field}	List of Possible Statistics
jabber.org/protocol/admin#delete-user/Average processing time	returned as an integer, it takes for admin#delete-user to execute.				
adhoc-command/http://jabber.org/protocol/admin#end-user-session last {interval}	The number of times admin#end-user-session command has been executed within the specified interval.	FINEST	Integer	hour minute second	sess-man/adhoc-command/l sess-man/adhoc-command/l sess-man/adhoc-command/l
adhoc-command/http://jabber.org/protocol/admin#end-user-session/Average processing time	The average time in ms, returned as an integer, it takes for admin#end-user-session to execute.	FINEST	Integer		sess-man/adhoc-command/l
adhoc-command/http://jabber.org/protocol/admin#get-active-users last {interval}	The number of times admin#get-active-users command has been executed within the specified interval.	FINEST	Integer		sess-man/adhoc-command/l sess-man/adhoc-command/l sess-man/adhoc-command/l
adhoc-command/http://jabber.org/protocol/admin#get-active-users/Average processing time	The average time in ms, returned as an integer, it takes for admin#get-active-users to execute.	FINEST	Integer		sess-man/adhoc-command/l
adhoc-command/http://jabber.org/protocol/admin#get-active-user-num last {interval}	The number of times admin#get-active-user-num command has been exe-	FINEST	Integer	hour minute second	sess-man/adhoc-command/l sess-man/adhoc-command/l sess-man/adhoc-command/l

Statistics Name	Description	Statistics Level	Format	Available {field}	List of Possible Statistics
	cuted within the specified interval.				
ad hoc-command/http://jabber.org/protocol/admin#get-active-user-num/Average processing time	The average time in ms, returned as an integer, it takes for admin#get-active-user-num to execute.	FINEST	Integer		sess-man/ad hoc-command/l
ad hoc-command/http://jabber.org/protocol/admin#get-idle-users last {interval}	The number of times admin#get-idle-users command has been executed within the specified interval.	FINEST	Integer	hour minute second	sess-man/ad hoc-command/l sess-man/ad hoc-command/l sess-man/ad hoc-command/l
ad hoc-command/http://jabber.org/protocol/admin#get-idle-users/Average processing time	The average time in ms, returned as an integer, it takes for admin#get-idle-users to execute.	FINEST	Integer		sess-man/ad hoc-command/l
ad hoc-command/http://jabber.org/protocol/admin#get-idle-users-num last {interval}	The number of times admin#get-idle-users-num command has been executed within the specified interval.	FINEST	Integer	hour minute second	sess-man/ad hoc-command/l sess-man/ad hoc-command/l sess-man/ad hoc-command/l
ad hoc-command/http://jabber.org/protocol/admin#get-idle-users-num/Average processing time	The average time in ms, returned as an integer, it takes for admin#get-idle-users-num to execute.	FINEST	Integer		sess-man/ad hoc-command/l
ad hoc-command/http://jabber.org/	The number of times admin#get-	FINEST	Integer	hour minute second	sess-man/ad hoc-command/l sess-man/ad hoc-command/l sess-man/ad hoc-command/l

Statistics Name	Description	Statistics Level	Format	Available {field}	List of Possible Statistics
proto-col/admin#get-online-users-list last {interval}	on-line-users-list command has been executed within the specified interval.				
adhoc-command/http://jabber.org/ proto-col/admin#get-online-users-list/Average processing time	The average time in ms, returned as an integer, it takes for admin#get-online-users-list to execute.	FINEST	Integer		sess-man/adhoc-command/l
adhoc-command/http://jabber.org/ proto-col/admin#get-top-active-users last {interval}	The number of times admin#get-top-active-users command has been executed within the specified interval.	FINEST	Integer	hour minute second	sess-man/adhoc-command/l sess-man/adhoc-command/l sess-man/adhoc-command/l
adhoc-command/http://jabber.org/ proto-col/admin#get-top-active-users/Average processing time	The average time in ms, returned as an integer, it takes for admin#get-top-active-users to execute.	FINEST	Integer		sess-man/adhoc-command/l
adhoc-command/http://jabber.org/ proto-col/admin#get-registered-users-list last {interval}	The number of times admin#get-registered-users-list command has been executed within the specified interval.	FINEST	Integer	hour minute second	sess-man/adhoc-command/l sess-man/adhoc-command/l sess-man/adhoc-command/l
adhoc-command/http://jabber.org/ proto-	The average time in ms, returned as an integer,	FINEST	Integer		sess-man/adhoc-command/l

Statistics Name	Description	Statistics Level	Format	Available {field}	List of Possible Statistics
col/admin#get-regis-tered-users-list/ Average processing time	it takes for admin#get-regis-tered-users-list to execute.				
adhoc-command/http://jabber.org/protocol/admin#get-user-roster last {interval}	The number of times admin#get-user-roster command has been executed within the specified interval.	FINEST	Integer	hour minute second	sess-man/adhoc-command/l sess-man/adhoc-command/l sess-man/adhoc-command/l
adhoc-command/http://jabber.org/protocol/admin#get-user-roster/Average processing time	The average time in ms, returned as an integer, it takes for admin#get-user-roster to execute.	FINEST	Integer		sess-man/adhoc-command/l
adhoc-command/http://jabber.org/protocol/admin#remove-user last {interval}	The number of times admin#remove-user command has been executed within the specified interval.	FINEST	Integer	hour minute second	sess-man/adhoc-command/l sess-man/adhoc-command/l sess-man/adhoc-command/l
adhoc-command/http://jabber.org/protocol/admin#remove-user/Average processing time	The average time in ms, returned as an integer, it takes for admin#remove-user to execute.	FINEST	Integer		sess-man/adhoc-command/l
adhoc-command/http://jabber.org/protocol/admin#user-stats last {interval}	The number of times admin#user-stats command has been executed within the specified interval.	FINEST	Integer	hour minute second	sess-man/adhoc-command/l sess-man/adhoc-command/l sess-man/adhoc-command/l
adhoc-command/http://	The average time in ms,	FINEST	Integer		sess-man/adhoc-command/l

Statistics Name	Description	Statistics Level	Format	Available {field}	List of Possible Statistics
jabber.org/ proto-col/admin#user-stats/Average processing time	returned as an integer, it takes for admin#user-stats to execute.				
adhoc-command/get-user-info last {interval}	The number of times get-user-info command has been executed within the specified interval.	FINEST	Integer	hour minute second	sess-man/adhoc-command/g sess-man/adhoc-command/g sess-man/adhoc-command/g
adhoc-command/get-user-info/Average processing time	The average time in ms, returned as an integer, it takes for get-user-info to execute.	FINEST	Integer		sess-man/adhoc-command/g
adhoc-command/modify-user last {interval}	The number of times modify-user command has been executed within the specified interval.	FINEST	Integer	hour minute second	sess-man/adhoc-command/r sess-man/adhoc-command/r sess-man/adhoc-command/r
adhoc-command/modify-user/Average processing time	The average time in ms, returned as an integer, it takes for modify-user to execute.	FINEST	Integer		sess-man/adhoc-command/r
adhoc-command/oauth-credentials last {interval}	The number of times oauth-credentials command has been executed within the specified interval.	FINEST	Integer	hour minute second	sess-man/adhoc-command/o sess-man/adhoc-command/o sess-man/adhoc-command/o
adhoc-command/oauth-credentials/Average processing time	The average time in ms, returned as an integer, it takes for oauth-credentials	FINEST	Integer		sess-man/adhoc-command/o



Statistics Name	Description	Statistics Level	Format	Available {field}	List of Possible Statistics
	tials to execute.				
adhoc-command/roster-fixer last {interval}	The number of times roster-fixer command has been executed within the specified interval.	FINEST	Integer	hour minute second	sess-man/adhoc-command/ sess-man/adhoc-command/ sess-man/adhoc-command/
adhoc-command/roster-fixer/Average processing time	The average time in ms, returned as an integer, it takes for roster-fixer to execute.	FINEST	Integer		sess-man/adhoc-command/
adhoc-command/roster-fixer-cluster last {interval}	The number of times roster-fixer-cluster command has been executed within the specified interval.	FINEST	Integer	hour minute second	sess-man/adhoc-command/ sess-man/adhoc-command/ sess-man/adhoc-command/
adhoc-command/roster-fixer-cluster/Average processing time	The average time in ms, returned as an integer, it takes for roster-fixer-cluster to execute.	FINEST	Integer		sess-man/adhoc-command/
adhoc-command/user-domain-perm last {interval}	The number of times user-domain-perm command has been executed within the specified interval.	FINEST	Integer	hour minute second	sess-man/adhoc-command/ sess-man/adhoc-command/ sess-man/adhoc-command/
adhoc-command/user-domain-perm/Average processing time	The average time in ms, returned as an integer, it takes for user-domain-perm to execute.	FINEST	Integer		sess-man/adhoc-command/
adhoc-command/user-roster	The number of times	FINEST	Integer	hour minute	sess-man/adhoc-command/ sess-man/adhoc-command/

Statistics Name	Description	Statistics Level	Format	Available {field}	List of Possible Statistics
ter-management last {interval}	user-roster-management command has been executed within the specified interval.			second	sess-man/adhoc-command/v
adhoc-command/user-roster-management/Average processing time	The average time in ms, returned as an integer, it takes for user-roster-management to execute.	FINEST	Integer		sess-man/adhoc-command/v
adhoc-command/user-roster-management-ext last {interval}	The number of times user-roster-management-ext command has been executed within the specified interval.	FINEST	Integer	hour minute second	sess-man/adhoc-command/v sess-man/adhoc-command/v sess-man/adhoc-command/v
adhoc-command/user-roster-management-ext/Average processing time	The average time in ms, returned as an integer, it takes for user-roster-management-ext to execute.	FINEST	Integer		sess-man/adhoc-command/v
Authentication timeouts	The number of connections that have timed out during the authentication process. Default timeout is 2 minutes.	FINEST	Integer		sess-man/Authentication
Closed user connections	User connections that have been terminated by the user (as	FINEST	Integer		sess-man/Closed user con

Statistics Name	Description	Statistics Level	Format	Available {field}	List of Possible Statistics
	opposed to the server).				
default-handler/Invalid registrations	Number of invalid registrations attempted with the server.	FINEST	Integer		sess-man/default-handler
default-handler/Registered users	Number of registered users for this server.	FINEST	Integer		sess-man/default-handler
Maximum user connections	Maximum number of connections that have been made during server instance, this number includes users connecting multiple times.	INFO	Integer		sess-man/Maximum user co
Maximum user sessions {today/yesterday}	The number of most simultaneous sessions within the specified interval. Today = previous 24 hours, Yesterday = 24 hours after previous 24 hours (does not go by calendar date).	INFO/FINEST	Integer		sess-man/Maximum user se sess-man/Maximum user se
Registered accounts	Sum total of registered accounts for the server.	FINEST	Integer		sess-man/Registered acco
Open user connections	The current number of open user connections. This may be interpreted as number of connections from users, however a user can have more than one connection (connection from	INFO	Integer		sess-man/Open user conn

Statistics Name	Description	Statistics Level	Format	Available {field}	List of Possible Statistics
	mobile and PC for example).				
Open user sessions	The current number of open user sessions.	INFO	Integer		sess-man/Open user sess
Total user connections	The cumulative number of connections that have been made to the server during the current instance.	FINER	Integer		sess-man/Total user conn
Total user sessions	The cumulative number of sessions that this server has negotiated during the current instance.	FINER	Integer		sess-man/Total user sess
presence/Users status changes	The number of presence changes for all users that have been conducted during the server instance.	INFO	Integer		sess-man/presence/Users sess-man/presence-state,
sess-man/Processor	Processor statistics will result in a field of labels and values exclusive to that processor.	FINEST	FIELD		sess-man/Processor: mess sess-man/Processor: http sess-man/Processor: jabl sess-man/Processor: vcar sess-man/Processor: amp sess-man/Processor: pres sess-man/Processor: dis sess-man/Processor: msg sess-man/Processor: urn sess-man/Processor: urn sess-man/Processor: jabl sess-man/Processor: urn sess-man/Processor: prp sess-man/Processor: pres sess-man/Processor: mess sess-man/Processor: defa sess-man/Processor: jabl sess-man/Processor: stan sess-man/Processor: pres sess-man/Processor: jabl sess-man/Processor: urn sess-man/Processor: sess sess-man/Processor: jabl

Statistics Name	Description	Statistics Level	Format	Available {field}	List of Possible Statistics
					sess-man/Processor: urn sess-man/Processor: http sess-man/Processor: vcar sess-man/Processor: sess sess-man/Processor: urn sess-man/Processor: jabl sess-man/Processor: Aver sess-man/Processor: Aver
The field shows as follows: , Queue: 0, AvTime: 0, Runs: 0, Lost: 0 Where: Queue: Number of packets in process queue AvTime: Average time in ms processor takes to conduct it's operation. Runs: Number of times Processor has been run. Lost: Number of packets lost during processing.					

## vhost-man

Statistics Name	Description	Statistics Level	Format	List of Possible Statistics
Checks is anonymous domain	Number of anonymous domain checks that have been run within vhost-man.	FINEST	Integer	vhost-man/Checks is anonym
Checks: is local domain	Number of local domain checks that have been run within vhost-man.	FINER	Integer	vhost-man/Checks: is local
Get components for local domain	Number of components loaded within local domain.	FINER	Integer	vhost-man/Get components f
Get components for non-local domain	Number of components loaded outside local domain.	FINEST	Integer	vhost-man/Get components f
Number of Vhosts	Number of configured and running Virtual Hosts.	FINE	Integer	vhost-man/Number of VHosts

## ws2s

No exclusive ws2s specific statistics.

---

# Chapter 13. Appendix II - Properties Guide

Tigase Team <team@tigase.com [mailto:team@tigase.com]> v8.0.0-RC1, 2019-01-30

## General

### admins

**Description:** Specifies a list of administrator accounts.

**Default value:** the administration account created when the server is setup. Typically it would be something like `admins = [ 'admin@server.com [mailto:admin@server.com]' ]`.

**Example:** `admins = [ 'admin@domain.com [mailto:admin@domain.com]', 'user2@domain.com [mailto:user2@domain.com]' ]`

**Possible values:** Comma separated values of Bare JIDs.

**Available since:** 2.0.0

## Certificate Container

The certificate container houses all configuration related to SSL certificate configuration. This container replaces a number of former — properties.

### ssl-certs-location

This option allows you to specify the location where SSL certificates are stored. The meaning of this property depends on the SSL container class implementation. By default it just points to the directory where the server SSL certificates are stored in files in PEM format.

Default location is `/certs` however it can be changed using the following setting:

```
}
'certificate-container' {
  -'ssl-certs-location' = -'/etc/vhost-certs'
}
```

This replaces the former `--ssl-certs-location` property.

### ssl-def-cert-domain

This property allows you to specify a default alias/domain name for certificate. It is mostly used to load certificates for unknown domain names during the SSL negotiation. Unlike in TLS protocol where the domain name is known at the handshaking time, for SSL domain name is not known, therefore, the server does not know which certificate to use. Specifying a domain name in this property allows you to use a certificate for a specific domain in such case. This property value is also sometimes used if there is no certificate for one of virtual domains and the container does not automatically generate a self-signed certificate, then it can use a default one.

This may be configured as follows:

```
}
'certificate-container' {
  -'ssl-def-cert-domain' = -'some.domain.com'
}
```

This replaces the former `--ssl-def-cert-domain` property.

## Component

**Description:** Container specifying component configuration. All components if they require configuration must be called in the `conf.tdsl` file in the following manner:

```
componentName (class: value) {
  <configuration>
}
```

DSL allows for custom naming of the component, and specifying of the class in the same line. This method replaces the old `comp-class` and `comp-name` style of configuration.

For example, what used to be

```
--comp-name-1 = socks5
--comp-class-1 = tigase.socks5.Socks5Component
--comp-name-2 = stun
--comp-class-2 = tigase.stun.StunComponent
```

is now

```
socks5 (class: tigase.socks5.Socks5Component) {}
stun (class: tigase.stun.StunComponent) {}
```

In fact, if you are using the default class & name for a component, you don't need to specify it either, so MUC in this is now called by

```
socks5 () {}
```

**Default value:** By default, component configuration runs of default, and does not need to be specified.

There are many many configuration options under each component, which are specified in component documentation.

## Ports

The `ports` property is a subclass of connections, which is used to set a ports list for a connection manager. 'list of ports' is a comma separated list of ports numbers. For example for the server to server connection manager named `s2s` the property would like like the example below:

```
s2s {
  connections {
    ports = [ 5290, 5291 -]
  }
}
```

Each port may be individually configured underneath ports

```
s2s {
  connections {
    ports = [ 5290, 5291 -]
    5291 {
      type = -'accept'
    }
  }
}
```

this replaces the `--cmpname-ports` property.

**Available since:** 8.0.0

## config-type

**Description:** This property sets the server type and determines what components are started up without needing to declare and configure all components. Possible values are listed below:

- `setup` - This setting will setup a basic server that is prepared for initial setup after unpacking. This is set by default, and starts up http component as well as basic server components. This should be changed after the server is configured.
- `default` - creates default configuration file. That is configuration which is most likely needed for a typical installation. Components included in configuration are: session manager, client-to-server connection manager and server-to-server connection manager.
- `session-manager` - creates 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.
- `connection-managers` - creates 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.
- `component` - generating a configuration with only one component - component managing external components connection, either XEP-0114 or XEP-0225. This is used to deploy a Tigase instance as external component connecting to the main server. You have to add more components handled by this instance, usually these are MUC, PubSub or any other custom components. You have to refer to description for `--comp-name` and `--comp-class` properties to learn how to add components to the Tigase instance. You also have to configure the external component connection, domain name, password, port, etc... Please look for a description for `--external` and `--bind-ext-hostnames` properties.
- ```` - If none is setup, the server will be unable to run as no key is set for null.

**Default value:** 'config-type' = 'setup'

**Possible values:** setup|default|connection-managers|session-manager|connection-managers|component

**Available since:** 2.0.0

## debug-packages

**Default value:** No default as Tigase does not expect custom classes out of the box.



**Example:** `'debug-packages' = [ 'com.company.CustomPlugin' , 'com.company.custom' ]`

**Possible values:** comma separated list of Java packages or classes.

**Description:** This property is used to turn debugging on for any package not located within the default Tigase packages. Be sure class case is correct.

**Available since:** 5.0.0

## debug

**Description:** The debug property is used to turn on the debug log for the specified Tigase package. For example if you want to turn debug logs on for the `tigase.server` package, then you have to use the `server` parameter. If you have any problems with your server the best way to get help from the Tigase team is to generate configuration with this enabled at a minimum and run the server. Then from the `logs/tigase-console.log` log file I can provide the best information for us to provide assistance. More details about server logging and adjusting logging level is described in the Debugging Tigase article in the admin guide. If you wish to debug packages not compiled with Tigase, use the `debug-packages` setting.

**Default value:** 'none'

**Example:** `debug = [ 'server', 'xmpp.impl' ]`

**Possible values:** Comma separated list of Tigase's package names.

**Available since:** 2.0.0

## monitoring

**Description:** This property activates monitoring interfaces through selected protocols on selected TCP/IP port numbers. For more details please refer to the monitoring guide in the user guide for details. Each monitoring protocol should be called in it's own child bean under `monitoring ( )`. If a protocol is not specified, monitoring under that will not be available.

**Default value:** By default monitoring is disabled.

**Example:**

```
monitoring ( ) {
    http ( ) {
        port = -'9080'
    -}
    jmx ( ) {
        port = -'9050'
    -}
    snmp ( ) {
        port = -'9060'
    -}
}
```

### Warning

DO NOT CONFUSE monitoring with monitor component.

**Possible values:** 'list of monitoring protocols with port numbers.'

**Available since:** 8.0.0

## plugins

**Description:** The former `--sm-plugins` property has been replaced by a new style of formatting with DSL. The former long unbroken string of pluses and minuses have been replaced by a compartmentalized style. Plugins controlled under session manager will now be children of the 'sess-man' bean. For example, to turn on the personal eventing protocol, the following may be used:

```
'sess-man' () {  
    pep ()  
}
```

Should any plugin require configuration, those configurations will be under it's own brackets. For example, this section not only turns on jabber:iq:auth but also sets the threads to 16.

```
'sess-man' () {  
    -'jabber:iq:auth' () {  
        threadsNo = 16  
    -}  
}
```

As you may have noticed, beans or configuration options that require escape characters such as `:` or `-` will fall into single quotes to contain any special characters. If no special characters are in the bean name, then no single quotes are not required. If you need to disable certain plugins, you can do so after declaring the bean.

```
'sess-man' () {  
    pep (active: false) {}  
}
```

Typically if a bean is called, it is automatically active. Session manager plugins will typically look like a list of plugins without configurations. The example section will show what one will look like.

**Default value:** 'none'

### Example:

```
'sess-man' () {  
    -'version' () {}  
    amp () {}  
    -'basic-filter' () {}  
    -'domain-filter' () {}  
    -'http:' {  
        {  
            -'jabber.org' {  
                protocol {  
                    commands () {}  
                    stats () {}  
                -}  
            -}  
        -}  
    -}
```

```
- 'jabber:iq:auth' () {
    threadsNo = 16
-}
- 'jabber:iq:privacy' () {}
- 'jabber:iq:private' () {}
- 'jabber:iq:register' () {}
- 'jabber:iq:roster' () {}
- 'message-archive-xep-0136' () {}
msgoffline (active: false) {}
pep () {}
- 'presence-state' () {}
- 'presence-subscription' () {}
starttls () {}
- 'urn:ietf:params:xml:ns:xmpp-bind' () {}
- 'urn:ietf:params:xml:ns:xmpp-sasl' () {}
- 'urn:ietf:params:xml:ns:xmpp-session' () {}
- 'urn:xmpp:ping' () {}
- 'vcard-temp' () {}
zlib () {}
}
```

**Possible values:** DSL format plugins list and configurations.

**Available since:** 8.0.0

## priority-queue-implementation

**Default value:** `tigase.util.PriorityQueueRelaxed`

**Example:** `'priority-queue-implementation'` = `'tigase.util.PriorityQueueStrict'`

**Possible values:** class name extending `tigase.util.PriorityQueueAbstract`.

**Description:** The `priority-queue-implementation` property sets Tigase's internal queue implementation. You can choose between already available and ready to use or you can create own queue implementation and let Tigase load it instead of the default one. Currently following queue implementations are available:

1. **`tigase.util.workqueue.PriorityQueueRelaxed`** - specialized priority queue designed to efficiently handle very high load and prevent packets loss for higher priority queues. This means that sometimes, under the system overload packets may arrive out of order in cases when they could have been dropped. Packets loss (drops) can typically happen for the lowest priority packets (presences) under a very high load.
2. **`tigase.util.workqueue.PriorityQueueStrict`** - specialized priority queue designed to efficiently handle very high load but prefers packet loss over packet reordering. It is suitable for systems with a very high load where the packets order is the critical to proper system functioning. This means that the packets of the same priority with the same source and destination address are never reordered. Packets loss (drops) can typically happen for all packets with the same probability, depending which priority queue is overloaded.
3. **`tigase.util.workqueue.NonpriorityQueue`** - specialized non-priority queue. All packets are stored in a single physical collection, hence they are never reordered. Packets are not prioritized, hence system critical packets may have to wait for low priority packets to be processed. This may impact the server functioning and performance in many cases. Therefore this queue type should be chosen very carefully.

Packets of the same type are never reordered. Packets loss (drops) can typically happen for all packets which do not fit into the single queue.

## Note

*Since the packets are processed by plugins in the SessionManager component and each plugin has own thread-pool with own queues packet reordering may happen regardless what queue type you set. The reordering may only happen, however between different packet types. That is 'message' may take over 'iq' packet or 'iq' packet may take over 'presence' packet and so on... This is unpredictable.*

**Available since:** 5.1.0

## roster-implementation

**Default value:** `RosterFlat.class.getCanonicalName()`

**Example:** `'roster-implementation' = 'my.pack.CustomRosterImpl'`

**Possible values:** Class extending `tigase.xmpp.impl.roster.RosterAbstract`.

**Description:** The `roster-implementation` property allows you to specify a different `RosterAbstract` implementation. This might be useful for a customized roster storage, extended roster content, or in some cases for some custom logic for certain roster elements.

**Available since:** 5.2.0

## s2s-secret

**Default value:** `none`

**Example:**

```
'vhost-man' {
  defaults {
    - 's2s-secret' = - 'some-s2s-secret'
  }
}
```

**Possible values:** 'ascii string.'

**Description:** This property is a global setting for s2s secrets to generate dialback keys on the Tigase installation. By default it is null, which means the secret is automatically generated for each s2s connection and handshake.

This is a global property which is overridden by settings for each vhost.

As in the example provided, 'defaults' settings for all virtual hosts for which the configuration is not defined. This settings is useful mostly for installations with many virtual hosts listed in the `init.property` file for which there is no individual settings specified. It allows to configure a default values for all of them, instead of having to provide individual configuration for each vhost.

It is also applied as a default value for all new vhosts added at run-time.

```
'virtual-hosts' = [ 'xmpp.testnet.net', 'xmpp.glacier.com:domain-filter-policy=OWN:s2s-secret=some-secret' ]
```

Available since: 5.2.0

## scripts-dir

**Default value:** `scripts/admin`

**Example:** `'scripts-dir' = '/opt/admin-scripts'`

**Possible values:** path to a directory on the file system.

**Description:** This property sets the directory where all administrator scripts for ad-hoc commands are stored.

Available since: 4.3.0

## ssl-container-class

**Default value:** `tigase.io.SSLContextContainer`

**Example:** `rootSslContextContainer (class: class.implementing.SSLContextContainerIFC) {}`

**Possible values:** a class implementing `tigase.io.SSLContextContainerIfc`.

**Description:** The `rootSslContextContainer` property allows you to specify a class implementing storage for SSL/TLS certificates. The class presented in the example to this description allows for loading certificates from PEM files which is a common storage used on many systems.

Available since: 5.0.0

## stats

The stats block contains settings for statistics collection. To begin the stats block, use the following:

```
stats {}
```

**Default value:** 'By default, stats is not listed in the `config.tdsl` file'

### Description

Tigase XMPP Server can store server statistics internally for a given period of time. This allows you to connect to a running system and collect all the server metrics along with historic data which are stored on the server. This is very useful when something happens on your production system you can connect and see when exactly this happened and what other metrics looked around this time. **Please be aware that Tigase XMPP Server produces about 1,000 different metrics of the system. Therefore caching large number of statistics sets requires lots of memory.**

## stats-history-size

Stats-history defines the size of the history buffer. That is how many complete sets of historic metrics to store in memory.

```
stats {  
  -'stats-history-size' = -'2160'
```

```
}
```

## stats-history-interval

Sets the interval for which statistics will be gathered from the server.

```
stats {  
  -'stats-history-interval' = -'10'  
}
```

## stats-logger

Allow enabling and configuring components responsible for storing statistic information. Note that this controls the logging system for retrieving using JMX, clients, or ad-hoc commands.

```
stats {  
  -'stats-logger' (class: value) {  
    <other settings>  
  -}  
}
```

Currently following classes are available:

- `tigase.stats.CounterDataArchivizer` - every execution put current basic server metrics (CPU usage, memory usage, number of user connections, uptime) into database (overwrites previous entry)
- `tigase.stats.CounterDataLogger` - every execution insert new row with new set of number of server statistics (CPU usage, memory usage, number of user connections per connector, number of processed packets of different types, uptime, etc) into the database
- `tigase.stats.CounterDataFileLogger` - every execution store all server statistics into separate file.

## frequency

stats-logger may also be controlled using frequency, which is the time interval between executions of the archiver `.execute()` method in seconds.

```
stats {  
  -'stats-logger' (class: tigase.stats.CounterDataLogger) {  
    repository() {  
      'default'() {  
        'data-source' = -'default';  
      }  
    }  
    frequency = -'60'  
  -}  
}
```

## stats-file-logger

This allows configuring of statistics gathering to an external file. This only has one class, and may be controlled independently from the internal statistics.

```
stats {
```

```
    -'stats-file-logger' (class: tigase.stats.CounterDataFileLogger) {  
        <other settings>  
    -}  
}
```

## frequency

stats-file-logger may also be controlled using frequency, which is the time interval between executions of the archiver `.execute()` method in seconds.

```
stats {  
    -'stats-file-logger' (class: tigase.stats.CounterDataFileLogger) {  
        frequency = -'60'  
    -}  
}
```

## file configuration

You can customize the file output for stats-file-logger using the following setting options, these are all optional.

```
stats {  
    -'stats-history-size' = -'2160'  
    -'stats-update-interval' = -'10'  
    -'stats-file-logger' (class: tigase.stats.CounterDataFileLogger) {  
        frequency = -'60'  
        -'stats-datetime' = -'true'  
        -'stats-datetime-format' = -'HH:mm:ss'  
        -'stats-directory' = -'logs/server_statistics'  
        -'stats-filename' = -'stat'  
        -'stats-level' = -'FINEST'  
        -'stats-unixtime' = -'false'  
    }
```

- **'stats-datetime'** - Whether to include date & time timestamp.
- **'stats-datetime-format'** - Specifies the formatting of datetime timestamp.
- **'stats-directory'** - The directory to which the statistics file should be saved.
- **'stats-filename'** - The filename prefix to name the output statistics file.
- **'stats-level'** - Sets the level of statistics to be gathered.
- **'stats-unixtime'** - Control the format of the timestamp to use java DateFormat pattern.

which configures accordingly: directory to which files should be saved, filename prefix, whether to include or not unix timestamp in filename, whether to include or not datetime timestamp, control format of timestamp (using java DateFormat pattern) and also set level of the statistics we want to save (using java Logger.Level)

## Database logger

This allows configuring of statistics gathering to a database. Without additional configuration default data source will be used but it's possible to store statistics in any database - simply define new data source and configure logger with it's name.

## Note

After enabling the component it's database schema should be loaded by executing `./scripts/tigase.sh upgrade-schema etc/tigase.conf` from the main Tigase directory

```
stats {
  -'stats-logger' (class: tigase.stats.CounterDataLogger) {
    repository() {
      'default'() {
        'data-source' = -'customDataSourceName';
      }
    }
    frequency = -'60'
  }
}
```

## Example configuration block

```
stats {
  -'stats-history-size' = -'2160'
  -'stats-update-interval' = -'10'
  -'stats-file-logger' (class: tigase.stats.CounterDataFileLogger) {
    frequency = -'120'
    -'stats-datetime' = -'false'
    -'stats-datetime-format' = -'HH:mm:ss'
    -'stats-directory' = -'logs/statistics'
    -'stats-filename' = -'output'
    -'stats-level' = -'WARNING'
    -'stats-unixtime' = -'true'
  }
  -'stats-logger' (class: tigase.stats.CounterDataLogger) {
    repository() {
      'default'() {
        'data-source' = -'default';
      }
    }
    frequency = -'60'
  }
}
```

**Available since:** 8.0.0

## stream-error-counter

**Description:** Add stream-error-counter to comma separated processors of components for which you wish to count the number of stream errors made. Without enabling this, statistics will return 0. This setting turns on stream-error-counter for both c2s and ws2s:

```
c2s {
  -'stream-error-counter' () {
    active = true
  }
}
ws2s {
```



```
- 'stream-error-counter' () {  
    active = true  
-}  
}
```

You may if you wish turn off stream error counters by setting `active = false`.

**Default value:** Stream error counters are not turned on by default, thus no default value is set.

**Example:**

```
<component> {  
    - 'stream-error-counter' () {  
        active = true  
    -}  
}
```

**Available since:** 7.1.0

## stringprep-processor

**Description:** The `'stringprep-processor'` property sets the stringprep processor for all JIDs handled by Tigase. The default `'simple'` implementation uses regular expressions to parse and check the user JID. Although it does not fulfill the RFC-3920 requirements, it also puts much less stress on the server CPU, hence impact on the performance is very low.

Other possible values are:

`'libidn'` - provides full stringprep processing exactly as described in the RFC-3920. It requires lots of CPU power and significantly impacts performance.

`'empty'` - doesn't do anything to JIDs. JIDs are accepted in the form they are received. No impact on the performance and doesn't use any CPU. This is suitable for use in automated systems where JIDs are generated by some algorithm, hence there is no way incorrect JIDs may enter the system.

**Default value:** `simple`

**Example:** `'stringprep-processor' = 'libidn'`

**Possible values:** `simple|libidn|empty`

**Available since:** 8.0.0

## test

**Default value:** By default test mode is disabled.

**Description:** This property sets the server for test mode, which means that all logging is turned off, offline message storage is off, and possibly some other changes to the system configuration are being made.

The idea behind this mode is to test Tigase XMPP Server with minimal performance impact from environment such as hard drive, database and others...

Test function has been replaced by the following setting:

```
logging {  
    rootLevel = - 'WARNING'  
}
```

**Available since:** 8.0.0

## tls-jdk-nss-bug-workaround-active

**Default value:** false

**Example:** 'tls-jdk-nss-bug-workaround-active' = true

**Possible values:** true|false

**Description:** This is a workaround for TLS/SSL bug in new JDK7 using the native library for keys generation and connection encryption used with new version of nss library.

This caused a number of problems with Tigase installed on systems with JDK7 and the new library installed, such as hanging connections, or broken SSL/TLS. There is some information on our wiki page [[https://projects.tigase.org/projects/tigase-server/wiki/Tigase\\_with\\_OpenJDK7\\_with\\_OpenSSL\\_101](https://projects.tigase.org/projects/tigase-server/wiki/Tigase_with_OpenJDK7_with_OpenSSL_101)]. Our earlier suggestion was to avoid using either JDK7 or the problematic native library. Now we have a proper fix/workaround which allows you to run Tigase with JDK7.

- <http://stackoverflow.com/q/10687200/427545>
- [http://bugs.sun.com/bugdatabase/view\\_bug.do;jsessionid=b509d9cb5d8164d90e6731f5fc44?bug\\_id=6928796](http://bugs.sun.com/bugdatabase/view_bug.do;jsessionid=b509d9cb5d8164d90e6731f5fc44?bug_id=6928796)

Note, while this setting is still supported, the issues mentioned above are fixed in v8 JDK.

**Available since:** 8.0.0

## trusted

**Default value:** none

**Example:** trusted = [ 'user@domain.com [mailto:user@domain.com]' ,  
'user-2@domain2.com [mailto:user-2@domain2.com]' ]

**Possible values:** comma separated list of user bare JIDs.

**Description:** The trusted property allows users to specify a list of accounts which are considered as trusted, thus whom can perform some specific actions on the server. They can execute some commands, send a broadcast message, set MOTD and so on. The configuration is similar to admins setting.

**Available since:** 8.0.0

# Repository

## authRepository

**Description:** Container specifying authentication repository. This container replaces the old auth-db property types, and may contain some other configuration values.

**Default value:**

```
authRepository {
```

```
<configuration>
}
```

This is the basic setup for `authRepository`, where `<configuration>` settings are global for all authentication databases. However, you may configure multiple databases individually.

**Example:**

```
authRepository {
  -'auth-repo-pool-size' = 50
  domain1.com () {
    cls = -'tigase.db.jdbc.JDBCRepository'
    -'data-source' = -'domain1'
  }
  domain2.com () {
    cls = -'tigase.db.jdbc.JDBCRepository'
    -'data-source' = -'domain2'
    -'auth-repo-pool-size' = 30
  }
}
```

## Configuration Values:

Container has the following options

### pool-size

This property sets the database connections pool size for the associated `UserRepository`.

#### Note

in some cases instead of default for this property setting for `data-repo-pool-size` is used if `pool-size` is not defined in `userRepository`. This depends on the repository implementation and the way it is initialized.

```
authRepository {
  default ()
    -'pool-size' = 10
}
```

This is a global property that may be overridden by individual repository settings:

```
userRepository {
  default () {
    -'pool-size' = 10
  }
  special-repo () {
    -'pool-size' = 30
  }
}
```

### cls

Defines the class used for repository connection. You can use this to specify specific drivers for different DB types.

Unless specified, the pool class will use the one included with Tigase. You may configure individual repositories in the same way. This replaces the former `--auth-repo-pool` property.

## Note

File conversion will not remove and convert this property, it **MUST BE DONE MANUALLY**.

Available since: 8.0.0

# authRepository

**Description:** Container specifying repository URIs. This container replaces the old `auth-db-uri` and `user-db-uri` property types.

## Default value:

```
dataSource {
    default () {
        uri = -'jdbc:mysql://localhost/tigasedb?user=tigase&password=tigase12'
    }
}
```

Once your configuration is setup, you will see the uri of your user database here. If other databases need to be defined, they will be listed in the same `dataSource` bean.

## Example:

```
dataSource {
    default () {
        uri = -'jdbc:mysql://localhost/tigasedb?user=tigase&password=tigase12'
    }
    -'default-auth' () {
        uri = -'jdbc:mysql://localhost/tigasedbath?user=tigase&password=tigase12'
    }
    -}
}
```

**Possible values:** Broken down list of customized names for DB URIs. Each name must have a defined uri property. DB name can be customized by the bean name.

## Note

URI name may be used as shorthand to define DB location URI in other containers, so be sure to name them uniquely.

## Note

`default ()` URI setting replaces the `user-db-uri` as well as the `auth-repo-uri` property.

# MSSQL

MSSql support works out of the box, however Tigase provides an open source driver for the database. We recommend using Microsoft's own driver for better functionality.

```
dataSource () {
    default () {
```

```
        uri = -'jdbc:jtds:sqlserver://localhost;databaseName=tigasedb;user=tigase_
    -}
}
```

Where the uri is divided as follows: jdbc:<driver>:sqlserver://<server address>;databaseName=<database name>;user=<username for db>;password=<password for db>;schema=dbo;lastUpdateCount=false;cacheMetaData=false We do not recommend modification of schema and onward unless you are explicit in your changes.

## MongoDb

For using mongoDB as the repository, the setting will look slightly different:

```
dataSource () {
    default () {
        uri = -'mongodb://username:password@localhost/dbname'
    -}
}
```

## pool-size

DataSource is an abstraction layer between any higher level data access repositories such as user-Repository or authRepository and SQL database or JDBC driver to be more specific. Many implementations use DataSource for DB connections and in fact on many installations they also share the same DataRepository instance if they connect to the same DB. In this case it is desired to use a specific connection pool on this level to avoid excessive number of connections to the database.

To do so, specify the number of database connection as an integer:

```
dataSource {
    default () {
        uri = -'jdbc:mysql://localhost/tigasedb?user=tigase&password=tigase12'
        -'pool-size' = -'50'
    -}
}
```

By default, the number of connections is 10.

**Available since:** 8.0.0

# Cluster

## cl-comp

**Description:** Container specifying cluster component configuration.

**Default value:** By default, the cl-comp container is not listed in the config.tdsl file.

**Example:**

```
'cl-comp' {
    <configuration>
}
```

## connect-all

The `cluster-connect-all` property is used to open active connections to all nodes listed in the `cluster-nodes` configuration property. This property should be used only on the node which is added to the live cluster at later time. Normally this new cluster node is not listed in the configuration of the existing cluster nodes. This is why they can not open connections the new node. The new node opens connection to all existing nodes instead. False is the default value and you can skip this option if you want to have it switched off which it is by default.

### Example

```
'cl-comp' {  
    - 'connect-all' = true  
}
```

This replaces the `--cluster-connect-all` property.

**Available since:** 8.0.0

## cluster-mode

**Description:** The property is used to switch cluster mode on. The default value is `false` so you can normally skip the parameter if you don't want the server to run in cluster mode. You can run the server in the cluster mode even if there is only one node running. The performance impact is insignificant and you will have the opportunity to connect mode cluster nodes at any time without restarting the server.

**Default value:** `false` Tigase by default does not run in clustered mode.

**Example:** `'cluster-mode' = 'true'`

**Possible values:** `true|false`

**Available since:** 8.0.0

## cluster-nodes

**Default value:** none

**Example:** `'cluster-nodes' = [ 'node1.domain:pass:port' , 'node2.domain:pass:port' , 'node3.domain:pass:port' ]`

**Possible values:** a comma separated list of hostnames.

**Description:** The property is used to specify a list of cluster nodes running on your installation. The node is the full DNS name of the machine running the node. Please note the proper DNS configuration is critical for the cluster to work correctly. Make sure the `'hostname'` command returns a full DNS name on each cluster node. Nodes don't have to be in the same network although good network connectivity is also a critical element for an effective cluster performance.

All cluster nodes must be connected with each other to maintain user session synchronization and exchange packets between users connected to different nodes. Therefore each cluster node opens a `'cluster port'` on which it is listening for connections from different cluster nodes. As there is only one connection between each two nodes Tigase server has to decide which nodes to connect to and which has to accept the connection. If you put the same list of cluster nodes in the configuration for all nodes this is not a problem. Tigase server has a way to find and void any conflicts that are found. If you however want to add a new node later on, without restarting and changing configuration on old nodes, there is no way the

old nodes will try to establish a connection to the new node they don't know them. To solve this particular case the next parameter is used.

### Note

Cluster nodes are not required to be configured, as they can automatically find/add/remove cluster nodes. This is for installations where nodes will be limited and static!

**Available since:** 8.0.0

## User connectivity

### bosh-close-connection

**Default value:** `false`

**Example:** `'bosh-close-connection' = true`

**Possible values:** `true|false`

**Description:** This property globally disables Bosh keep-alive support for Tigase server. It causes the Bosh connection manager to force close the HTTP connection each time data is sent to the Bosh client. To continue communication the client must open a new HTTP connection.

This setting is rarely required but on installations where the client cannot control/disable keep-alive Bosh connections and keep-alive does not work correctly for some reason.

**Available since:** 8.0.0

### bosh-extra-headers-file

**Default value:** `'etc/bosh-extra-headers.txt'`

**Example:** `'bosh-extra-headers-file' = '/path/to/file.txt'`

**Possible values:** `'path to a file on the filesystem.'`

**Description:** This property allows you to specify a path to a text file with additional HTTP headers which will be sent to a Bosh client with each request. This gives some extra flexibility for Bosh clients running on some systems with special requirements for the HTTP headers and some additional settings.

By default a file distributed with the installation contains following content:

```
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, OPTIONS
Access-Control-Allow-Headers: Content-Type
Access-Control-Max-Age: 86400
```

This can be modified, removed or replaced with a different content on your installation.

**Available since:** 8.0.0

### client-access-policy-file

**Default value:** `etc/client-access-policy.xml`

**Example:** `'client-access-policy-file' = ''/path/to/access-policy-file.xml'`

**Possible values:** path to a file on the filesystem.

**Description:** The `client-access-policy-file` property allows control of the cross domain access policy for Silverlight based web applications. The cross domain policy is controlled via XML file which contains the policy and rules.

By default Tigase is distributed with an example policy file which allows for full access from all sources to the whole installation. This is generally okay for most Bosh server installations. The configuration through the property and XML files allows for a very easy and flexible modification of the policy on any installation.

**Available since:** 5.2.0

## client-port-delay-listening

**Description:** The property allows to enabled or disable delaying of listening for client connections **in cluster mode** until the cluster is correctly connected.

**Default value:** `true`

**Example:**

```
<component> {  
  -'port-delay-listening' = false  
-}
```

**Possible values:** `true, false`

In cluster mode, in order to ensure correct user status broadcast, we are delaying opening client ports (components: `c2s`, `ws2s`, `bosh`) and enable those only after cluster is fully and correctly connected (i.e. either there is only single node or in case of multiple nodes all nodes connected correctly).

It's possible to enable/disable this on per-component basis with the following configuration:

```
bosh {  
  -'port-delay-listening' = true  
}  
c2s {  
  -'port-delay-listening' = true  
}  
ws2s {  
  -'port-delay-listening' = true  
}
```

Maximum delay time depends on the component and it's multiplication of `ConnectionManager` default connection delay times `30s` - in case of client connection manager this delay equals `60s`.

### Note

Only applicable if **Cluster Mode** is active!

**Available since:** 7.1.0



## cross-domain-policy-file

**Default value:** `etc/cross-domain-policy.xml`

**Example:** `'cross-domain-policy-file' = '/path/to/cross-domain-policy.xml'`

**Possible values:** path to a file on the file system.

**Description:** This property allows you to set a path to a file with cross domain access policy for flash based clients. This is a standard XML file which is sent to the flash client upon request.

A default file distributed with Tigase installations allows for full access for all. This is good enough for most use cases but it can be changed by simply editing the file.

This is a global property that can also be overridden by configuring connection managers [ `c2s`, `s2s`, `ws2s`, `bosh`, `ext`, etc] and they may all have their own policies.

```
c2s {
    - 'cross-domain-policy-file' = - '/path/to/cross-domain-policy.xml'
}
```

**Available since:** 5.1.0

## domain-filter-policy

**Default value:** `ALL`

**Example:** `domain-filter-policy' = 'LOCAL`

**Possible values:** `ALL | LOCAL | OWN | BLOCK | LIST=domain1;domain2 | BLACKLIST=domain1;domain2`

**Description:** The `domain-filter-policy` property is a global setting for setting communication filtering for vhosts. This function is kind of an extension of the same property which could be set on a single user level. However, in many cases it is desired to control users communication not on per user-level but on the domain level. Domain filtering (communication filtering) allows you to specify with whom users can communicate for a particular domain. It enables restriction of communications for a selected domain or for the entire installation. A default value `ALL` renders users for the domain (by default for all domains) able to communicate with any user on any other domains. Other possible values are:

1. `ALL` a default value allowing users to communicate with anybody on any other domain, including external servers.
2. `LOCAL` allows users to communicate with all users on the same installation on any domain. It only blocks communication with external servers.
3. `OWN` allows users to communicate with all other users on the same domain. Plus it allows users to communicate with subdomains such as **`muc.domain`**, **`pubsub.domain`**, etc...
4. `BLOCK` value completely blocks communication for the domain or for the user with anybody else. This could be used as a means to temporarily disable account or domain.
5. `LIST` property allows to set a list of domains (users' JIDs) with which users on the domain can communicate (i.e. *whitelist*).
6. `BLACKLIST` - user can communicate with everybody (like `ALL`), except contacts on listed domains.

This is a global property which is overridden by settings for particular vhosts.

```
`'virtual-hosts' = [ -'domain2:domain-filter-policy=OWN' -],
```

A default settings for all virtual hosts for which the configuration is not defined. This settings is useful mostly for installations with many virtual hosts listed in the `init.property` file for which there is no individual settings specified. It allows default value for all of servers, instead of having to provide individual configuration for each vhost.

ALL is also applied as a default value for all new vhosts added at run-time.

**Available since:** 5.2.0

## see-other-host

--cmSeeOtherHost has been replaced with using `seeOtherHost` setting, and can be configured for each connection manager (c2s, s2s, etc..)

**Default value:** `tigase.server.xmppclient.SeeOtherHostHashed`

**Example:**

```
<connectionManager> {  
  seeOtherHost (class: value) { -}  
}
```

**Possible values:** 'none' 'or class implementing SeeOtherHostIfc.'

**Description:** Allows you to specify a load balancing mechanism by specifying `SeeOtherHostIfc` implementation. More details about functionality and implementation details can be found in Tigase Load Balancing documentation.

**Available since:** 8.0.0

## watchdog\_timeout

**Default value:** 1740000

**Example:** `watchdog_timeout=60000`

**Possible values:** any integer.

**Description:** The `watchdog_timeout` property allows for fine-tuning ConnectionManager Watchdog (service responsible for detecting broken connections and closing them). Timeout property relates to the amount of time (in miliseconds) after which lack of response/activity on a given connection will be considered such connection as broken and close it. In addition to global configuration presented above a per component configuration is possible:

```
<ConnectionManager> {  
  watchdog_timeout = 60000L  
}
```

for example (for C2SConnectionManager):

```
c2s {  
  watchdog_timeout = 150000L
```

```
}
```

All related configuration options:

- watchdog\_Ping\_Type
- watchdog\_delay
- watchdog\_timeout

**Available since:** 8.0.0

## watchdog\_delay

**Default value:** 600000

**Example:** `watchdog_delay = '30000'`

**Possible values:** 'any integer.'

**Description:** `watchdog_delay` configuration property allows configuring delay (in milliseconds) between subsequent checks that ConnectionManager Watchdog (service responsible for detecting broken connections and closing them) will use to verify the connection. In addition to global configuration presented above a per component configuration is possible:

```
<ConnectionManager> {  
    watchdog_delay = 60000L  
}
```

for example (for ClusterConnectionManager):

```
'cl-comp' {  
    watchdog_delay = 150000L  
}
```

All related configuration options:

- watchdog\_Ping\_Type
- watchdog\_delay
- watchdog\_timeout

**Available since:** 8.0.0

## watchdog\_ping\_type

**Default value:** whitespace

**Example:** `watchdog_ping_type = 'XMPP'`

**Possible values:** WHITESPACE,XMPP

**Description:** `watchdog_ping_type` configuration property allows configuring of the type of ping that ConnectionManager Watchdog (service responsible for detecting broken connections and closing them) will use to check the connection. In addition to global configuration presented above a per component configuration is possible:

```
<ConnectionManager> {  
  watchdog_ping_type = - 'XMPP'  
}
```

for example (for ClusterConnectionManager):

```
cl-comp {  
  watchdog_ping_type = - 'WHITESPACE'  
}
```

All related configuration options:

- watchdog\_ping\_type
- watchdog\_Delay
- watchdog\_timeout

**Available since:** 8.0.0

## ws-allow-unmasked-frames

**Default value:** false

**Example:** 'ws-allow-unmasked-frames' = true

**Possible values:** true|false

**Description:** RFC 6455 specifies that all clients must mask frames that it sends to the server over WebSocket connections. If unmasked frames are sent, regardless of any encryption, the server must close the connection. Some clients however, may not support masking frames, or you may wish to bypass this security measure for development purposes. This setting, when enabled true, will allow connections over websocket to be unmasked to the server, and may operate without Tigase closing that connection.

**Available since:** 8.0.0

# External

## bind-ext-hostnames

**Default value:** none

**Example:** 'bind-ext-hostnames' = [ 'pubsub.host.domain' ]

**Possible values:** comma separated list of domains.

**Description:** This property enables setting a list of domains to be bound to the external component connection. Let's say we have a Tigase instance with only MUC and PubSub components loaded and we want to connect this instance to the main server via external component protocol. Using --external property we can define a domain (perhaps muc.devel.tigase.org), password, TCP/IP port, remote host address, connection type, etc... This would make one of our components (MUC) visible on the remote server.

To make the second component (PubSub) visible we would need to open another connection with the domain name (pubsub.devel.tigase.org) for the other component. Of course the second connection is redundant as all communication could go through a single connection. This is what this property is used. In

our example with 2 components you can just put the 'pubsub.devel.tigase.org' domain as a value to this property and it will bind the second domain to a single connection on top of the domain which has been authenticated during protocol handshaking.

**Available since:** 5.0.0

## default-virtual-host

**Description:** The `default-virtual-host` property allows setting of the name of default virtual host that is served by the installation. It is loaded during startup of the application and stored in the database.  
**It may only contain single domain name!**

Any additional configuration options or additional virtual hosts domains should be added and configured using ad-hoc commands such as `Add new item`, `Update item configuration` and `Remove an item` available at the JID of the `VHostManager` component of your installation (`vhost-man@your-server-domain`).

**Available since:** 8.0.0

## ext

**Description:** This property defines parameters for external component connections.

The component is loaded the same way as all other Tigase components. In your `config.tdsl` file you need to add the external class:

```
ext (class: tigase.server.ext.ComponentProtocol) {}
```

This will load the component with an empty configuration and is practically useless. You have to tell the component on what port to listen to (or on what port to connect to) and external domains list with passwords.

Those values need to be configured while the Tigase XMPP Server is running using XMPP ad-hoc commands such as `Add new item`, `Update item configuration` and `Remove an item` available at the JID of the external component which you have just enabled (`ext@your-server-domain`).

**Possible values:** external domains parameters list.

**Available since:** 4.3.0

**Removed in:** 8.0.0

# Performance

## cm-ht-traffic-throttling

**Default value:** `xmpp:25k:0:disc,bin:200m:0:disc`

**Example:** `'cm-ht-traffic-throttling' = 'xmpp:25k:0:disc,bin:200m:0:disc'`

**Possible values:** comma separated list of traffic limits settings.

**Description:** This property is used to specify traffic limit of non-user connections, that is s2s, external components and other high traffic server connections. The meaning of the property and values encoded are in the same way as for the `cm-traffic-throttling` property.

**Available since:** 8.0.0

## cm-traffic-throttling

**Default value:** `xmpp:2500:0:disc,bin:20m:0:disc`

**Example:** `'cm-traffic-throttling' = 'xmpp:2500:0:disc,bin:20m:0:disc'`

**Possible values:** comma separated list of traffic limits settings.

**Description:** The `cm-traffic-throttling` property allows you to limit traffic on user connections. These limits are applied to each user connection and if a limit is exceeded then a specified action is applied.

The property value is a comma separated list of traffic limits settings. For example the first part: `xmpp:2500:0:disc` specifies traffic limits for XMPP data to 2,500 packets allowed within last minute either sent to or received from a user and unlimited (0) total traffic on the user connection, in case any limit is exceeded the action is to **disconnect** the user.

- **[xmpp|bin]** traffic type, `xmpp` - XMPP traffic, that is limits refer to a number of XMPP packets transmitted, `bin` - binary traffic, that is limits refer to a number of bytes transmitted.
- **2500** maximum traffic allowed within 1 minute. 0 means unlimited, or no limits.
- **0** maximum traffic allowed for the life span of the connection. 0 means unlimited or no limits.
- **[disc|drop]** action performed on the connection if limits are exceeded. `disc` - means disconnect, `drop` - means drop data.

**Available since:** 5.1.3

## elements-number-limit

**Default value:** 1000

**Possible values:** any integer.

**Description:** `elements-number-limit` configuration property allows configuring a Denial of Service protection mechanism which limits number of elements sent in stanza. It must be configured on a per `ConnectionManager` basis:

```
'<ConnectionManager>' {  
    -'elements-number-limit' = ###  
}
```

for example (for `ClusterConnectionManager`):

```
'cl-comp' {  
    -'elements-number-limit' = 100000
```

**Available since:** 5.2.0

## hardened-mode

**Default value:** `false`

**Example:** `'hardened-mode' = true`

**Possible values:** true|false

**Description:** Enabling hardened mode affects handling of security aspects within Tigase. It turns off workarounds for SSL issues, turns off SSLv2 and forces enabling more secure ciphers suites. It also forces requirement of StartTLS.

Enabling it requires UnlimitedJCEPolicyJDK [<http://www.oracle.com/technetwork/java/javase/downloads/jce-7-download-432124.html>] installed. We prefer to use OracleJDK as our tests revealed that using OpenJDK in hardened mode may cause issues with some clients on some platforms.

**Available since:** 5.2.0

## max-queue-size

**Default value:** default queue size is variable depending on RAM size.

**Example:** 'max-queue-size' = 10000

**Possible values:** integer number.

**Description:** The max-queue-size property sets internal queues maximum size to a specified value. By default Tigase sets the queue size depending on the maximum available memory to the Tigase server process. It set's 1000 for each 100MB memory assigned for JVM. This is enough for most cases. If you have however, an extremely busy service with Pubsub or MUC component generating huge number of packets (presence or messages) this size should be equal or bigger to the maximum expected number of packets generated by the component in a single request. Otherwise Tigase may drop packets that it is unable to process.

**Available since:** 5.1.0

## net-buff-high-throughput

**Default value:** 64k

**Example:** 'net-buff-high-throughput' = '256k'

**Possible values:** network buffer size as integer.

**Description:** The net-buff-high-throughput property sets the network buffer for high traffic connections like s2s or connections between cluster nodes. The default is 64k and is optimal for medium traffic websites. If your cluster installation can not cope with traffic between nodes try to increase this number.

**Available since:** 4.3.0

## net-buff-standard

**Default value:** 2k

**Example:** 'net-buff-standard' = '16k'

**Possible values:** network buffer size as integer.

**Description:** This property sets the network buffer for standard (usually c2s) connections, default value is 2k and is optimal for most installations.

**Available since:** 4.3.0

## nonpriority-queue

**Default value:** false

**Example:** 'nonpriority-queue' = true

**Possible values:** true|false

**Description:** The nonpriority property can be used to switch to non-priority queues usage in Tigase server (value set to 'true'). Using non-priority queues prevents packets reordering. By default Tigase uses priority queues which means that packets with highest priority may take over packets with lower priority (presence updates) which may result in packets arriving out of order.

This may happen however only for packets of different types. That is, messages may take over presence packets. However, one message never takes over another message for the same user. Therefore, out of order packet delivery is not an issue for the most part.

**Available since:** 5.0.0

## VHost / domain

### vhost-anonymous-enabled

**Default value:** true

**Example:** 'vhost-anonymous-enabled' = 'false'

**Possible values:** true|false

**Description:** The vhost-anonymous-enabled property specifies whether anonymous user logins are allowed for the installation for all vhosts.

This is a global property which is overridden by settings for particular vhost.

Default settings for all virtual hosts are used when this property is not defined. This settings is useful mostly for installations with many virtual hosts listed in the `init.property` file for which there is no individual settings specified. It allows the configuration of default values for all of them, instead of having to provide individual configuration for each vhost.

It is also applied as a default value for all new vhosts added at run-time.

This is a global property which is overridden by settings for particular vhost.

```
'virtual-hosts' = [ -'domain4:vhost-anonymous-enabled = true' -]
```

**Available since:** 8.0.0

### vhost-disable-dns-check

**Default value:** false

**Example:** 'vhost-disable-dns-check' = 'true'



**Possible values:** `true|false`

**Description:** This property disables DNS validation when adding or editing vhosts in Tigase server. This also exempts administrative accounts from validation. With this property enabled, you will not benefit from seeing if proper SRV records are set so other people can connect to specific vhosts from outside your network.

This is a global property which is overridden by settings for particular vhost.

```
'virtual-hosts' = [ -'domain4:vhost-disable-dns-check = true' -]
```

**Available since:** 8.0.0

## vhost-max-users

**Default value:** 0

**Example:** `'vhost-max-users' = '1000'`

**Possible values:** integer number.

**Description:** The `vhost-max-users` property specifies how many user accounts can be registered on the installations for all vhosts.

**0 - zero** means unlimited and this is a default. Otherwise greater than zero value specifies accounts number limit.

This is a global property which is overridden by settings for particular vhost.

The default setting is used for all virtual hosts for which the configuration is not defined. This settings is most useful for installations with many virtual hosts listed in the `init.property` file for which there is no individual settings specified. It provides an ability to use default values for all of them, instead of having to provide individual configuration for each vhost.

This is a global property which is overridden by settings for particular vhost.

```
'virtual-hosts' = [ -'domain3:vhost-max-users = 200' -]
```

**Available since:** 8.0.0

## vhost-message-forward-jid

**Default value:** `<null>`

**Example:** `'vhost-message-forward-jid' = 'archive@domain.com [mailto:archive@domain.com]'`

**Possible values:** 'valid JID'

**Description:** This is a global property for message forwarding for the installation. This property is normally specified on the vhost configuration level, however if you want to forward all messages on your installation and you have many virtual domains this property allows to set message forwarding for all of them. A valid JID must be specified as the forwarding destination. Also a message forwarding plugin must be loaded and activated on the installation for the message forwarding to work.

The null value is used as a default when no configuration is set. This setting is mostly useful for installations with many virtual hosts listed in the `init.property` file for which there is no individual settings

specified. It provides the ability to configure a default values for all of them, instead of having to provide individual configuration for each vhost.

It is also applied as a default value for all new vhosts added at run-time.

This is a global property which is overridden by settings for particular vhost.

```
'virtual-hosts' = [ -'domain3:vhost-message-forward-jid = archive@domain.com' -]
```

**Available since:** 8.0.0

## vhost-presence-forward-jid

**Default value:** <null>

**Example:** 'vhost-presence-forward-jid' = 'presence-collector@domain.com  
[mailto:presence-collector@domain.com]'

**Possible values:** valid JID.

**Description:** This is a global property for presence forwarding function for the installation. All user status presences will be forwarded to given XMPP address which can be a component or any other XMPP entity. If the destination entity is a bot connected via c2s connection it probably should be addressed via full JID (with resource part) or the standard XMPP presence processing would refuse to deliver presences from users who are not in the contact list.

This is a global property which is overridden by settings for particular vhost.

The null value is used as a default when no configuration is set. This settings is useful for installations with many virtual hosts listed in the `init.property` file for which there is no individual settings specified. It enables the ability to configure default values for all of them, instead of having to provide individual configuration for each vhost.

It is also applied as a default value for all new vhosts added at run-time.

This may be used on a per-vhost basis

```
'virtual-hosts' = [ -'domain3:vhost-presence-forward-jid = presence-collector@domain.com' -]
```

**Available since:** 8.0.0

## vhost-register-enabled

**Default value:** true

**Example:** 'vhost-register-enabled' = false

**Possible values:** true|false

**Description:** `vhost-register-enabled` is a global property which allows you to switch on/off user registration on the installation. Setting this property to `false` does not disable the registration plugin on the server. You can enable registration for selected domains in the domain configuration settings.

This is a global property which is overridden by settings for particular vhost.

The `true` value is used as a default when no configuration is set. This settings is useful for installations with many virtual hosts listed in the `init.property` file for which there is no individual settings spec-

ified. It allows admins to configure default values for all of them, instead of having to provide individual configuration for each vhost.

It is also applied as a default value for all new vhosts added at run-time.

This may be used on a per-vhost basis

```
'virtual-hosts' = [ -'domain3:vhost-register-enabled = false' -]
```

**Available since:** 8.0.0

## vhost-tls-required

**Default value:** `false`

**Example:** `'vhost-tls-required' = true`

**Possible values:** `true|false`

**Description:** This property is a global settings to switch on/off TLS required mode on the Tigase installation. Setting this property to `false` does not turn TLS off. TLS is still available on the server but as an option and this is the client's decision whether to use encryption or not. If the property is set to `true` the server will not allow for user authentication or sending any other user data before TLS handshaking is completed.

This is a global property which is overridden by settings for particular vhost.

The `false` value is used as a default when no configuration is set. This settings is useful for installations with many virtual hosts listed in the `init.property` file for which there is no individual settings specified. It allows admins to configure default values for all of them, instead of having to provide individual configuration for each vhost.

It is also applied as a default value for all new vhosts added at run-time.

This may be used on a per-vhost basis

```
'virtual-hosts' = [ -'domain3:vhost-tls-required = true' -]
```

**Available since:** 8.0.0

# Tigase Server Extras - mDNS support

## Overview

Tigase mDNS component provides you with ability to publish domain name of your XMPP server (ending with `.local`) in the local network using multicast DNS (also known as DNS-SD, Zeroconf or Bonjour).

## Enabling mDNS

To enable this component you need to add mDNS component to your configuration file:

```
mdns ( ) {  
}
```

This lines will enable mDNS support and will start broadcasting hostname of your host in the local network as `hostname.local` and will broadcast DNS records for XMPP server hosted at this domain.

## Using different domain name

If you are hosting different domain than hostname of your server with `.local` suffix, then you can set it in mDNS component settings by setting `serverHost` property to the name of your domain without suffix `.local`.

**Example of broadcasting mDNS for domain `example.local`.**

```
mdns () {  
    serverHost = -'example'  
}
```

## Forcing single server for domain

It is possible to enforce Tigase mDNS component to check if there is no other host providing services for chosen domain name. By setting property `singleServer` to `true`. If this feature is enabled, then mDNS component checks if chosen domain is already in use (broadcasted in multicast DNS) and if so it stops startup of the server. This feature make it possible to start up Tigase and broadcast XMPP server mDNS information if already existing mDNS information resolves to the IP address of the host on which you are starting Tigase XMPP Server.

**Example enabling single server mode.**

```
mdns () {  
    singleServer = true  
}
```

---

# Chapter 14. HTTP API component

Tigase HTTP API component is a generic container used to provide other HTTP related features as modules. It is configured by default to run under name of `http`. Installations of Tigase XMPP Server run this component enabled by default under the same name even if not configured.

## Available modules

### Admin UI module

This is very simple module for administration of Tigase XMPP Server using HTTP browser. It allows administrators to execute ad-hoc commands from HTTP browser allowing to change some configuration options at runtime. It can be accessed by pointing your browser to `http://server.address:8080/admin/` and logging in with admin credentials.

### Index module

This module is deployed at `/` by default and provides list of installed and available modules for the virtual host when requested.

### REST module

This module provides REST-like API for accessing Tigase XMPP Server. It uses Groovy scripts to process HTTP requests and prepare responses.

### Server status module

#### Warning

This module is still a work in progress!

This module is designed to present current server status and report possible issues.

### Setup module

Module is created to act as a web based installer and configuration utility for Tigase XMPP Server. Allows you to modify basic Tigase XMPP Server settings, ie. related to database access. Changes may be saved to configuration file from this module.

### Web UI module

This module contains full web client based on Tigase JaXMPP [<http://www.tigase.net/content/jaxmpp-library/>] client library allowing user to chat, manage contacts list (roster), browse message archive, etc. For more information on this module, consult the Administration Guide [[http://docs.tigase.org/tigase-server/snapshot/Administration\\_Guide/html/#\\_use\\_of\\_the\\_http\\_api](http://docs.tigase.org/tigase-server/snapshot/Administration_Guide/html/#_use_of_the_http_api)].

### User Status Endpoint module

This module is designed as an endpoint required for REST API User Status to work properly. It is not accessible using HTTP/REST API, so it can (and in most cases should) be active.

# Common module configuration

## Enabling/disabling module

Every module can be activated or disabled by adjusting it's activity in following way:

```
http {
    %module_id% (active: false) {}
}
```

### Note

You need to replace `%module_id%` with the id of module which you want to change activity (in this case, it will disable module).

**Disabling REST module.**

```
http {
    rest (active: false) {}
}
```

## Context path

This property allows you to change the context path that is used by module. In other words, it allows you to change the prefix used by module. By default every module (with exception of the Index module) uses a context path that is the same as module id. For example, the REST module ID results in the context path `/rest`

**Changing context path for REST module to `/api`.**

```
http {
    rest {
        context-path = -'/api'
    -}
}
```

## List of virtual hosts

This provides the ability to limit modules to be available only on listed virtual hosts, and allows to set context path to `/` for more than one module. Property accepts list of strings, which in the case of `config.tdsl` file format is list of comma separated domain names and in DSL it is written as list of strings (see Complex Example).

**Moving the REST module to be available only for requests directed to `api.example.com`.**

```
http {
    rest {
        vhosts = [ -'api.example.com' -]
    -}
}
```

## Complex example

In this example we will disable the Index module and move REST module to `http://api.example.com/` and `http://rest.example.com`.

```
http {
  index (active: false) {}
  rest {
    context-path = - '/'
    vhosts = [ - 'api.example.com', - 'rest.example.com' - ]
  }
}
```

## Module specific configuration

Tigase will try to start a standalone Jetty HTTP server at port 8080 and start up the default modules, including `RestModule` which will add context for REST API in the `/rest` path. `RestModule` will also load all groovy scripts located in `scripts/rest/*` directories and will bind them to proper actions for the `/rest/*` paths.

**NOTE:** Scripts that handle HTTP requests are available in the component repository in `src/scripts/groovy/tigase/rest/` directory.

Tigase's REST Component comes with two modules that can be enabled, disabled, and configured separately. Common settings for modules for component properties are used in the following format: `component_name (module: value) {}` the following settings are available for both listed modules:

- `active` - Boolean values `true/false` to enable or disable the module.
- `context-path` - Path of HTTP context under which the module should be available.
- `vhosts` - Comma separated list of virtual hosts for which the module should be available. If not configured, the module will be available for all vhosts.

## Rest Module

This is the Module that provides support for the REST API. Available properties:

- `rest-scripts-dir` - Provides ability to specify path to scripts processing REST requests if you do not wish to use default (`scripts/rest`).

## API keys

In previous version it was possible to configure `api-keys` for REST module using entries within configuration file. In the recent version we decided to remove this configuration option. Now, by default Tigase XMPP Server requires API key to be passed to all requests and you need to configure them before you will be able to use REST API.

Instead, you should use ad-hocs available on the REST module JID to:

- Add API key (`api-key-add`)
- Update API key (`api-key-update`)
- Remove API key (`api-key-remove`);

### Tip

If you have Admin UI enabled, you may log in using admin credentials to this UI and when you select **CONFIGURATION** section on the left sidebar, it will expand and allow you to execute any of those ad-hoc commands mentioned above.

Requests made to the HTTP service must conclude with one of the API keys defined using ad-hoc commands: `http://localhost:8080/rest/adhoc/sess-man@domain.com?api-key=test1`

## Note

If you want to allow access to REST API without usage of any keys, it is possible. To do so, you need to add an API key with `API key` field value equal `open_access`.

## DNS Web Service module

For web based XMPP clients it is not possible to execute DNS SRV requests to find address of XMPP server hosting for particular domain. To solve this the DNS Web Service module was created.

It handles incoming HTTP GET request and using passed `domain` and `callback` HTTP parameters executes DNS requests as specified in XEP-0156: Discovering Alternative XMPP Connection Methods [<https://xmpp.org/extensions/xep-0156.html>]. Results are returned in JSON format for easy processing by web based XMPP client.

By default it is deployed at `dns-webservice`

## Parameters

<code>domain</code>	Domain name to look for XMPP SRV client records.
<code>callback</code>	Due to security reasons web based client may not be able to access some DNS Web Service due to cross-domain AJAX requests. Passing optional <code>callback</code> parameter sets name of callback for JSONP requests and results proper response in JSONP format.

## Discover way to connect to XMPP server hosting `sure.im` domain.

Sending HTTP GET request to `http://our-xmpp-server:8080/dns-websevice/?domain=sure.im&version=2` you will receive following response:

```
{
  domain: -'sure.im',
  c2s: [
    {
      host: -'tigase.me',
      ip: ['198.100.157.101', '198.100.157.103', '198.100.153.203'],
      port: 5222,
      priority: 5
    }
  ],
  bosh: [
    {url: 'http://blue.sure.im:5280/bosh'},
    {url: 'http://green.sure.im:5280/bosh'},
    {url: 'http://orange.sure.im:5280/bosh'}
  ],
  websocket: [
    {url: 'ws://blue.sure.im:5290/'},
    {url: 'ws://green.sure.im:5290/'},
    {url: 'ws://orange.sure.im:5290/'}
```



```
- ]  
}
```

As you can see in here we have names and IP address of XMPP servers hosting sure . im domain as well as list of URI for establishing connections using BOSH or WebSocket.

This module is activated by default. However, if you are operating in a test environment where you may not have SRV and A records setup to the domain you are using, you may want to disable this in your config.tdsl file with the following line:

```
rest {  
  - 'dns-webservice' (active: false) {}  
}
```

## Enabling password reset mechanism

It is possible to provide users with a mechanism for a password change in case if they forgot their password to the XMPP account. To do that you need to have `tigase-extras.jar` in your classpath (it is part of `-dist-max` distribution package), enable mailer and `account-email-password-resetter`.

### Example configuration.

```
account-email-password-resetter () {}  
mailer (class: tigase.extras.mailer.Mailer) {  
  - 'mailer-from-address' = - 'email-address@to-send-emails-from'  
  - 'mailer-smtp-host' = - 'smtp.email.server.com'  
  - 'mailer-smtp-password' = - 'password-for-email-account'  
  - 'mailer-smtp-port' = - '587' # Email server SMTP port  
  - 'mailer-smtp-username' = - 'username-for-email-account'  
}
```

### Note

You need to replace example configuration parameters with correct ones.

With this configuration in place and after restart of Tigase XMPP Server at url `http://localhost:8080/rest/user/resetPassword` will be available web form which may be used for password reset.

### Note

This mechanism will only work if user provided real email address during account registration and if user still remembers and has access to email address used during registration.

## Admin UI Guide

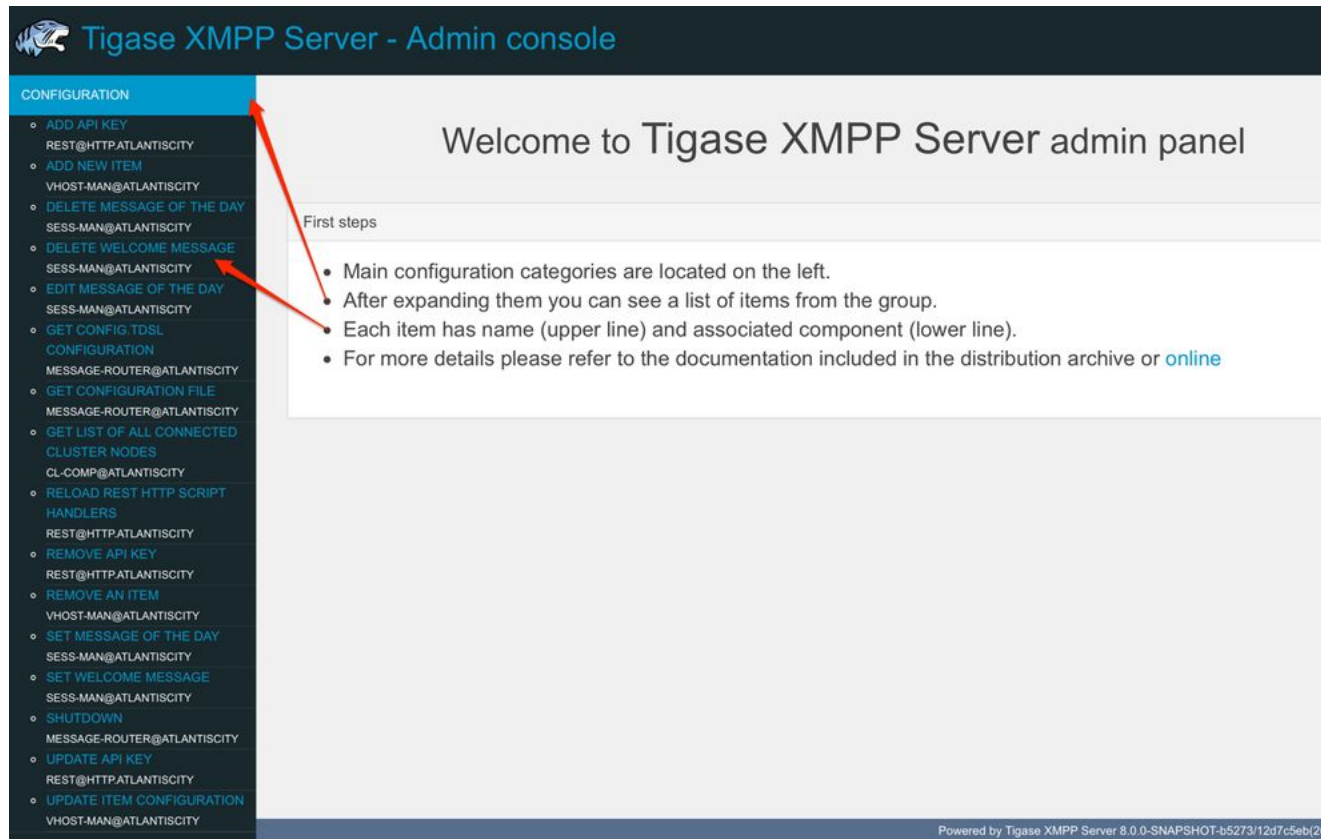
The Admin User Interface is an HTTP-based interface that sends REST commands to the server to update configurations, change settings, and retrieve statistics.

## A Note about REST

REST stands for REpresentational State Transfer which is a stateless communication method that in our case passes commands using HTTP GET, PUT, POST, and DELETE commands to resources within the Tigase server.

## General overview of the UI

After navigating to the Admin WebUI you will see basic information about navigation. The panel itself consists of two main parts: \* left navigation menu, which groups all configuration items into categories; \* central, main configuration page displaying configuration options of the selected items.



Each configuration item has name (upper line) and associated component (lower line) as some features can be executed in the context of different component (for example Update Item Configuration can be executed for VirtualHost Manager or ExternalConnection Manager)

## Configuration

Allows you to configure some of the servers settings, such as message of the day, welcome message or initialize shutdown of the cluster node.

## Example Scripts

This is a list of script examples that can be run and do menial functions for each component. They may not have particular value themselves, but are present to be used as reference when writing custom scripts. Get list of available commands is one script, that is present for every component that is active on the server, and as its title implies, will provide a list of all commands for that component. Lastly, the two scripts from the Scripting section of this guide. Generally, there is not much needed to see in this section.

## Notifications

This section has one simple command: to be able to send a mass message to all logged in users. There are three types of messages that can be sent from this section: - **normal** Messages will show as a pop-up in most clients. - **headline** Certain clients will take headline messages and insert them into MUC or chats between users, otherwise it will create a pop-up like normal messages. - **chat** Chat messages will open up a chat dialog with users.

## Other

This section contains a considerable list of options and settings affecting server functions.

### Activate log tracker for a user

This allows you to set a log file to track a specific user. Set the bare or full JID of the user you want to log, and a name of the files you wish the log to be written to. The files will be written in the root Tigase directory unless you give a directory like logs/filename. The log files will be named with a .0 extension and will be named .1, .2, .3 and so on as each file reaches 10MB by default. filename.0 will always be the most recent. Logging will start once the command has been issued, and cease once the server restarts.

### Add SSL certificate

Here you can add SSL certificates from PEM files to specific virtual hosts. Although Tigase can generate its own self-signed certificates, this will override any default certificates. The certificates cannot contain a passphrase, or be encrypted. Be sure that the contents contain both the certificate and private key data. You also have the option to save the certificate to disk, making the change permanent.

### Add listener script

This section allows you to create a custom function for the eventbus component. These scripts can have the server conduct certain operations if set criteria are met. You may write the script in either Groovy or ECMAScript. Please see the eventbus section for more details.

### Add Monitor Task

You can write scripts for Groovy or ECMAScript to add to monitor tasks here. This only adds the script to available scripts however, you will need to run it from another prompt. Note that these scripts may only work with the monitor component.

### Add Monitor Timer Task

This section allows you to add monitor scripts in Groovy while using a delay setting which will delay the start of the script.

### Add New Item - ext

Depending on whether you have any external components loaded or not, this may show. This allows you to add additional external components to the running instance of Tigase.

### Add New Item - Vhost

This allows you to add new virtual hosts to the XMPP server. A breakdown of the fields is as follows:

- Domain name: the full domain name of the new vhost. Tigase will not add anything to this domain, so if it is to be a subdomain of example.com, you will need to enter sub.domain.com.
- Enabled: Whether the domain is turned on or off.
- Anonymous enabled: Allow anonymous logins.
- In-band registration: Whether or not to allow users to register accounts upon login.
- TLS required: Require logins to the vhost to conduct a TLS handshake before opening streams.
- S2S secret: a server-generated code to differentiate traffic between servers, typically there is no need to enter your own, but you may if you need to get into low level code.
- Domain filter policy: Sets the filter policy for this domain, see This section for a description of the rules.
- Domain filter domains: a specific setting to restrict or control cross domain traffic.
- Max users: maximum users allowed to be registered to the server.
- Allowed C2S, BOSH, Websocket ports: Comma separated list of ports that this vhost will check for all of these services.
- Presence forward address: specific address where presence information is forwarded too. This may be handy if you are looking to use a single domain for presence processing and handling.
- Message forward address: Specific address where all messages will be sent too. This may be useful to you if you have a single server handling AMP or message storage and want to keep the load there.
- Other Parameters: Other settings you may wish to pass to the server, consider this a section for options after a command.
- Owner: The owner of the vhost who will also be considered an administrator.
- Administrators: comma separated list of JIDs who will be considered admins for the vhost.
- XEP-0136 Message Archiving Enabled: Whether to turn on or off this feature.
- XEP-0136 Required store method: If XEP-0136 is turned on, you may restrict the portion of message that is saved. This is required for any archiving, if null, any portion of the message may be stored.
- Client certificate required: Whether the client should submit a certificate to login.
- Client certificate CA: The Certificate Authority of the client certificate.
- XEP-0136 retention period: integer of number of days message archives will be set.
- Trusted JIDs: Comma separated list of JIDs who will be added to the trusted list, these are JIDS that may conduct commands, edit settings, or other secure work without needed secure logins.
- XEP-0136 retention type: Sets the type of data that retention period will use. May be User defined (custom number type), Unlimited, or Number of Days.
- XEP-0136 - store MUC messages: Whether or not to store MUC messages for archiving. Default is user, which allows users to individually set this setting, otherwise true/false will override.

- see-other-host redirection enabled: in servers that have multiple clusters, this feature will help to automatically repopulate the cluster list if one goes down, however if this is unchecked, that list will not change and may attempt to send traffic to a down server.
- XEP-0136 Default store method: The default section of messages that will be stored in the archive.

## Change user inter-domain communication permission

Here you can restrict users to be able to communicate on specific domains, this works similar to the domain filtering policy using the same rule sets. For more details, see Domain Based Packet Filtering section for rule details and specifics. Note that the changes may be made to multiple JIDs at the same time.

## Connections Time

Lists the longest and average connection time from clients to servers.

## Create Node

This section allows you to create a new node for the pubsub component. Here is a breakdown of the fields:

- The node to create: this is the name of the node that will be created.
- Owner JID: user JID who will be considered the owner of the node.
- pubsub#node type: sets the type of node the the new node will be. Options include:
  - **leaf** Node that can publish and be published too.
  - **collection** A collection of other nodes.
- A friendly name for the node: Allows spaces and other characters to help differentiate it from other nodes.
- Whether to deliver payloads with event notifications: as it says, to publish events or not.
- Notify subscribers when the configuration changes: default is false
- Persist items to storage: whether or not to physically store items in the node.
- Max # of items to persist: Limit how many items are kept in the node archive.
- The collection with which the node is affiliated: If the node is to be in a collection, place that node name here.
- Specify the subscriber model: Choose what type of subscriber model will be used for this node. Options include:
  - **authorize** - Requires all subscriptions to be approved by the node owner before items will be published to the user. Also only subscribers may retrieve items.
  - **open** - All users may subscribe and retrieve items from the node.
  - **presence** - Typically used in an instant message environment. Provides a system under which users who are subscribed to the owner JID's presence with a from or both subscription may subscribe from and retrieve items from the node.

- **roster** - This is also used in an instant message environments, Users who are both subscribed to the owners presence AND is placed in specific allowed groups by the roster are able to subscribe to the node and retrieve items from it.
- **whitelist** - Only explicitly allowed JIDs are allowed to subscribe and retrieve items from the node, this list is set by the owner/administrator.
- Specify the Publisher model: Choose what type of publisher model will be used for this node. Options include:
  - **open** - Any user may publish to this node.
  - **publishers** - Only users listed as publishers may be able to publish.
  - **subscribers** - Only subscribers may publish to this node.
- When to send the last published item: This allows you to decide if and when the last published item to the node may be sent to newly subscribed users.
  - **never** - Do not send the last published item.
  - **on\_sub** - Send the last published item when a user subscribes to the node.
  - **on\_sub\_and\_presence** - Send the last published item to the user after a subscription is made, and the user is available.
- The domains allowed to access this node: Comma separated list of domains for which users can access this node. By default is blank, and there is no domain restriction.
- Whether to deliver items to available users only: Items will only be published to users with available status if this is selected.
- Whether to subscription expired when subscriber going offline: This will make all subscriptions to the node valid for a single session and will need to be re-subscribed upon reconnect.
- The XSL transformation which can be applied to payloads in order to generate an appropriate message body element: Since you want a properly formatted <body> element, you can add an XSL transformation here to address any payloads or extra elements to be properly formatted here.
- The URL of the XSL transformation which can be applied to payloads in order to generate an appropriate message body element: This would be the URL of the XSL Transform, e.g. <http://www.w3.org/1999/XSL/Transform>.
- Roster groups allowed to subscribe: a list of groups for whom users will be able to subscribe. If this is blank, no user restriction will be imposed.
- Notify subscribers when owner changes their subscription or affiliation state: This will have the node send a message in the case of an owner changing affiliation or subscription state.
- Allows get list of subscribers for each subscriber: Allows subscribers to produce a list of other subscribers to the node.
- Whether to sort collection items by creation date or update time: options include
  - **byCreationDate** - Items will be sorted by the creation date, i.e. when the item was made.

- **byUpdateTime** - Items will be sorted by the last update time, i.e. when the item was last edited/published/etc..

## DNS Query

A basic DNS Query form.

## Default config - Pubsub

Here you may set the default configuration for any new pubsub node. These changes will be made for all future nodes, but will not affect currently active nodes.

## Default room config

This page allows admins to set the default configuration for any new MUC rooms that may be made on the server.

## Delete Monitor Task

This removes a monitor task from the list of available monitor scripts. This action is not permanent as it will revert to initial settings on server restart.

## Delete Node

Provides a space to remove a node from the server. It must be the full name of the node, and only one node can be removed at a time.

## Deleting ALL Nodes

This page allows the logged in admin to delete all nodes from the associated vhost. This change is irreversible, be sure to read and check the box before submitting the command.

## Fix User's Roster

You can fix a users roster from this prompt. Fill out the bare JID of the user and the names you wish to add or remove from the roster. This will NOT edit a user's roster, but rather compare client roster to database and fix any errors between them.

## Fix User's Roster on Tigase Cluster

This does the same as the Fix User's Roster, but can apply to users who may not be logged into the local vhost, but are logged into a clustered server.

## Get User Roster

As the title implies this gets a users' roster and displays it on screen. You can use a bare or full JID to get specific rosters.

## Get any file

Enables you to see the contents of any file in the tigase directory. By default you are in the root directory, if you wish to go into directory use the following format: logs/tigase.log.0

## Get Configuration File

If you don't want to type in the location of a configuration file, you can use this prompt to bring up the contents of either `tigase.conf` or `config.tdsl`.

## Get config.tdsl File

Will output the current `config.tdsl` file, this includes any modifications made during the current server session.

## Get list available commands

This may be listed multiple times for different components, but this will do as the section suggest and list available commands for that particular component.

## Load test

Here you can run a test with the `pubsub` component on any node to test functionality and proper settings for the node.

## Load Errors

Will display any errors the server encounters in loading and running. Can be useful if you need to address any issues.

## New command script

This space allows you to create a new command script that will work within the associated component. Note that under the hyperlinked title, there is a listing of `muc.server.org` or `pubsub.server.org`, use these to determine where the new command will operate.

## OAuth Credentials

This allows the setting of new custom OAuth credentials for the server, and you can also require the use of OAuth tokens for users when they login. This is a setting for the specific host you are logged into. If you are logged into `xmpp1.domain.com`, it will not affect settings for `xmpp2.domain.com`.

## Pre-Bind BOSH user session

This allows a JID to be paired with a BOSH session before that user logs in, can reduce CPU use if you have a user that logs in via BOSH on a regular basis, or a web client that will regularly connect. You may also specify `HOLD` and `WAIT` integers to affect how BOSH operates with the associated JID.

## Publish item to node

This window allows you to not only test, but publish an item to the specified node. All fields must be filled in in order to avoid the server dropping an improperly formatted stanza.

## Read ALL nodes

This will load a tree of `pubsub` nodes in memory, it will not output anything as it is mainly for developer use.



## Rebuild database

This will force Tigase to rebuild databases for the pubsub component, this may be useful for pubsub subscribers who continue to get pushed events after they unsubscribe.

## Reload component repository

This will reload any vhosts that the server is running. This may be useful if one is disconnected or broken during runtime.

## Remove an item

This will remove a running vhost from the server, you will be presented with a list to pick from.

## Remove command script

Like new command script, take a look at the subheading to determine which component you want to remove the script from. Once there, select the command you wish to remove from the server. If remove from disk is selected, then the change will be permanent. Otherwise, the command will be removed until the next server restart.

## Remove listener script

Select from a list the listener script you wish to remove. This will only affect custom listener scripts added to the eventbus component.

## Remove room

This provides fields to remove a room from the MUC component. you may suggest an alternative room which will move occupants to the alternative room once the current one is removed.

## Retrieve items

Here you can retrieve items from PubSub nodes, this simulates the get IQ stanza from the pubsub component. - Service name - The address of the pubsub component. - Node name - Item node to retrieve items from. - Item ID - The item ID of the item you wish to retrieve. - Items Since - UTC timestamp to start search from: YYYY-MM-DDTHH:MM:SSZ

## S2S Bad State Connections

This will list any connections to other servers that are considered bad or stale. This will populate very rarely as Tigase automatically adjusts around clustered servers that go down. In the event a connection stays bad, it is recommended to reset those connections in the next space.

## S2S Reset Bad State Connections

This will reset the connections with other servers that are considered bad and have shown up in the S2S Bad State Connections page.

## S2S Get CID Connection State

For internal developer use only.

## Subscribe to a node

This provides a space for an administrator to manually have a JID subscribe to a particular node.

## Unsubscribe from node

Here you can unsubscribe users from a particular node. Users can be a comma separated list.

## Update item configuration

Typically you will see only one item for vhost-man, but some additional components (ie. ext) may provided them as well. They each have their own sections, but provide for a plethora of server options. Changes to the server are done in real time, and may not be permanent.

### vhost-man

You will be presented with a list of domains that Tigase is currently hosting, you will be able to change settings for one domain at a time using this function. Once a domain is selected, you will be able to set or change the following settings:

- Domain name: the full domain name of the new vhost. Tigase will not add anything to this domain, so if it is to be a subdomain of example.com, you will need to enter sub.domain.com.
- Enabled: Whether the domain is turned on or off.
- Anonymous enabled: Allow anonymous logins.
- In-band registration: Whether or not to allow users to register accounts upon login.
- TLS required: Require logins to the vhost to conduct a TLS handshake before opening streams.
- S2S secret: a server-generated code to differentiate traffic between servers, typically there is no need to enter your own, but you may if you need to get into low level code.
- Domain filter policy: Sets the filter policy for this domain, see This section for a description of the rules.
- Domain filter domains: a specific setting to restrict or control cross domain traffic.
- Max users: maximum users allowed to be registered to the server.
- Allowed C2S, BOSH, Websocket ports: Comma separated list of ports that this vhost will check for all of these services.
- Presence forward address: specific address where presence information is forwarded too. This may be handy if you are looking to use a single domain for presence processing and handling.
- Message forward address: Specific address where all messages will be sent too. This may be useful to you if you have a single server handling AMP or message storage and want to keep the load there.
- Other Parameters: Other settings you may wish to pass to the server, consider this a section for options after a command.
- Owner: The owner of the vhost who will also be considered an administrator.
- Administrators: comma separated list of JIDs who will be considered admins for the vhost.
- XEP-0136 Message Archiving Enabled: Whether to turn on or off this feature.

- XEP-0136 Required store method: If XEP-0136 is turned on, you may restrict the portion of message that is saved. This is required for any archiving, if null, any portion of the message may be stored.
- Client certificate required: Whether the client should submit a certificate to login.
- Client certificate CA: Client Certificate Authority.
- XEP-0136 retention period: Integer of number of days message archives will be set.
- Trusted JIDs: Comma separated list of JIDs who will be added to the trusted list, these are JIDs that may conduct commands, edit settings, or other secure work without needed secure logins.
- XEP-0136 retention type: Sets the type of data that retention period will use. May be User defined (custom number type), Unlimited, or Number of Days.
- XEP-0136 - store MUC messages: Whether or not to store MUC messages for archiving. Default is user, which allows users to individually set this setting, otherwise true/false will override.
- see-other-host redirection enabled: in servers that have multiple clusters, this feature will help to automatically repopulate the cluster list if one goes down, however if this is unchecked, that list will not change and may attempt to send traffic to a down server.
- XEP-0136 Default store method: The default section of messages that will be stored in the archive.

## Update user roster entry

This section allows admins to edit individual users rosters, although it provides similar functionality to fix users roster, this is designed for precision editing of a user roster.

- Roster owner JID: The BareJID of the user roster you wish to edit.
- JID to manipulate: The specific BareJID you want to add/remove/change.
- Comma separated groups: Groups you wish to add the JID too.
- Operation Type: What function will be performed?
  - **Add** - Add the JID to manipulate to the owner JID's roster and groups.
  - **Remove** - Remove the JID to manipulate from the owner JID's roster and groups.
- Subscription type: The type of subscription stanza that will be sent to the server, and subsequently between the two users will be employed.
  - **None** - Select this if neither the owner or the user to be manipulated wishes to receive presence information.
  - **From** - The Roster Owner will not receive presence information from the JID to manipulate, but the opposite will be true.
  - **To** - The JID to manipulate will not receive presence information from the Roster Owner, but the opposite will be true.
  - **Both** - Both JIDs will receive presence information about each other.

## Update user roster entry extended version

This section is an expanded version of the previous one, all fields already specified are the same with these additions:

- Roster owner name: A friendly name or nickname if you wish to change/create one. **not required**
- Comma separated of owner groups: Groups that the user wants to join/leave. **not required**
- Roster item JID: The specific JID that needs to be edited.
- Roster item name: A friendly name or nickname that will be changed/created. **not required**
- Comma separated list of item groups: A group or list of groups that the roster item JID will be added to/removed from.
- Action:
  - **Add/update item** - Will add or update the item JID in the roster owner's roster.
  - **Remove item** - Will remove the item JID from the roster owner's roster.
  - **Add/update both rosters** - Will add or update the item in both roster owner and roster item's roster.
  - **Remove from both rosters** - Will remove the item from both roster owner and roster item's roster.

## Scripts

This section will enable administrators to custom write or enter their own scripts for specific components. Each active component will have an entry for new and remove command scripts and scripts written there will be for that component.

### New Command Script

- Description: A friendly name of the script, will be the title of the link in the menu on the left.
- Command ID: Internal command that Tigase will use when referencing this script.
- Group: The group for the script, which may be any of the headings on the left (Configuration, Example scripts, Notifications, Other etc..) or your own. If no group exists, a new one will be created.
- Language: The language the script is written in. Currently Tigase supports Groovy and EMCAScript.
- Script text: the fulltext of the script.
- Save to disk: Scripts that are saved to disk will be permanently stored in the server's directory /scripts/admin/[Component]/commandID.js **NOTE** Scripts that are NOT saved to disk will not survive a server restart.

### Remove Command Script

As with New Command Script, there is an entry for each component. This page will provide a space to remove commands for the selected component. You will be provided a list of scripts associated with that component. You also have the open to remove from disk, which will permanently delete the script from the hard drive the server is on. If this is unchecked, the script will be unavailable until the next restart.

## Statistics

This section is more useful to test statistics scripts and components, as many of them produce very small amounts of information, however these may be collected by other components or scripts for a better information display.

## Get User Statistics

Provides a script output of user statistics including how many active sessions are in use, number of packets used, specific connections and their packet usage and location. All resources will return individual stats along with IP addresses.

## Get Active User List

Provides a list of active users under the selected domain within the server. An active user is considered a user currently logged into the XMPP server.

## Get list of idle users

Provides a list of users who are idle on the server.

## Get list of online users

Provides a list of users who are currently online.

## Get number of active users

Provides a list of active users, users who are not idle or away.

## Get number of idle users

Provides a number of idle users.

## Get top active users

Will produce a list of user-limited users who are considered most active in packets sent.

# Users

## Add User

Here you can add new users to any domain handled by vHosts, users are added to database immediately and are able to login. **NOTE: You cannot bestow admin status to these users in this section.**

## Change User Password

This enables you to change the password of any user in the database. Although changes will take effect immediately, users currently logged in will not know the password has been changed until they try to log in again.

## Delete User

This removes the user or users (comma separated) from the database. The deleted users will be kicked from the server once submit is clicked.

## End user session

Disconnects the current selected user by ending their session with the server.

## Get User Info

This section allows admins to get information about a specific user including current connections as well as offline and online messages awaiting delivery.

## Get registered user list

This will display all registered users for the selected domain up to the number specified.

## Modify User

Allows you to modify some user details including E-mail and whether it is an active user.

# Tigase Web Client

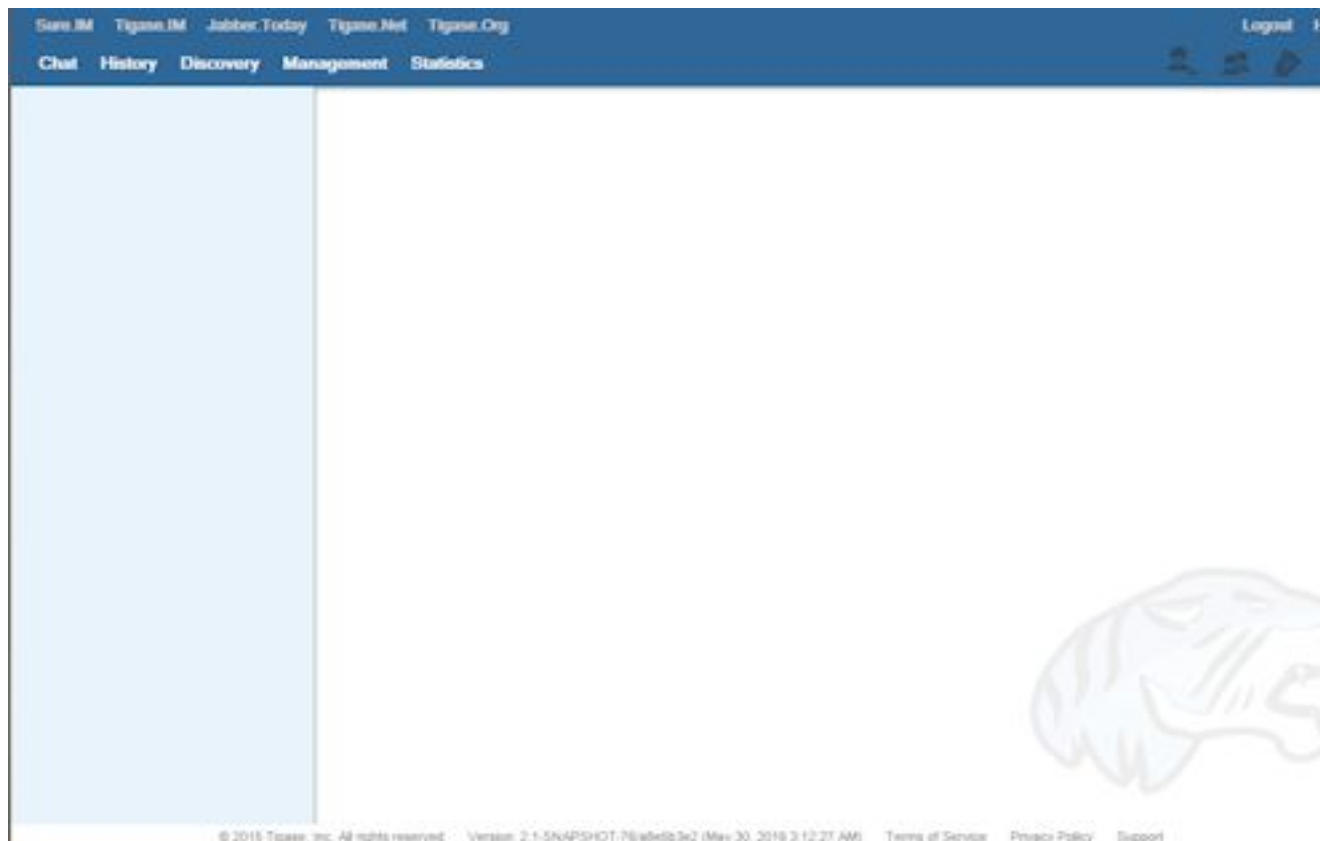
Tigase now has a fully featured XMPP client built right into the HTTP interface. Everything you would expect from an XMPP client can now be done from the comfort of your browser window with no software install required!

The web client is active and available by default on servers v7.2.0 and later.

To access the client, point a browser to the following address: xmpp.your-server.net:8080/ui/

It will ask you for a login, any bare JID of users registered with the server will work. **NOTE: Use your bare JID for login**

Once you have logged in successfully, you will be presented with the following screen.



The commands are broken into categories shown here. All changes made in these sections are instant and should be seen the same as if you were using an external XMPP client like Psi.

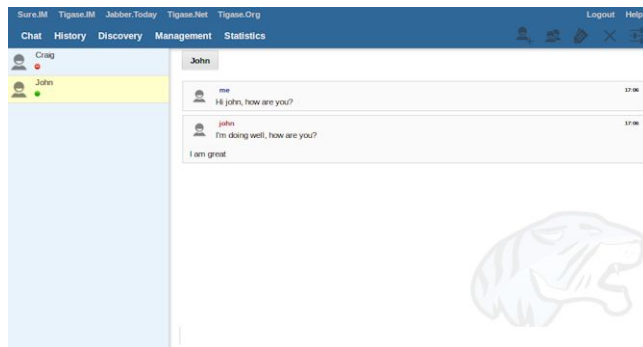
**NOTE** The BOSH client will automatically translate all requests to the server name. In some rare cases this may not be resolvable by the browser and you will be unable to login. Should that happen, you may disable that feature using the following line in your config.tdsl:

```
bosh {
  - 'send-node-hostname' = false
}
```

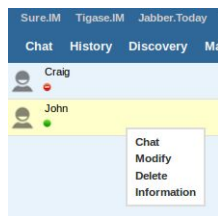
You may have to specifically designate the bosh URL when using the advanced tag in the login screen.

## Chat

This is the default window, and your main interface for chatting inside XMPP with this server. **NOTE: you can only communicate to users logged onto the current server, or connected clusters** Users from your roster will be on the left panel, the right all active discussions and MUCs, as well as the currently selected chat will be available.



Users that are logged in and on your roster will be displayed on the left side. Double-clicking will bring up a new chat window with the user. You can Right-click on them to bring up a sub menu with the following;



- **Chat** replicates a double-click and opens a new window for chat.
- **Modify** brings up a dialogue that allows you to change the JID of the contact, a nickname, and group.
- **Delete** removes the user from your roster. This will also remove subscription authorization for the selected user to receive presence information effectively removing you from their roster. **NOTE: this will not block user packets from your JID**
- **Info** brings up the User Info (this is the disco#info command for the selected user)

The top right section has a few icons with specific functionality, they are;



adds a new user to your roster.



creates a new Multi-user chatroom.



allows you to edit your user information such as picture and nickname.



closes the active chat window.



provides a place to change your password or publish changes to your user info. **NOTE: you are limited to changing the General fields**

## Discovery

This is your service discovery panel, which breaks down by component in the sidebar. Each component name and its associated JID is listed to help you find what you need. Most components give you an option to Execute commands with a few exceptions allowing browsing and the ability to join a MUC.

**Browse** allows you to dig deeper into certain components; for example list the chatrooms available in the MUC component. At the top of the page the specific JID of the component are you in will be displayed. This is a text field, and can be edited to reflect the JID of the component (or just the server name) to navigate.



**Join to Room** will join you to a MUC room that is selected. Alternatively, selecting Join to Room while MUC component is selected, you can join and start a new MUC room.

**Execute Command** Provides a hierarchy of commands and options to view and edit settings, run commands and scripts, view contents of files, and see statistics. Since each Component can have a unique structure it is best to explore each to see what options are available.

## Management

This is an advanced window for settings and management for the XMPP server.

## Configuration

Here you can manage some of the server settings.

## Notifications

This section has one simple command: to be able to send a mass message to all logged in users. You may choose to change the type of message to headline or Normal which will show as a pop-up in most XMPP clients. Chat messages will open up a chat dialog with users.

## Other

This section contains a considerable list of options and settings affecting server functions.

## Activate log tracker for a user

This allows you to set a log file to track a specific user. Set the bare or full JID of the user you want to log, and a name of the files you wish the log to be written to. The files will be written in the root Tigase directory unless you give a directory like logs/filename. The log files will be named with a .0 extension



and will be named .1, .2, .3 and so on as each file reaches 10MB by default. filename.0 will always be the most recent. Logging will cease once the server restarts.

## Add SSL certificate

Here you can add SSL certificates from PEM files to specific virtual hosts. Although Tigase can generate its own self-signed certificates, this will override those default certificates.

## Add Monitor Task

You can write scripts for Groovy or ECMAScript to add to monitor tasks here. This only adds the script to available scripts however, you will need to run it from another prompt.

## Add Monitor Timer Task

This section allows you to add monitor scripts in Groovy while using a delay setting which will delay the start of the script.

## Add New Item - ext

Provides a method to add external components to the server. By default you are considered the owner, and the Tigase load balancer is automatically filled in.

## Add New Item - Vhost

This allows you to add new virtual hosts to the XMPP server. A breakdown of the fields is as follows:

- Domain name: the full domain name of the new vhost. Tigase will not add anything to this domain, so if it is to be a subdomain of example.com, you will need to enter sub.domain.com.
- Enabled: Whether the domain is turned on or off.
- Anonymous enabled: Allow anonymous logins.
- In-band registration: Whether or not to allow users to register accounts upon login.
- TLS required: Require logins to the vhost to conduct a TLS handshake before opening streams.
- S2S secret: a server-generated code to differentiate traffic between servers, typically there is no need to enter your own, but you may if you need to get into low level code.
- Domain filter policy: Sets the filter policy for this domain, see This section for a description of the rules.
- Domain filter domains: a specific setting to restrict or control cross domain traffic.
- Max users: maximum users allowed to be registered to the server.
- Allowed C2S, BOSH, Websocket ports: Comma separated list of ports that this vhost will check for all of these services.
- Presence forward address: specific address where presence information is forwarded too. This may be handy if you are looking to use a single domain for presence processing and handling.
- Message forward address: Specific address where all messages will be sent too. This may be useful to you if you have a single server handling AMP or message storage and want to keep the load there.

- Other Parameters: Other settings you may wish to pass to the server, consider this a section for options after a command.
- Owner: The owner of the vhost who will also be considered an administrator.
- Administrators: comma separated list of JIDs who will be considered admins for the vhost.
- XEP-0136 Message Archiving Enabled: Whether to turn on or off this feature.
- XEP-0136 Required store method: If XEP-0136 is turned on, you may restrict the portion of message that is saved. This is required for any archiving, if null, any portion of the message may be stored.
- Client certificate required: Whether the client should submit a certificate to login.
- Client certificate CA: The Certificate Authority of the client certificate.
- XEP-0136 retention period: integer of number of days message archives will be set.
- Trusted JIDs: Comma separated list of JIDs who will be added to the trusted list, these are JIDS that may conduct commands, edit settings, or other secure work without needed secure logins.
- XEP-0136 retention type: Sets the type of data that retention period will use. May be User defined (custom number type), Unlimited, or Number of Days.
- XEP-0136 - store MUC messages: Whether or not to store MUC messages for archiving. Default is user, which allows users to individually set this setting, otherwise true/false will override.
- see-other-host redirection enabled: in servers that have multiple clusters, this feature will help to automatically repopulate the cluster list if one goes down, however if this is unchecked, that list will not change and may attempt to send traffic to a down server.
- XEP-0136 Default store method: The default section of messages that will be stored in the archive.

## Change user inter-domain communication permission

You can restrict users to only be able to send and receive packets to and from certain virtual hosts. This may be helpful if you want to lock users to a specific domain, or prevent them from getting information from a statistics component.

## Connections Time

Lists the longest and average connection time from clients to servers.

## DNS Query

A basic DNS Query form.

## Default config - Pubsub

This section enables you to change the default pubsub node configuration for all future nodes. **Note: these changes will be reset on server restart.** - pubsub#node type: sets the type of node the the new node will be. Options include:

- **leaf** Node that can publish and be published too.
- **collection** A collection of other nodes.

- A friendly name for the node: Allows spaces and other characters to help differentiate it from other nodes.
- Whether to deliver payloads with event notifications: as it says, to publish events or not.
- Notify subscribers when the configuration changes: default is false
- Persist items to storage: whether or not to physically store items in the node.
- Max # of items to persist: Limit how many items are kept in the node archive.
- The collection with which the node is affiliated: If the node is to be in a collection, place that node name here.
- Specify the subscriber model: Choose what type of subscriber model will be used for this node. Options include:
  - **authorize** - Requires all subscriptions to be approved by the node owner before items will be published to the user. Also only subscribers may retrieve items.
  - **open** - All users may subscribe and retrieve items from the node.
  - **presence** - Typically used in an instant message environment. Provides a system under which users who are subscribed to the owner JID's presence with a from or both subscription may subscribe from and retrieve items from the node.
  - **roster** - This is also used in an instant message environments, Users who are both subscribed to the owners presence AND is placed in specific allowed groups by the roster are able to subscribe to the node and retrieve items from it.
  - **whitelist** - Only explicitly allowed JIDs are allowed to subscribe and retrieve items from the node, this list is set by the owner/administrator.
- Specify the Publisher model: Choose what type of publisher model will be used for this node. Options include:
  - **open** - Any user may publish to this node.
  - **publishers** - Only users listed as publishers may be able to publish.
  - **subscribers** - Only subscribers may publish to this node.
- When to send the last published item: This allows you to decide if and when the last published item to the node may be sent to newly subscribed users.
  - **never** - Do not send the last published item.
  - **on\_sub** - Send the last published item when a user subscribes to the node.
  - **on\_sub\_and\_presence** - Send the last published item to the user after a subscription is made, and the user is available.
- The domains allowed to access this node: Comma separated list of domains for which users can access this node. If left blank there is no domain restriction.
- Whether to deliver items to available users only: Items will only be published to users with available status if this is selected.

- Whether to subscription expired when subscriber going offline: This will make all subscriptions to the node valid for a single session and will need to be re-subscribed upon reconnect.
- The XSL transformation which can be applied to payloads in order to generate an appropriate message body element: Since you want a properly formatted <body> element, you can add an XSL transformation here to address any payloads or extra elements to be properly formatted here.
- The URL of the XSL transformation which can be applied to payloads in order to generate an appropriate message body element: This would be the URL of the XSL Transform, e.g. <http://www.w3.org/1999/XSL/Transform>.
- Roster groups allowed to subscribe: a list of groups for whom users will be able to subscribe. If this is blank, no user restriction will be imposed.
- Notify subscribers when owner changes their subscription or affiliation state: This will have the node send a message in the case of an owner changing affiliation or subscription state.
- Allows get list of subscribers for each subscriber: Allows subscribers to produce a list of other subscribers to the node.
- Whether to sort collection items by creation date or update time: options include
  - **byCreationDate** - Items will be sorted by the creation date, i.e. when the item was made.
  - **byUpdateTime** - Items will be sorted by the last update time, i.e. when the item was last edited/published/etc..

## Default room config

Allows you to set the default configuration for new MUC rooms. This will not be able to modify current in use and persistent rooms.

## Delete Monitor Task

This removes a monitor task from the list of available monitor scripts. This action is not permanent as it will revert to initial settings on server restart.

## Fix User's Roster

You can fix a users roster from this prompt. Fill out the bare JID of the user and the names you wish to add or remove from the roster. You can edit a users roster using this tool, and changes are permanent.

## Fix User's Roster on Tigase Cluster

This does the same as the Fix User's Roster, but can apply to users in clustered servers.

## Get User Roster

As the title implies this gets a users' roster and displays it on screen. You can use a bare or full JID to get specific rosters.

## Get any file

Enables you to see the contents of any file in the tigase directory. By default you are in the root directory, if you wish to go into directory use the following format: logs/tigase.log.0

## Get Configuration File

If you don't want to type in the location of a configuration file, you can use this prompt to bring up the contents of either `tigase.conf` or `config.tdsl`.

## Get config.tdsl File

Will output the current `config.tdsl` file, this includes any modifications made during the current server session.

## Load Errors

Will display any errors the server encounters in loading and running. Can be useful if you need to address any issues.

## New command script - Monitor

Allows you to write command scripts in Groovy and store them physically so they can be saved past server restart and run at any time. Scripts written here will only be able to work on the Monitor component.

## New command script - MUC

Allows you to write command scripts in Groovy and store them physically so they can be saved past server restart and run at any time. Scripts written here will only be able to work on the MUC component.

## OAuth credentials

Uses OAuth to set new credentials and enable or disable a registration requirement with a signed form.

## Pre-Bind BOSH user session

Allows admins to pre-bind a BOSH session with a full or bare JID (with the resource automatically populated on connection). You may also specify `HOLD` or `WAIT` parameters.

## Reload component repository

This will show if you have any external components and will reload them in case of any stuck threads.

## Scripts

This section provides a list of command scripts for all active components. Each component has the following options - **New command script** provides a method to author new command scripts for specific components written in EMCAScript or Groovy. You do have an option to save the script to disk which will make the script permanent within the server. - **Remove command script** allows you to remove the selected script from the repository. If Remove from disk is not checked, the script will be unavailable until server restart. If it is, it will be permanently removed from the server.

Newly made commands will be listed under the Group listing in the left column.

## Statistics

These statistics might be more useful as script results yield small bits of data, but you may find them useful when looking for server loads or finding user issues.

## Get User Statistics

Provides a script output of user statistics including how many active sessions are in use, number of packets used, specific connections and their packet usage and location. All resources will return individual stats along with IP addresses.

## Get Active User List

Provides a list of active users under the selected domain within the server. An active user is considered a user currently logged into the XMPP server.

## Get list of idle users

This will list all idle users separated by vhost.

## Get list of online users

This will list users separated by the vhost they are connected to. The list will include the bare JID as well as any resources for that JID.

## Get number of active users

This displays the number of current active users.

## Get number of idle users

This section returns the number of active users per specific vhost.

## Get top active users

This will list the top number of active users by packets sent and online time. This list will only be built with users currently online and from all vhosts.

# Users

## Add New User

Here you can add new users to any domain handled by vHosts, users are added to database immediately and are able to login. **NOTE: You cannot bestow admin status to these users in this section.**

## Change user password

Allows for admins to change the password of a specific user without needing to know the original password for the selected bare JID. Users currently logged in will not know password has been changed until they attempt to re-login.

## Delete user

Provides a text window for admins to input the bare JID of the user they wish to remove from the server.

## Get User Info

This section allows admins to get information about a specific user including current connections as well as offline and online messages awaiting delivery.

## **Get registered user list**

Provides a list of vhosts to search and a maximum number of users to list. Once run, the script will display a list of registered bare JIDs of users from the selected vhost.

## **Modify User**

Allows you to modify some user details including E-mail and whether it is an active user.

---

# Chapter 15. HTTP File Upload component

Tigase's HTTP File Upload component is an implementation of XEP-0363 HTTP File Upload [<http://xmpp.org/extensions/xep-0363.html>] specification. This allows file transfer between XMPP clients by uploading a file to HTTP server and sending only link to download file to recipient.

This implementation makes use of the HTTP server used by Tigase XMPP Server and Tigase HTTP API component to provide web server for file upload and download.

By default this component is **disabled** and needs to be enabled in configuration file before it can be used. Another requirement is that the proper database schema needs to be applied to database which will be used by component.

## Enabling HTTP File Upload Component

**Configuration.**

```
upload() {}
```

## Metadata repository

Running the component requires a repository where it can store information about allocated slots. For this, a metadata repository is used. It is possible to specify a specific implementation of `FileUploadRepository` for every domain.

By default, metadata for all domains will be stored in the `default` repository. Implementation of which will be selected based on kind of data source defined as `default`.

## DummyFileUploadRepository

This is very simple repository which does not store any data. Due to that, it can be very fast! However, it is not able to remove old uploads and apply any upload limits.

## JDBCFileUploadRepository

This repository implementation stores data in database used to store procedures and functions. By default, data should be stored in the `tig_hfu_slots` table but it can be changed by modification of stored procedures or reconfiguration of the repository implementation to use different stored procedures and functions than provided.

## Storage

Component contains a pluggable storage mechanism, which means that it is relatively easy to implement custom storage provisions. By default `DirectoryStore` based storage is used.

Currently following storage providers are available out of the box.



## DirectoryStore

This storage mechanism places files in subdirectories with names that correspond to the id of allocated slot [http://xmpp.org/extensions/xep-0363.html#intro:]. If required, it is possible to group all slot directories allocated by single user in a directory containing this user name.

By default there is no redundancy if this store is used in clustered environment. Every file will be stored on a single cluster node.

Available properties:

path	Contains path to directory in which subdirectory with files will be created on the local machine. ( <i>default: data/upload</i> )
group-by-user	Configures if slots directories should be grouped in user directories. ( <i>default: false</i> )

## Logic

Logic is responsible for generation of URI and applying limits. It groups all configuration settings related to allocation of slots, etc.

Available properties:

local-only	Allow only users with accounts on the local XMPP server to use this component for slot allocation. ( <i>default: true</i> )
max-file-size	Set maximum size of a single allocated slot (maximum file size) in bytes. ( <i>default: 5000</i> )
port	Specifies the port which should be used in generating the upload and download URI. If it is not set, then secured (HTTPS) server port will be used if available, and plain HTTP in other case. ( <i>default: not set</i> )
protocol	Protocol which should be used. <b>This is only used in conjunction with port.</b> Possible values are: <ul style="list-style-type: none"><li>• http</li><li>• https</li></ul>
serverName	Server name to use as domain part in generated URI. ( <i>default: server hostname</i> )
upload-uri-format	Template used in generation of URI for file upload. ( <i>default: {proto}://{serverName}:{port}/upload/{userId}/{slotId}/{filename}</i> )
download-uri-format	Template used in generation of URI for file download. ( <i>default: {proto}://{serverName}:{port}/upload/{slotId}/{filename}</i> )

## URI template format

Every block in the template between { and } is a named part which will be replaced by the property value during generation of URI for slot.

Blocks possible to use:

proto	Name of protocol.
serverName	Domain name of server.
port	Port on which HTTPS (or HTTP) server is listening.
userId	JID of user requesting slot allocation.
domain	Domain of user requesting slot allocation.
slotId	Generated ID of slot.
filename	Name of file to upload.

### Note

slotId and filename are required to be part of every URI template.

### Warning

Inclusion of userId or domain will speed up the lookup for slot id during upload and download operation if more than one metadata repository is configured. However, this may lead to leak of user JID or user domain if message with URI containing this part will be send to recipient which is unaware of the senders' JID (ie. in case of anonymous MUC room).

## File upload expiration

From time to time it is required to remove expired file to make place for new uploads. This is done by the `expiration` task.

Available properties:

expiration-time	How long the server will keep uploaded files. Value in Java Period format [ <a href="https://docs.oracle.com/javase/8/docs/api/java/time/Period.html#parse-java.lang.CharSequence-">https://docs.oracle.com/javase/8/docs/api/java/time/Period.html#parse-java.lang.CharSequence-</a> ] ( <b>default: P30D - 30 days</b> )
period	How often the server should look for expired files to remove. Value in Java Period format [ <a href="https://docs.oracle.com/javase/8/docs/api/java/time/Period.html#parse-java.lang.CharSequence-">https://docs.oracle.com/javase/8/docs/api/java/time/Period.html#parse-java.lang.CharSequence-</a> ] ( <b>default: P1D - 1 day</b> )
delay	Time since server start up before the server should look for expired files to remove. Value in Java Period format [ <a href="https://docs.oracle.com/javase/8/docs/api/java/time/Period.html#parse-java.lang.CharSequence-">https://docs.oracle.com/javase/8/docs/api/java/time/Period.html#parse-java.lang.CharSequence-</a> ] ( <b>default: 0</b> )
limit	Maximum number of files to remove during a single execution of <code>expiration</code> . ( <b>default: 10000</b> )

## Examples

### Complex configuration example

Configuration with a separate repository for metadata to `example.com` pointing to `file_upload` data source, custom upload and download URI, maximum file size set to 10MB, expiration done every 6 hours and grouping of slot folders by user jid.

### Complex configuration example.

```
upload() {
  logic {
    local-only = false
    max-file-size = 10485760
    upload-uri-format = -'{proto}://{serverName}:{port}/upload/{userId}/{slot}'
    download-uri-format = -'{proto}://{serverName}:{port}/upload/{domain}/{slot}'
  }

  expiration {
    period = P6H
  }

  repositoryPool {
    -'example.com' () {
      data-source = -"file_upload"
    }
  }

  store {
    group-by-user = true
  }
}
```

## Example configuration for clustering with HA

Configuration for high availability in a cluster with common storage at `/mnt/shared` and both servers available as `upload.example.com`

### Example configuration with HA.

```
upload() {
  logic {
    upload-uri-format = -'{proto}://upload.example.com:{port}/upload/{userId}'
    download-uri-format = -'{proto}://upload.example.com:{port}/upload/{domain}'
  }

  store {
    path = -'/mnt/shared/upload'
  }
}
```

---

# Chapter 16. HTTP server

HTTP server instance is provided as `httpServer` by default. The server will only be active and enabled if either the HTTP API component or HTTP File Upload component is enabled. This project uses the default implementation of an http server provided by `HttpServer` [<https://docs.oracle.com/javase/8/docs/jre/api/net/httpserver/spec/com/sun/net/httpserver/package-summary.html>] found embedded in Java JDK.

## Note

This implementation is good only for small installations of if there is **no requirement** for a high performance HTTP server. If this is do not match your requirements, it is recommended to use Jetty as the embedded HTTP server using Tigase HTTP API - Jetty HTTP Server project.

## Dependencies

The default HTTP server implementation requires almost no dependencies as most calls are already embedded within JDK 8. However as a common API for processing HTTP requests is needed, as is the same for HTTP server from JDK and Jetty, we have decided to use HTTP Servlet API in version 3.1.

The required files can be downloaded from Tigase HTTP API project [<https://projects.tigase.org/projects/tigase-http-api/files>] section or using following link `servlet-api-3.1.jar` [<https://projects.tigase.org/attachments/download/1504/servlet-api-3.1.jar>]

Please note that this file is included in dist-max, exe, and jar installer distributions of Tigase XMPP server.

## Configuration Properties

The HTTP server can be configured using any of all of the following properties. Note that these settings only apply to the default implementation provided by Tigase HTTP API.

ports	This property is used to configure on which ports on HTTP server should listen for incoming connections. If it is not set then default port 8080 will be used
connections	It is used to group configurations passed to ports
{port}	For every {port} you can pass separate configuration. To do so you will need to replace {port} with port number, ie. 8080. For every port you can pass following properties:
socket	Sets type of socket used for handling incoming connections. Accepted values are: <ul style="list-style-type: none"><li>• <code>plain</code> - port will work in plain HTTP mode (<b>default</b>)</li><li>• <code>ssl</code> - port will work in HTTPS mode</li></ul>
domain	This property is used to configure domain name of SSL certificate which should be used by HTTP server running on this port (if <code>socket</code> is set to <code>ssl</code> ). If it is not set (or it will be omitted) then Tigase XMPP Server will try to use SSL certificate for the host to which client tries to connect. If there will be no SSL certificate for that domain name, then default SSL certificate of Tigase XMPP Server will be used.

## Additional properties of embedded HTTP server

With embedded HTTP server, you have a few additional properties within `executor` section, which you can pass to adjust this HTTP server.

executor	Name of the subsection	
threads		This property is used to configure the number of threads used to handle HTTP requests, ie. 10
request-timeout		Property used to set timeout for processing a single HTTP request (in milliseconds), ie. 30000
accept-timeout		Property used to set timeout for reading HTTP request headers (in milliseconds), ie. 2000

## Examples

Below are few examples for use in DSL based configuration format and older properties based format.

### HTTPS on port 8443 with SSL certificate for example.com

In configuration file `httpServer` related configuration should look like this:

```
httpServer {
  connections {
    8443 () {
      socket = ssl
      domain = -'example.com'
    }
  }
}
```

### Changing port from 8080 to 8081

```
httpServer {
  connections {
    8080 (active: false) {}
    8081 () {}
  }
}
```

## Usage of Jetty HTTP server as HTTP server

As mentioned before it is possible to use Jetty as HTTP server for improved performance. Jetty API can be used in one of two forms: Standalone and OSGi.

### Standalone

In this case the Jetty instance is created and configured internally by Tigase HTTP API. This allows for the same configuration properties used as for default HTTP server configuration.

**Configuration with use of standalone Jetty HTTP Server.**

```
httpServer (class: tigase.http.jetty.JettyStandaloneHttpServer) {  
    -...  
}
```

**OSGi**

This can only be used when Tigase is running inside OSGi container. If this is used Tigase HTTP API will try to retrieve Jetty HTTP server from OSGi container and use it.

**Note**

Jetty HTTP server instance is not configured by Tigase. We would only use this instance for deployment.

**Configuration in OSGi mode with use of Jetty HTTP Server.**

```
httpServer (class: tigase.http.jetty.JettyOSGiHttpServer) {  
    -...  
}
```

---

# Chapter 17. Tigase Message Archiving Component

Tigase Team <team@tigase.net [mailto:team@tigase.net]> :toc: :numbered: :website: <http://tigase.net>  
:Date: 2016-11-11

Welcome to Tigase Message Archiving component guide

## Tigase Message Archiving Component

Tigase Message Archiving Component originated as implementation of XEP-0136: Message Archiving [<http://xmpp.org/extensions/xep-0136.html>] to add support for archiving of messages exchanged using Tigase XMPP Server. In current version component supports also XEP-0313: Message Archive Management [<http://xmpp.org/extensions/xep-0313.html>] specification to allow easier access to archived messages.

## Announcement

### Major changes

Tigase Message Archiving component has undergone a few major changes to our code and structure. To continue to use Tigase Message Archiving component, a few changes may be needed to be made to your systems. Please see them below:

### Database schema changes

We decided to no longer use *in-code* database upgrade to update database schema of Message Archiving component and rather provide separate schema files for every supported database.

Additionally we moved from *in-code* generated SQL statements to stored procedures which are now part of provided database schema files.

To continue usage of new versions of Message Archiving component it is required to manually load new component database schema, see the section called “Preparation of database” section for informations about that.

### Warning

Loading of new database schema is required to use new version of Message Archiving component.

## New features

### Support for Message Archive Management protocol

Now Tigase Message Archiving component supports searching of archived message using XEP-0313: Message Archive Management [<http://xmpp.org/extensions/xep-0313.html>] protocol.

For details on how to enable this feature look into the section called “Support for MAM”

## Support for using separate database for different domains

Since this version it is possible to use separate archived messages based on domains. This allows you to configure component to store archived message for particular domain to different database.

For more informations please look into the section called “Using separate store for archived messages”

## Additional features

Tigase Message Archiving Component contains few additional features useful for working with message archives.

## Querying in all messages

Feature allows user to search all of his archived messages without a need to specify who was send/receiver of this message. To search in all messages, request sent to retrieve archived messages should not contain with attribute. As a result, when with attribute is omitted <chat/> element of response will not contain with attribute but every <to/> and <from/> element will contain with attribute.

## Querying by part of message body

This feature allows user to query for messages or collections containing messages which in body of a message contained text passed by user.

To execute request in which user wants to find messages with "test failed" string XMPP client needs to add following element

```
<query xmlns="http://tigase.org/protocol/archive#query">
  <contains>test failed</contains>
</query>
```

as child element of @retrieve@ or @list@ element of request.

## Example query requests

### Example 1

Retrieving messages with "test failed" string with user juliet@capulet.com between 2014-01-01 00:00:00 and 2014-05-01 00:00:00

```
<iq type="get" id="query2">
  <retrieve xmlns='urn:xmpp:archive'
    with='juliet@capulet.com'
    from='2014-01-01T00:00:00Z'
    end='2014-05-01T00:00:00Z'>
    <query xmlns="http://tigase.org/protocol/archive#query">
      <contains>test failed</contains>
    </query>
  </retrieve>
</iq>
```



## Example 2

Retrieving collections containing messages with "test failed" string with user juliet@capulet.com between 2014-01-01 00:00:00 and 2014-05-01 00:00:00

```
<iq type="get" id="query2">
  <list xmlns='urn:xmpp:archive'
    with='juliet@capulet.com'
    from='2014-01-01T00:00:00Z'
    end='2014-05-01T00:00:00Z'>
    <query xmlns="http://tigase.org/protocol/archive#query">
      <contains>test failed</contains>
    </query>
  </list>
</iq>
```

## Querying by tags

This feature adds support for tagging messages archived by Message Archiving component and by default is disabled (to learn how to enable this feature please see the section called “Enabling support for tags” section).

Tagging can be done only by user sending message as to tag message tag needs to be included in message content (message body to be exact).

Currently there are 2 types of tags supported:

- hashtag - words prefixed by "hash" (#) are stored with prefix and used as tag, i.e. #Tigase
- mention - words prefixed by "at" (@) are stored with prefix and used as tag, i.e. @Tigase

Custom feature allows user to query/retrieve messages or collections from archive only containing particular tag or tags. To execute request in which user wants to retrieve only messages tagged with @User1 and #People XMPP client executing request needs to add following element as child element of <retrieve/> element or <list/> element:

```
<query xmlns="http://tigase.org/protocol/archive#query">
  <tag>#People</tag>
  <tag>@User1</tag>
</query>
```

## Querying/retrieving list of messages or collections

### Example 1

Request to retrieve messages tagged with @User1 and #People from chat with user juliet@capulet.com between 2014-01-01 00:00:00 and 2014-05-01 00:00:00

```
<iq type="get" id="query2">
  <retrieve xmlns='urn:xmpp:archive'
    with='juliet@capulet.com'
    from='2014-01-01T00:00:00Z'
    end='2014-05-01T00:00:00Z'>
    <query xmlns="http://tigase.org/protocol/archive#query">
      <tag>#People</tag>
      <tag>@User1</tag>
    </query>
  </retrieve>
</iq>
```

```
        </query>
    </retrieve>
</iq>
```

## Example 2:

Request to retrieve collections containing messages tagged with @User1 and #People from chat with user juliet@capulet.com between 2014-01-01 00:00:00 and 2014-05-01 00:00:00

```
<iq type="get" id="query2">
  <list xmlns='urn:xmpp:archive'
    with='juliet@capulet.com'
    from='2014-01-01T00:00:00Z'
    end='2014-05-01T00:00:00Z'>
    <query xmlns="http://tigase.org/protocol/archive#query">
      <tag>#People</tag>
      <tag>@User1</tag>
    </query>
  </list>
</iq>
```

## Retrieving list of tags used by user starting with some text

To search for hashtags or user names already used following request might be used:

```
<iq type="set" id="query2">
  <tags xmlns="http://tigase.org/protocol/archive#query" like="#test"/>
</iq>
```

which will return suggested similar hashtags which were found in database for particular user if following response:

```
<iq type="result" id="query2">
  <tags xmlns="http://tigase.org/protocol/archive#query" like="#test">
    <tag>#test1</tag>
    <tag>#test123</tag>
    <set xmlns="http://jabber.org/protocol/rsm">
      <first index='0'>0</first>
      <last>1</last>
      <count>2</count>
    </set>
  </tags>
</iq>
```

## Automatic archiving of MUC messages

If this feature is enabled MUC messages are stored in Message Archiving repository and are added in same way as for any other messages and `jid` of MUC room is used as `jid` of message sender, so if MUC message sent from `test@muc.example.com` was stored then to retrieve this messages `test@muc.example.com` needs to be passed as `with` attribute to message retrieve request. Retrieved MUC messages will be retrieved in same format as normal message with one exception - each message will contain `name` attribute with name of occupant in room which sent this message.

This feature is by default disabled but it is possible to enable it for particular user. Additionally it is possible to change default setting on installation level and on hosted domain level to enable this feature, disable

feature or allow user to decide if user want this feature to be enabled. For more information about configuration of this feature look at the section called “Configuration of automatic archiving of MUC messages”

### Note

- It is worth to mention that even if more than one user resource joined same room and each resource will receive same messages then only single message will be stored in Message Archiving repository.
- It is also important to note that MUC messages are archived to user message archive only when user is joined to MUC room (so if message was sent to room but it was not sent to particular user)

## Database

### Preparation of database

Before you will be able to use Tigase Message Archiving Component and store messages in particular database you need to initialize this database. We provide few schemas for this component for MySQL, PostgreSQL, SQLServer and DerbyDB.

They are placed in `database/` directory of installation package and named in `dbtype-message-archiving-version.sql`, where `dbname` is name of database type which this schema supports and `version` is version of a Message Archiving Component for which this schema is designed.

You need to manually select schema for correct database and component and load this schema to database. For more information about loading database schema look into the section called “Database Preparation” section of ???

### Upgrade of database schema

Database schema for our components may change between versions and if so it needs to be updated before new version may be started. To upgrade schema please follow instructions from the section called “Preparation of database” section.

### Note

If you use SNAPSHOT builds then schema may change for same version as this are versions we are still working on.

## Schema description

Tigase Message Archiving component uses few tables and stored procedures. To make it easier to find them on database level they are prefixed with `tig_ma_`.

### Table `tig_ma_jids`

This table stores all jids related to stored messages, ie. from `to` and `from` attributes of archived stanzas.

Field	Description	Comments
<code>jid_id</code>	Database ID of a JID	
<code>jid</code>	Value of a bare JID	

Field	Description	Comments
jid_sha1	SHA1 value of lowercased bare JID	Used for proper bare JID comparison during lookup.  (N/A to PostgreSQL schema)
domain	Domain part of a bare JID	Stored for easier lookup of messages owned by users of a particular domain

## Table `tig_ma_msgs`

Table stores archived messages.

Field	Description	Comments
msg_id	Database ID of a message	
owner_id	ID of a bare JID of a message owner	References <code>jid_id</code> from <code>tig_ma_jids</code>
buddy_id	ID of a bare JID of a message recipient/sender (different than owner)	References <code>jid_id</code> from <code>tig_ma_jids</code>
buddy_res	Resource part of a message recipient/sender JID	
ts	Timestamp of a message	Timestamp of archivization or delayed delivery
direction	Direction of message	0 - sent by owner  1 - received by owner
type	Message type	Value of message <code>type</code> attribute
body	Body of a message	
msg	Serialized message	
stanza_hash	Hash of message parts	It is used to make sure that message is stored only once

## Table `tig_ma_tags`

Table stores tags of archived messages. It stores one tag for many messages using `tig_ma_msgs_tags` to store relation between tag and a message.

Field	Description	Comments
tag_id	Database ID of a tag	
owner_id	ID of a bare JID of a tag owner	ID of bare JID of owner for which messages with this tag were archived
tag	Actual tag value	

## Table `tig_ma_msgs_tags`

Table stores relations between tags and archived messages with this tags.

Field	Description	Comments
msg_id	Database ID of a message	References <code>msg_id</code> from <code>tig_ma_msgs</code>
tag_id	Database ID of a tag	References <code>tag_id</code> from <code>tig_ma_tags</code>

## Configuration

To enable Tigase Message Archiving Component you need to add following block to `etc/config.tdsl` file:

```
message-archive () {  
}
```

It will enable component and configure it under name `message-archive`. By default it will also use database configured as `default` data source to store data.

## Custom Database

You can specify a custom database to be used for message archiving. To do this, define the `archive-repo-uri` property.

```
'message-archive' () {  
  - 'archive-repo-uri' = - 'jdbc:mysql://localhost/messagearchivedb?user=test&pass=  
}
```

Here, `messagearchivedb` hosted on `localhost` is used.

## XEP-0136 Support

To be able to use Message Archiving component with XEP-0136: Message Archiving [<http://xmpp.org/extensions/xep-0136.html>] protocol, you additionally need to enable `message-archive-xep-0136` SessionManager processor:

```
sess-man {  
  message-archive-xep-0136 () {  
    -}  
}
```

This is required for some advanced options.

## Support for MAM

If you want to use Message Archiving component with XEP-0313: Message Archive Management [<http://xmpp.org/extensions/xep-0313.html>] protocol, then you need to enable `urn:xmpp:mam:1` SessionManager processor:

```
sess-man {  
  - 'urn:xmpp:mam:1' () {  
    -}  
}
```

## Setting default value of archiving level for message on a server

Setting this property will change default archiving level for messages for every account on server for which per account default archiving level is not set. User will be able to change this value setting default modes as described in XEP-0136 section 2.4 [<http://xmpp.org/extensions/xep-0136.html#pref-default>]

Possible values are:

false	Messages are not archived
body	Only message body will be stored. Message without a body will not be stored with this value set
message	While message stanza will be archived (if message should be stored, see the section called “Saving Options”)
stream	In this mode every stanza should be archived. ( <i>Not supported</i> )

To set default level to message you need to set `default-store-method` of `message-archive` processor to `message`:

```
sess-man {
    message-archive {
        default-store-method = -'message'
    }
}
```

## Setting required value of archiving level for messages on a server

Setting this property will change required archiving level for messages for every account on server. User will be able to change this to any lower value by setting default modes as described in XEP-0136 section 2.4 [<http://xmpp.org/extensions/xep-0136.html#pref-default>] but user will be allowed to set higher archiving level. If this property is set to higher value then default archiving level is set then this setting will be used as default archiving level setting.

Possible values for this setting are the same as values for default archiving level setting, see the section called “Setting default value of archiving level for message on a server” for list of possible values.

To set required level to body you need to set `required-store-method` of `message-archive` processor to `body`:

```
sess-man {
    message-archive {
        required-store-method = `body`
    }
}
```

## Enabling support for tags

To enable this feature Message Archiving component needs to be configured properly. You need to add `tags-support = true` line to `message-archiving` configuration section of `etc/config.tdsl` file. Like in following example:

```
message-archive {
    tags-support = true
}
```

where:

- `message-archive` - is name of configuration section under which Message Archiving component is configured to run

## Saving Options

By default, Tigase Message Archive will only store the message body with some metadata, this can exclude messages that are lacking a body. If you decide you wish to save non-body elements within Message Archive, you can now configure this by setting `msg-archive-paths` to list of elements paths which should trigger saving to Message Archive. To additionally store messages with `<subject/>` element:

```
sess-man {
  message-archive {
    msg-archive-paths = [ -'/message/result[urn:xmpp:mam:1]' -'/message/body'
    -}
  }
}
```

Where above will set the archive to store messages with `<body/>` or `<subject/>` elements and for message with `<result xmlns="urn:xmpp:mam:1"/>` element not to be stored.

### Warning

It is recommended to keep entry for not storing message with `<result xmlns="urn:xmpp:mam:1"/>` element as this are results of MAM query and contain messages already stored in archive!

### Tip

Enabling this for elements such as `iq`, or `presence` will quickly load the archive. Configure this setting carefully!

## Configuration of automatic archiving of MUC messages

As mentioned above no additional configuration options than default configuration of Message Archiving component and plugin is needed to let user decide if he wants to enable or disable this feature (but it is disabled by default). In this case user to enable this feature needs to set settings of message archiving adding `muc-save` attribute to `<default/>` element of request with value set to `true` (or to `false` to disable this feature).

To configure state of this feature on installation level, it is required to set `store-muc-messages` property of `message-archive` SessionManager processor:

```
sess-man {
  message-archive {
    store-muc-messages = -'value'
    -}
  }
}
```

where `value` may be one of following values:

- |                    |  |
|--------------------|--|
| <code>user</code>  | allows value to be set on domain level or by user if domain level setting allows that                    |
| <code>true</code>  | enables feature for every user in every hosted domain (cannot be overridden by on domain or user level)  |
| <code>false</code> | disables feature for every user in every hosted domain (cannot be overridden by on domain or user level) |

To configure state of this feature on domain level, you need to execute vhost configuration command. In list of fields to configure domain, field to set this will be available with following values:

`user` allows user to stat of this feature (if allowed on installation level)

`true` enables feature for users of configured domain (user will not be able to disable)

`false` disables feature for users of configured domain (user will not be able to disable)

## Purging Information from Message Archive

This feature allows for automatic removal of entries older than a configured number of days from the Message Archive. It is designed to clean up database and keep its size within reasonable boundaries. If it is set to 1 day and entry is older than 24 hours then it will be removed, i.e. entry from yesterday from 10:11 will be removed after 10:11 after next execution of purge.

There are 3 settings available for this feature: To enable the feature:

```
'message-archive' {  
    -'remove-expired-messages' = true  
}
```

This setting changes the initial delay after the server is started to begin removing old entries. In other words, MA purging will not take place until the specified time after the server starts. Default setting is PT1H, or one hour.

```
-'remove-expired-messages-delay' = -'PT2H'
```

This setting sets how long MA purging will wait between passes to check for and remove old entries. Default setting is P1D which is once a day.

```
-'remove-expired-messages-period' = -'PT2D'
```

You may use all settings at once if you wish.

**NOTE** that these commands are also compatible with `unified-archive` component, just replace `message` with `unified`.

## Configuration of number of days in VHost

VHost holds a setting that determines how long a message needs to be in archive for it to be considered old and removed. This can be set independently per Vhost. This setting can be modified by either using the HTTP admin, or the update item execution in adhoc command.

This configuration is done by execution of Update item configuration adhoc command of vhost-man component, where you should select domain for which messages should be removed and then in field XEP-0136 - retention type select value Number of days and in field XEP-0136 - retention period (in days) enter number of days after which events should be removed from MA.

In adhoc select domain for which messages should be removed and then in field XEP-0136 - retention type select value Number of days and in field XEP-0136 - retention period (in days) enter number of days after which events should be removed from MA.

In HTTP UI select Other, then Update Item Configuration (Vhost-man), select the domain, and from there you can set XEP-0136 retention type, and set number of days at XEP-0136 retention period (in days).



## Using separate store for archived messages

It is possible to use separate store for archived messages, to do so you need to configure new `DataSource` in `dataSource` section. Here we will use `message-archive-store` as a name of a data source. Additionally you need to pass name of newly configured data source to `dataSourceName` property of default repository of Message Archiving component.

Example:

```
dataSource {
    message-archive-store () {
        uri = -'jdbc:postgresql://server/message-archive-database'
    -}
}

message-archive {
    repositoryPool {
        default () {
            dataSourceName = -'message-archive-store'
        -}
    -}
}
```

It is also possible to configure separate store for particular domain, i.e. `example.com`. Here we will configure data source with name `example.com` and use it to store data for archive:

```
dataSource {
    -'example.com' () {
        uri = -'jdbc:postgresql://server/example-database'
    -}
}

message-archive {
    repositoryPool {
        -'example.com' () {
            # we may not set dataSourceName as it matches name of domain
        -}
    -}
}
```

### Note

With this configuration messages for other domains than `example.com` will be stored in default data source.

## Setting Pool Sizes

There are a high number of prepared statements which are used to process and archive messages as they go through the server, and you may experience an increase in resource use with the archive turned on. It is recommended to decrease the repository connection pool to help balance server load from this component using the `Pool Size` property:

```
'message-archive' (class: tigase.archive.MessageArchiveComponent) {
    -'archive-repo-uri' = -'jdbc:mysql://localhost/messagearchivedb?user=test&pass'
```

```
    - 'pool-size' = 15  
  }
```

## Message Tagging Support

Tigase now is able to support querying message archives based on tags created for the query. Currently, Tigase can support the following tags to help search through message archives: - `hashtag` Words prefixed by a hash (#) are stored with a prefix and used as a tag, for example `#Tigase` - `mention` Words prefixed by an at (@) are stored with a prefix and used as a tag, for example `@Tigase`

**NOTE:** Tags must be written in messages from users, they do not act as wildcards. To search for `#Tigase`, a message must have `#Tigase` in the `<body>` element.

This feature allows users to query and retrieve messages or collections from the archive that only contain one or more tags.

## Activating Tagging

To enable this feature, the following line must be in the `config.tdsl` file (or may be added with Admin or Web UI)

```
'message-archive' (class: tigase.archive.MessageArchiveComponent) {  
    - 'tags-support' = true  
}
```

## Usage

To execute a request, the tags must be individual children elements of the `retrieve` or `list` element like the following request:

```
<query xmlns="http://tigase.org/protocol/archive#query">  
    <tag>#People</tag>  
    <tag>@User1</tag>  
</query>
```

You may also specify specific senders, and limit the time and date that you wish to search through to keep the resulting list smaller. That can be accomplished by adding more fields to the `retrieve` element such as `'with'`, `'from'`, and `'end'`. Take a look at the below example:

```
<iq type="get" id="query2">  
    <retrieve xmlns='urn:xmpp:archive'  
        with='juliet@capulet.com'  
        from='2014-01-01T00:00:00Z'  
        end='2014-05-01T00:00:00Z'  
        <query xmlns="http://tigase.org/protocol/archive#query">  
            <tag>#People</tag>  
            <tag>@User1</tag>  
        </query>  
    </retrieve>  
</iq>
```

This stanza is requesting to retrieve messages tagged with `@User1` and `#people` from chats with the user `juliet@capulet.com` [mailto:juliet@capulet.com] between January 1st, 2014 at 00:00 to May 1st, 2014 at 00:00.

**NOTE:** All times are in Zulu or GMT on a 24h clock.

You can add as many tags as you wish, but each one is an **AND** statement; so the more tags you include, the smaller the results.

## Usage

Now that we have the archive component running, how do we use it? Currently, the only way to activate and modify the component is through XMPP stanzas. Lets first begin by getting our default settings from the component:

```
<iq type='get' id='prefq'>
  <pref xmlns='urn:xmpp:archive' />
</iq>
```

It's a short stanza, but it will tell us what we need to know, Note that you do not need a from or a to for this stanza. The result is as follows:

```
<iq type='result' id='prefq' to='admin@domain.com/cpu'>
  <pref xmlns='urn:xmpp:archive'>
    <auto save='false' />
    <default otr='forbid' muc-save="false" save="body" />
    <method use="prefer" type="auto" />
    <method use="prefer" type="local" />
    <method use="prefer" type="manual" />
  </pref>
</iq>
```

See below for what these settings mean.

## XEP-0136 Field Values

<auto/>	<ul style="list-style-type: none"><li>• <b>Required Attributes</b><ul style="list-style-type: none"><li>• save= Boolean turning archiving on or off</li></ul></li><li>• <b>Optional Settings</b><ul style="list-style-type: none"><li>• scope= Determines scope of archiving, default is 'stream' which turns off after stream end, or may be 'global' which keeps auto save permanent,</li></ul></li></ul>
<default/>	<p>Default element sets default settings for OTR and save modes, includes an option for archive expiration.</p> <ul style="list-style-type: none"><li>• <b>Required Attributes</b><ul style="list-style-type: none"><li>• otr= Specifies setting for Off The Record mode. Available settings are:<ul style="list-style-type: none"><li>• approve The user <b>MUST</b> explicitly approve OTR communication.</li><li>• concede Communications <b>MAY</b> be OTR if requested by another user.</li><li>• forbid Communications <b>MUST NOT</b> be OTR.</li><li>• oppose Communications <b>SHOULD NOT</b> be OTR.</li></ul></li></ul></li></ul>

- `prefer` Communications SHOULD be OTR.
- `require` Communications MUST be OTR.
- `save=` Specifies the portion of messages to archive, by default it is set to `body`.
  - `body` Archives only the items within the `<body/>` elements.
  - `message` Archive the entire XML content of each message.
  - `stream` Archive saves every byte of communication between server and client. (Not recommended, high resource use)

- **Optional Settings**

- `expire=` Specifies after how many seconds should the server delete saved messages.

`<item/>`

The Item element specifies settings for a particular entity. These settings will override default settings for the specified JIDS.

- **Required Attributes**

- `JID=` The Jabber ID of the entity that you wish to put these settings on, it may be a full JID, bare JID, or just a domain.
- `otr=` Specifies setting for Off The Record mode. Available settings are:
  - `approve` The user MUST explicitly approve OTR communication.
  - `concede` Communications MAY be OTR if requested by another user.
  - `forbid` Communications MUST NOT be OTR.
  - `oppose` Communications SHOULD NOT be OTR.
  - `prefer` Communications SHOULD be OTR.
  - `require` Communications MUST be OTR.
- `save=` Specifies the portion of messages to archive, by default it is set to `body`.
  - `body` Archives only the items within the `<body/>` elements.
  - `message` Archive the entire XML content of each message.
  - `stream` Archive saves every byte of communication between server and client. (Not recommended, high resource use)

- **Optional Settings**

- `expire=` Specifies after how many seconds should the server delete saved messages.

`<method/>`

This element specifies the user preference for available archiving methods.

- `type=` The type of archiving to set
  - `auto` Preferences for use of automatic archiving on the user's server.
  - `local` Set to use local archiving on user's machine or device.
  - `manual` Preferences for use of manual archiving to the server.
- `use=` Sets level of use for the type
  - `prefer` The selected method should be used if it is available.
  - `concede` This will be used if no other methods are available.
  - `forbid` The associated method MUST not be used.

Now that we have established settings, lets send a stanza changing a few of them:

```
<iq type='set' id='pref2'>
  <pref xmlns='urn:xmpp:archive'>
    <auto save='true' scope='global' />
    <item jid='domain.com' otr='forbid' save='body' />
    <method type='auto' use='prefer' />
    <method type='local' use='forbid' />
    <method type='manual' use='concede' />
  </pref>
</iq>
```

This now sets archiving by default for all users on the domain.com server, forbids OTR, and prefers auto save method for archiving.

## Manual Activation

Turning on archiving requires a simple stanza which will turn on archiving for the use sending the stanza and using default settings.

```
<iq type='set' id='turnon'>
  <pref xmlns='urn:xmpp:archive'>
    <auto save='true' />
  </pref>
</iq>
```

A successful result will yield this response from the server:

```
<iq type='result' to='user@domain.com' id='turnon' />
```

Once this is turned on, incoming and outgoing messages from the user will be stored in `tig_ma_msgs` table in the database.

## Limitations

- Component groups messages in collections using date of a messages instead of id of message thread, due to fact that some clients are sending messages with no thread id (ie. Psi, Psi+).
- Only bare JID is stored of sender or recipient.

---

# Chapter 18. Tigase PubSub Component

Welcome to Tigase PubSub component guide

## PubSub Component

Tigase's Publish Subscribe component is an XEP-0060 [<http://www.xmpp.org/extensions/xep-0060.html>] compliant plugin handling all publish and subscribe activity within Tigase server. This is enabled as default with the pubsub name, however you may include the following line if you wish to customize it's configuration.

```
pubsub ( ) { }
```

You may change the name so long as you specify the pubsub class within parenthesis.

## Announcement

### Major changes

Tigase pubsub component has undergone a few major changes to our code and structure. To continue to use Tigase pubsub component, a few changes may be needed to be made to your systems. Please see them below:

### Database schema changes

Current version comes with changes to database schema to improve JID comparison during lookup of nodes, subscriptions, affiliations, etc.

To continue usage of new versions of pubsub component it is required to manually load new component database schema, see database preparation section for more information.

### Warning

Loading of new database schema is required to use new version of pubsub component.

### Changes in REST API

We continuously work on improving usability and making our REST API easier to use we added support for handling JSON requests in REST API for pubsub. At the same time we decided to slightly modify responses in XML sent by REST API to make responses in JSON and XML similar.

For more informations about current REST API please look into Rest API section.

### New features

### Support for using separate database for different domains

Since this version it is possible to use separate pubsub nodes and items based on domains. This allows you to configure component to store informations about nodes and items for particular domain to different database.

For more informations please look into using multiple databases.

## Support for MAM

In this version we added support for XEP-0313: Message Archive Management [<http://xmpp.org/extensions/xep-0313.html>:] protocol which allows any MAM compatible XMPP client with pubsub support to retrieve items published on pubsub nodes using MAM protocol for querying.

## Configuration

### Pubsub naming

Within Tigase, all pubsub component address **MUST** be domain-based address and not a JID style address. This was made to simplify communications structure. Tigase will automatically set component names to pubsub.domain, however any messages send to pubsub@domain will result in a `SERVICE_UNAVAILABLE` error.

Pubsub nodes within Tigase can be found as a combination of JID and node where nodes will be identified akin to service discovery. For example, to address a friendly node, use the following structure:

```
<iq to='pubsub.domain'>
  <query node='friendly node' />
</iq>
```

### Configure Roster Maximum size

Administrators can configure the maximum allowable roster size per user via the config.tdsll file.

```
'sess-man' {
  -'jabber:ica:roster' {
    max_roster_size = -'100'
  }
}
```

This sets the roster limit to 100 entries per user. It can be set to any integer, however by default no limit is set and no configuration is set in config.tdsl file.

### Store Full XML of Last Presence

Tigase can store a more detailed `<unavailable/>` presence stanza to include timestamps and other information.

### Requirements

Ensure that `presence-offline` plugin is enabled in config.tdsl. To do this, add be sure `presence-offline` is listed under `sess-man`

```
'sess-man' {
  -'presence-offline' () {}
}
```

The following two lines in `sess-man` configure options to broadcast probes to offline users.

```
'sess-man' {
  -'skip-offline' = -'false'
```

```
- 'skip-offline-sys' = - 'false'
}
```

Without these lines, Tigase will not send presence probes to users that the server knows to be offline.

The full XML presence is stored under the `tig_pairs` table with a pkey of `last-unavailable-presence` will look like this:

```
<presence from="user@example.com" xmlns="jabber:client" type="unavailable">
<status>Logged out</status>
<delay stamp="2015-12-29T16:51:50.748Z" xmlns="urn:xmpp:delay"/></presence>
```

As you can see, the plugin has added a delay stamp which indicates the last time they were seen online. This may be suppressed by using the following line in your `config.tdsl` file.

```
'sess-man' {
  - 'delay-stamp' = - 'false'
}
```

You may also limit probe responses only to newly connected resources.

```
'sess-man' {
  - 'probe-full-jid' = - 'true'
}
```

When a user logs on, they will receive the same full unavailable presence statements from contacts not logged in. Also the repository entry containing their last unavailable presence will be removed.

**NOTE: This will increase traffic with users with many people on their rosters.**

## Using separate store

As mentioned above, by default Tigase pubsub component uses default data source configured for Tigase XMPP Server. It is possible to use separate store by pubsub component. To do so you need to configure new `DataSource` in `dataSource` section. Here we will use `pubsub-store` as name of newly configured data source. Additionally you need to pass name of newly configured data source to `dataSourceName` property of default DAO of pubsub component.

```
dataSource {
  pubsub-store () {
    uri = - 'jdbc:postgresql://server/pubsub-database'
  }
}

pubsub () {
  dao {
    default () {
      dataSourceName = - 'pubsub-store'
    }
  }
}
```

It is also possible to configure separate store for particular domain, ie. `pubsub.example.com`. Here we will configure data source with name `pubsub.example.com` and use it to store data for pubsub nodes and items at `pubsub.example.com`:



```
dataSource {
    -'pubsub.example.com' () {
        uri = -'jdbc:postgresql://server/example-database'
    -}
}

pubsub () {
    dao {
        -'pubsub.example.com' () {
            # we may not set dataSourceName as it matches name of domain
        -}
    -}
}
```

### Note

With this configuration, data for other domains than `pubsub.example.com` will be stored in default data source.

## Enabling PEP support

To enable XEP-0163: Personal Eventing Protocol [<http://xmpp.org/extensions/xep-0163.html>] support it is required to set `persistent-pep` property of `pubsub` component to `true`, set `send-last-published-item-on-presence` property of component to `true` and enable `pep` `SessionManager` processor.

```
pubsub () {
    persistent-pep = true
    send-last-published-item-on-presence = true
}

sess-man () {
    pep () {
    -}
}
```

### Note

If your `pubsub` component uses different name than `pubsub` then you need to set `pubsub-jid` property of `pep` processor to JID of `pubsub` component make it aware of a different name of a `pubsub` component.

**Example with `pubsub` component named `events` hosted at server named `servername.com` and enabled PEP.**

```
events () {
    persistent-pep = true
    send-last-published-item-on-presence = true
}

sess-man () {
    pep () {
        -'pubsub-jid' = -'events@servername.com'
    -}
}
```

## Enabling REST API

To use REST API for pubsub component it is required that:

- Tigase HTTP API component is installed and configured properly. For information about HTTP API component installation please look into HTTP component documentation.
- Tigase pubsub REST scripts are copied to HTTP API REST scripts directory In installation package this is already done and scripts are in proper locations. dd\* JID of HTTP API component needs to be added to list of trusted jids of Tigase pubsub component trusted property (if http is name of HTTP API component)

```
pubsub () {  
    trusted = [ -'http@{clusterNode}' -];  
}
```

## Changing nodes cache size

By default Tigase pubsub component caches node configuration of 2000 last loaded nodes. If there are many requests to database to load node configuration and your installation contains many nodes it may be a good idea to increase number of cached nodes.

To do this you need to set `pubsub-repository-cache-size` property of pubsub component to new size.

```
pubsub () {  
    pubsub-repository-cache-size = 4000  
}
```

## Enable sending last published item on presence

By default it is not possible to use delivery of last published item when users broadcasts initial presence. To do so you need to set `send-last-published-item-on-presence` of pubsub component to `true`. This will allow you to configure nodes to send last published item on presence.

```
pubsub () {  
    send-last-published-item-on-presence = true  
}
```

## Tune handling of low memory

If there is less than 10% of free heap memory available during publication of item then Tigase pubsub component will trigger Garbage Collection and if there is still very little amount of free memory will slow down delivery of notifications for published items (waiting about 1 second before continuing).

If you have assigned a lot of memory to Tigase XMPP Server or in your case this delay is not acceptable you can adjust it by pubsub component properties:

- setting `pubsub-high-memory-usage-level` to percentage of heap memory accepted as near OOM state
- setting `pubsub-low-memory-delay` to number of milliseconds to wait to throttle delivery of notifications

```
pubsub () {
```

```
pubsub-high-memory-usage-level = 95
pubsub-low-memory-delay = 100
}
```

## Disable automatic subscription of node creator

During creation of node pubsub component subscribes creator to pubsub node and delivers notifications to creator. If in your case you do not want this behavior, you may set `auto-subscribe-node-creator` property of pubsub component to `false`.

```
pubsub () {
    auto-subscribe-node-creator = false
}
```

## Database

### Preparation of database

Before you will be able to use Tigase PubSub Component you need to initialize database. We provide few schemas for this component for MySQL, PostgreSQL, SQLServer and DerbyDB.

They are placed in `database/` directory of installation package and named in `dbtype-pubsub-version.sql`, where `dbname` in name of database type which this schema supports and `version` is version of a PubSub component for which this schema is designed.

You need to manually select schema for correct database and component and load this schema to database. For more information about loading database schema look into database preparation section of this guide.

### Upgrade of database schema

Database schema for our components may change between versions and if so it needs to be updated before new version may be started. To upgrade schema please follow instructions from the database preparation section.

#### Note

If you use SNAPSHOT builds then schema may change for same version as this are versions we are still working on.

### Schema description

Tigase PubSub component uses few tables and stored procedures. To make it easier to identify tables and stored procedures used by PubSub component they are prefixed with `tig_pubsub_`.

#### Table `tig_pubsub_service_jids`

This table stores all jids for which PubSub component contains nodes.

Field	Description	Comments
<code>service_id</code>	Database ID of a service JID	
<code>service_jid</code>	Value of a service JID	
<code>service_jid_sha</code>	SHA1 value of lowercased service JID	Used for proper bare JID comparison during lookup.

Field	Description	Comments
		(N/A to PostgreSQL schema)

### Table `tig_pubsub_jids`

This table stores all jids related to PubSub nodes, ie. subscriber, affiliates, creators, publishers, etc.

Field	Description	Comments
<code>jid_id</code>	Database ID of a bare JID	
<code>jid</code>	Value of a bare JID	
<code>jid_sha1</code>	SHA1 value of lowercased bare JID	Used for proper bare JID comparison during lookup.  (N/A to PostgreSQL schema)

### Table `tig_pubsub_nodes`

Table stores nodes tree structure and node configuration.

Field	Description	Comments
<code>node_id</code>	Database ID of a node	
<code>service_id</code>	ID of service JID	References <code>service_id</code> from <code>tig_pubsub_service_jids</code>
<code>name</code>	Name of PubSub node	
<code>name_sha1</code>	SHA1 of PubSub node name	Used for indexing and faster lookup.  (N/A to PostgreSQL schema)
<code>type</code>	Type of PubSub node	0 - collection  1 - leaf
<code>title</code>	Title of PubSub node	
<code>description</code>	Description of a node	
<code>creator_id</code>	ID of JID of creator	References <code>jid_id</code> from <code>tig_pubsub_jids</code>
<code>creation_date</code>	Timestamp of creation time	
<code>configuration</code>	Serialized configuration of PubSub node	
<code>collection_id</code>	Points collection (parent node)	References <code>node_id</code> from <code>tig_pubsub_nodes</code>

### Table `tig_pubsub_affiliations`

Table stores affiliations between PubSub nodes and jids.

Field	Description	Comments
<code>node_id</code>	ID of a node	References <code>node_id</code> from <code>tig_pubsub_nodes</code>
<code>jid_id</code>	ID of a user JID	References <code>jid_id</code> from <code>tig_pubsub_jids</code>

Field	Description	Comments
affiliation	Affiliation value	

### Table `tig_pubsub_subscriptions`

Table stores subscriptions of jids to PubSub nodes.

Field	Description	Comments
node_id	ID of a node	References node_id from <code>tig_pubsub_nodes</code>
jid_id	ID of a user JID	References jid_id from <code>tig_pubsub_jids</code>
subscription	Subscription value	
subscription_id	Id of a subscription	

### Table `tig_pubsub_items`

Table stores items of PubSub nodes.

Field	Description	Comments
node_id	ID of a node	References node_id from <code>tig_pubsub_nodes</code>
id	Id of an items	
id_sha1	SHA1 of item id	Indexed and used for faster lookup (N/A to PostgreSQL schema)
creation_date	Creation date	
publisher_id	ID of publisher JID	References jid_id from <code>tig_pubsub_jids</code>
update_date	Timestamp of last item modification	
data	Item payload	

## PubSub Schema Changes

Tigase PubSub Component is currently version 3.3.0 which is introduced in Tigase server v8.0.0.

### PubSub 3.2.0 Changes

PubSub v 3.2.0 adds a new procedure `TigPubSubGetNodeMeta` which supports PubSub metadata retrieval while conducting a `disco#info` query on nodes.

You will need to upgrade your database if you are not using v3.2.0 schema. Tigase will report being unable to load PubSub component if you do not have this schema version.

The MySQL schema can be found Here [<https://projects.tigase.org/projects/tigase-pubsub/repository/revisions/master/entry/database/mysql-pubsub-schema-3.2.0.sql>].

The Derby schema can be found Here [<https://projects.tigase.org/projects/tigase-pubsub/repository/changes/database/derby-pubsub-schema-3.2.0.sql>].

The PostgreSQL schema can be found Here [<https://projects.tigase.org/projects/tigase-pubsub/repository/changes/database/postgresql-pubsub-schema-3.2.0.sql>].

The MS SQL schema can be found Here [<https://projects.tigase.org/projects/tigase-pubsub/repository/changes/database/sqlserver-pubsub-schema-3.2.0.sql>].

The same files are also included in all distributions of v8.0.0 in [tigaseroot]/database/. All changes to database schema are meant to be backward compatible.

For instructions how to manually upgrade the databases, please refer to Tigase v7.1.0 Schema Updates section.

## Upgrading older installations (pre-v3.0.0 Schema)

To update older installations of Tigase to the PubSub Schema v3.0.0 follow these instructions. Note this should be done before upgrading to PubSub v3.1.0.

Step by Step guide.

### Prepare Old Database for Upgrade

In database directory of Tigase installation you will find SQL files which will prepare old database schema for upgrade using following this naming pattern: <database\_type>-pubsub-schema-3.0.0-pre-upgrade.sql Where <database\_type> can be one of the following: mysql, sqlserver, ie. for MySQL you will find the file mysql-pubsub-schema-3.0.0-pre-upgrade.sql. You need to execute statements from this file on your source database, which will drop old procedures and functions used to access database and also this statements will rename old tables by adding suffix \_1 to each of old tables. Example:

```
MySQL  mysql -u tigase -p tigase_pubsub < database/mysql-pubsub-schema-3.0.0-pre-upgrade.sql
```

```
MS SQL sqlcmd -S %servername% -U %root_user% -P %root_pass% -d %database% -i database\sqlserver-pubsub-schema-3.0.0-pre-upgrade.sql
```

### Update Tigase PubSub Component

For this you need to copy the Tigase PubSub Component jar file to jars directory inside Tigase XMPP Server installation directory. It is also recommended to copy files from database directory of Tigase PubSub Component to database directory in Tigase XMPP Server installation directory.

If you happen to use one of the the distribution packaged (either installer or -dist-max flavored archive) then all required files are already available - both new schema files will be available in database/ directory as well as both versions of PubSub component will be present in jars/ directory - PubSub3 as tigase-pubsub.jar and PubSub2 as tigase-pubsub-2.2.0.jar.old (provided for compatibility reasons).

### Load New Schema

In the database directory you will find files containing new schemas for:

- MySQL - mysql-pubsub-schema-3.0.0.sql
- PostgreSQL - postgresql-pubsub-schema-3.0.0.sql

- MSSQL - `sqlserver-pubsub-schema-3.0.0.sql`
- DerbyDB - `derby-pubsub-schema-3.0.0.sql` and `pubsub-db-create-derby.sh`

For most databases, with the exception of Derby, you only need to execute statements from the proper file. For example:

MySQL        `mysql -u tigase -p tigase_pubsub < database/mysql-pubsub-schema-3.0.0.sql`

MS SQL       `sqlcmd -S %servername% -U %root_user% -P %root_pass% -d %database% -i database\sqlserver-pubsub-schema-3.0.0.sql`

PostgreSQL   `psql -h $DB_HOST -q -U ${USR_NAME} -d $DB_NAME -f database/sqlserver-pubsub-schema-3.0.0.sql`

For DerbyDB you need to execute the `pubsub-db-create-derby.sh` script and pass proper JDBC URI to database to which you want to load schema (if database does not exist, it will be created).

`database/pubsub-db-create-derby.sh`

**NOTE:** It is possible to use same database which was used before - then after upgrade you will have new tables and old tables with `_1` suffix.

## Execute Migration Utility

In the `/database` directory you will find the `pubsub-db-migrate.sh` file which you need to execute and pass arguments with JDBC URIs needed to connect to source and destination database. If you used dedicated tables for PubSub you will also need to pass a class name used to access database (value of `pubsub-repo-class` variable from `etc/config.tdsl` file).

Example for dedicated table used for PubSub:

```
database/pubsub-db-migrate.sh --in-repo-class tigase.pubsub.repository.PubSubDAO
-in -'jdbc:mysql://localhost/tigase_pubsub?user=tigase&password=passwd'
-out -'jdbc:mysql://localhost/tigase_pubsub?user=tigase&password=passwd'
```

Example for use without dedicated PubSub tables:

```
database/pubsub-db-migrate.sh
-in -'jdbc:mysql://localhost/tigase?user=tigase&password=passwd'
-out -'jdbc:mysql://localhost/tigase?user=tigase&password=passwd'
```

Example for use with dedicated tables in a Windows environment:

```
database/pubsub-db-migrate.cmd --in-repo-class tigase.pubsub.repository.PubSubDAO
-in -'jdbc:sqlserver://<hostname>\\<instance>:<port>;databaseName=<name>;user=tiga
-out -'jdbc:sqlserver://<hostname>\\<instance>:<port>;databaseName=<name>;user=tig
```

During execution this utility will report information about migration of PubSub data to the new schema, and the same information will be store in `pubsub_db_migration.log`.

## Finish

After successful migration you will have all data copied to new tables. Old tables will be renamed by adding suffix `_1`. After verification that everything works OK, you can delete old tables and it's content as it want be used any more.

## Features

### AdHoc Commands

Similar to the HTTP API, AdHoc commands based on groovy scripts can be sent to this component to do a number of tasks. All scripts for these Ad-hoc commands are found at `sec/main/groovy/tigase/admin` in source distributions, or at this link [<https://projects.tigase.org/projects/tigase-pubsub/repository/revisions/master/show/src/main/groovy/tigase/admin>]. To use them, the scripts need to be copied into the `scripts/admin/pubsub` folder in the Tigase installation directory. For all examples, the component address will be `pubsub.example.com`.

#### Create a Node

Ad-hoc command node: `create-node` Required role: Service Administrator

Command requires fields `node` and `pubsub#node_type` to be filled with proper values for execution. - `node` Field containing id of node to create. - `pubsub#node_type` Contains one of two possible values. \* `leaf-node` Node that will be published. \* `collection` Node that will contain other nodes.

Other fields are optional fields that can be set to change configuration of newly create node to different configuration than default.

Example call using TCLMT:

```
bin/tclmt.sh --u admin@example.com --p admin123 remote pubsub.example.com create-n
```

#### Delete a Node

Ad-hoc command node: `delete-node` Required role: Service Administrator

Command requires `node` field to be filled. - `node` Field containing id of node to delete.

Example call using TCLMT:

```
bin/tclmt.sh --u admin@example.com --p admin123 remote pubsub.example.com delete-n
```

#### Subscribe to a Node

Ad-hoc command node: `subscribe-node` Required role: Service Administrator

Command requires `node` and `jids` nodes to be filled. - `node` Field containing node to subscribe to. - `jids` Field containing list of JIDs to subscribe to the node.

Example call using TCLMT:

```
bin/tclmt.sh --u admin@example.com --p admin123 remote pubsub.example.com subscrib
```

#### Unsubscribe to a Node

Ad-hoc command node: `unsubscribe-node` Required role: Service Administrator

Command requires `node` and `jids` nodes to be filled. - `node` Field containing node to unsubscribe to. - `jids` Field containing list of JIDs to unsubscribe to the node.



Example call using TCLMT:

```
bin/tclmt.sh --u admin@example.com --p admin123 remote pubsub.example.com unsubscribe
```

## Publish an item to a Node

Ad-hoc command node: `publish-item` Required role: Service Administrator

Command requires fields `node` and `entry` to be filled. - `node` Field containing id of node to publish to. - `item-id` Field may contain id of entry to publish, can be empty. - `entry` Field should contain multi-line entry content that should be valid XML values for items.

This command due to it's complexity cannot be easily executed by TCLMT using default remote script which provides support for basic adhoc commands. Example call using TCLMT:

```
bin/tclmt.sh --u admin@example.com --p admin123 remote pubsub.example.com publish-
```

Example Groovy script to execute create-node command using JAXMPP2

```
import tigase.jaxmpp.j2se.Jaxmpp
import tigase.jaxmpp.core.client.AsyncCallback
import tigase.jaxmpp.core.client.exceptions.JaxmppException
import tigase.jaxmpp.core.client.xmlpp.stanzas.Stanza
import tigase.jaxmpp.core.client.SessionObject
import tigase.jaxmpp.j2se.ConnectionConfiguration
import tigase.jaxmpp.core.client.xml.Element
import tigase.jaxmpp.core.client.xml.DefaultElement
import tigase.jaxmpp.core.client.xmlpp.forms.JabberDataElement

Jaxmpp jaxmpp = new Jaxmpp();

jaxmpp.with {
    getConnectionConfiguration().setConnectionType(ConnectionConfiguration.CONNECT_REMOTE);
    getConnectionConfiguration().setUserJID("admin@example.com");
    getConnectionConfiguration().setUserPassword("admin123");
}

jaxmpp.login(true);

def packet = IQ.create();
packet.setAttribute("to", "-pubsub.example.com");

Element command = new DefaultElement("command");
command.setXMLNS("http://jabber.org/protocol/commands");
command.setAttribute("node", "-create-node");
packet.addChild(command);

Element x = new DefaultElement("x");
x.setXMLNS("jabber:x:data");

command.addChild(x);

def data = new JabberDataElement(x);
data.addTextSingleField("node", "-example");
data.addListSingleField("pubsub#node_type", "-leaf");
```

```
jaxmpp.send(packet, new AsyncCallback() {
    void onError(Stanza responseStanza, tigase.jaxmpp.core.client.XMPPEException.Er
        println -"received error during processing request";
    -}

    void onSuccess(Stanza responseStanza) throws JaxmppException {
        x = responseStanza.getFirstChild("command").getFirstChid("x");
        data = new JabberDataElement(x);
        def error = data.getField("Error");
        println -"command executed with result = -" + (error -? -"failure, error =
    -}

    void onTimeout() {
        println -"command timed out"
    -}
});

Thread.sleep(30000);
jaxmpp.disconnect();
```

## PubSub Node Presence Protocol

### Occupant Use Case

### Log in to Pubsub Node

To log in to PubSub Node user must send presence to PubSub component with additional information about node:

```
<presence
  from='hag66@shakespeare.lit/pda'
  id='n13mt31'
  to='pubsub.shakespeare.lit'>
  <pubsub xmlns='tigase:pubsub:1' node='princely_musings' />
</presence>
```

Component will publish this information in node:

```
<message from='pubsub.shakespeare.lit' to='francisco@denmark.lit' id='foo'>
  <event xmlns='http://jabber.org/protocol/pubsub#event'>
    <items node='princely_musings'>
      <item>
        <presence xmlns='tigase:pubsub:1' node='princely_musings' jid='hag66@shake
      </item>
    </items>
  </event>
</message>
<message from='pubsub.shakespeare.lit' to='bernardo@denmark.lit' id='bar'>
  <event xmlns='http://jabber.org/protocol/pubsub#event'>
    <items node='princely_musings'>
      <item>
        <presence xmlns='tigase:pubsub:1' node='princely_musings' jid='hag66@shake
      </item>
```

```
</items>
</event>
</message>
```

And then will send notification with presences of all occupants to new occupant.

## Log out from PubSub Node

To logout from single node, user must send presence stanza with type unavailable:

```
<presence
  from='hag66@shakespeare.lit/pda'
  type='unavailable'
  to='pubsub.shakespeare.lit'>
  <pubsub xmlns='tigase:pubsub:1' node='princely_musings' />
</presence>
```

Component will send events to all occupants as described:

```
<message from='pubsub.shakespeare.lit' to='francisco@denmark.lit' id='foo'>
  <event xmlns='http://jabber.org/protocol/pubsub#event'>
    <items node='princely_musings'>
      <item>
        <presence xmlns='tigase:pubsub:1' node='princely_musings' jid='hag66@shake
      </item>
    </items>
  </event>
</message>
```

If component receives presence stanza with type unavailable without specified node, then component will log out user from all nodes he logged before and publish events.

## Retrieving list of all Node Subscribers

To retrieve list of node subscribers, node configuration option `tigase#allow_view_subscribers` must be set to true:

```
<iq type='set'
  from='hamlet@denmark.lit/elsinore'
  to='pubsub.shakespeare.lit'
  id='config2'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub#owner'>
    <configure node='princely_musings'>
      <x xmlns='jabber:x:data' type='submit'>
        <field var='FORM_TYPE' type='hidden'>
          <value>http://jabber.org/protocol/pubsub#node_config</value>
        </field>
        <field var='tigase#allow_view_subscribers'><value>1</value></field>
      </x>
    </configure>
  </pubsub>
</iq>
```

When option is enabled, each subscriber may get list of subscribers the same way as owner [<http://xmpp.org/extensions/xep-0060.html#owner-subscriptions-retrieve>].

```
<iq type='get'
  from='hamlet@denmark.lit/elsinore'
  to='pubsub.shakespeare.lit'
  id='subman1'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub#owner'>
    <subscriptions node='princely_musings' />
  </pubsub>
</iq>
```

There is extension to filter returned list:

```
<iq type='get'
  from='hamlet@denmark.lit/elsinore'
  to='pubsub.shakespeare.lit'
  id='subman1'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub#owner'>
    <subscriptions node='princely_musings'>
      <filter xmlns='tigase:pubsub:1'>
        <jid contains='@denmark.lit' -/>
      </filter>
    </subscriptions>
  </pubsub>
</iq>
```

In this example will be returned all subscriptions of users from domain "denmark.lit". == Offline Message Sink :author: Bartosz Malkowski :version: v2.0 November 2016. Reformatted for v8.0.0.

Messages sent to offline users is published in pubsub node, from where that message is sent to all the node subscribers as a pubsub notification.

```
<message from='pubsub.coffeebean.local' to='bard@shakespeare.lit' id='foo'>
  <event xmlns='http://jabber.org/protocol/pubsub#event'>
    <items node='message_sink'>
      <item id='ae890ac52d0df67ed7cfd51b644e901'>
        <message type="chat" xmlns="jabber:client" id="x2ps6u0004"
          to="userB_h6x1bt0002@coffeebean.local"
          from="userA_uyhx8p0001@coffeebean.local/1149352695-tigase-20">
          <body>Hello</body>
        </message>
      </item>
    </items>
  </event>
</message>
```

## Configuration

The pubsub node must be created and configured beforehand:

### Create node

```
<iq type='set'
  to='pubsub.coffeebean.local'
  id='create1'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <create node='message_sink' />
  </pubsub>
</iq>
```

```
</pubsub>
</iq>
```

After that is done, you need to add SessionManager as a publisher:

### Add sess-man as publisher

```
<iq type='set'
  to='pubsub.coffeebean.local'
  id='ent2'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub#owner'>
    <affiliations node='message_sink'>
      <affiliation jid='sess-man@coffeebean.local' affiliation='publisher' />
    </affiliations>
  </pubsub>
</iq>
```

Finally, the 'msgoffline' offline messages processor must be configured as well

### config.tdsl configuration

```
sess-man {
  msgoffline () {
    msg-pubsub-jid = -'pubsub.coffeebean.local'
    msg-pubsub-node = -'message_sink'
    msg-pubsub-publisher = -'sess-man@coffeebean.local'
  }
}
```

## Usage

Because these sinks use a standard pubsub component, administration of the sink node is identical to any other pubsub node. XEP-0060 [<http://www.xmpp.org/extensions/xep-0060>] defines standard pubsub usage and management.

## Managing Subscriptions

### Add new Subscriber

```
<iq type='set'
  to='pubsub.coffeebean.local'
  id='subman2'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub#owner'>
    <subscriptions node='message_sink'>
      <subscription jid='bard@shakespeare.lit' subscription='subscribed' />
    </subscriptions>
  </pubsub>
</iq>
```

### Remove Subscriber

```
<iq type='set'
  to='pubsub.coffeebean.local'
  id='subman2'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub#owner'>
    <subscriptions node='message_sink'>
```

```
<subscription jid='bard@shakespeare.lit' subscription='none' />
</subscriptions>
</pubsub>
</iq>
```

## REST API

All example calls to pubsub REST API are prepared for pubsub component running at `pubsub.example.com`. It is required to replace this value with JID of pubsub component from your installation.

It is possible to provide parameters to requests as:

**XML** All parameters passed in content of HTTP request needs to be wrapped with `<data/>` tag as root tag of XML document, while returned parameters will be wrapped `<result/>` tag as root tag of XML document.

**JSON** Parameters must be passed as serialized JSON object. Additionally `Content-Type` header of HTTP request needs to be set to `application/json`.

## Create a node

HTTP URI: `/rest/pubsub/pubsub.example.com/create-node`

Available HTTP methods:

### GET

Method returns example content which contains all required and optional parameters that may be passed to newly created node.

### POST

Command requires fields `node` and `pubsub#node_type` to be filled with proper values for execution.

- `node` - field should contain id of node to create
- `owner` - field may contains jid which should be used as jid of owner of newly created node (will use jid of Tigase HTTP API Component if not passed)
- `pubsub#node_type` - should contain type of node to create (two values are possible: `leaf` - node to which items will be published, `collection` - node which will contain other nodes)

Example content to create node of id `example` and of type `leaf` and with owner set to `admin@example.com`.

## Using XML

### Request in XML.

```
<data>
  <node>example</node>
  <owner>admin@example.com</owner>
  <pubsub prefix="true">
    <node_type>leaf</node_type>
```

```
</pubsub>
</data>
```

**Response in XML.**

```
<result>
  <Note type="fixed">
    <value>Operation successful</value>
  </Note>
</result>
```

**Using JSON****Request in JSON.**

```
{
  -"node" -: -"example",
  -"owner" -: -"admin@example.com",
  -"pubsub#node_type" -: -"leaf"
}
```

**Response in JSON.**

```
{
  -"Note": -"Operation successful"
}
```

**Delete a node**

HTTP URI: `/rest/pubsub/pubsub.example.com/delete-node`

Available HTTP methods:

**GET**

Method returns example content which contains all required and optional parameters that may be passed.

**POST**

Command requires field `node` to be filled.

- `node` - field should contain id of node to delete

Example content to delete node with id `example`

**Using XML****Request in XML.**

```
<data>
  <node>example</node>
</data>
```

**Response in XML.**

```
<result>
```

```
<Note type="fixed">
  <value>Operation successful</value>
</Note>
</result>
```

## Using JSON

### Request in JSON.

```
{
  -"node" -: -"example"
}
```

### Response in JSON.

```
{
  -"Note" -: -"Operation successful"
}
```

## Subscribe to a node

HTTP URI: `/rest/pubsub/pubsub.example.com/subscribe-node`

Available HTTP methods:

### GET

Method returns example content which contains all required and optional parameters that may be passed.

### POST

Command requires fields `node` and `jids` to be filled.

- `node` - field should contain id of node to subscribe to
- `jids` - field should contain list of jids to be subscribed to node

Example content to subscribe to node with id `example` users with jid `test1@example.com` and `test2@example.com`

## Using XML

### Request in XML.

```
<data>
  <node>example</node>
  <jids>
    <value>test1@example.com</value>
    <value>test2@example.com</value>
  </jids>
</data>
```

### Response in XML.

```
<result>
  <Note type="fixed">
```



```
<value>Operation successful</value>
</Note>
</result>
```

## Using JSON

### Request in JSON.

```
{
  -"node" -: -"example",
  -"jids" -: [
    -"test1@example.com",
    -"test2@example.com"
  -]
}
```

### Response in JSON.

```
{
  -"Note" -: -"Operation successful"
}
```

## Unsubscribe from a node

HTTP URI: `/rest/pubsub/pubsub.example.com/unsubscribe-node`

Available HTTP methods:

### GET

Method returns example content which contains all required and optional parameters that may be passed.

### POST

Command requires fields `node` and `jids` to be filled.

- `node` - field should contain id of node to unsubscribe from
- `jids` - field should contain list of jids to be unsubscribed from node

Example content to unsubscribe from node with id `example` users `test1@example.com` and `test2@example.com`

## Using XML

### Request in XML.

```
<data>
  <node>example</node>
  <jids>
    <value>test@example.com</value>
    <value>test2@example.com</value>
  </jids>
</data>
```

### Response in XML.

```
<result>
  <Note type="fixed">
    <value>Operation successful</value>
  </Note>
</result>
```

## Using JSON

### Request in JSON.

```
{
  -"node" -: -"example.com",
  -"jids" -: [
    -"test@example.com",
    -"test2@example.com"
  -]
}
```

### Response in JSON.

```
{
  -"Note" -: -"Operation successful"
}
```

## Publish an item to a node

HTTP URI: `/rest/pubsub/pubsub.example.com/publish-item`

Available HTTP methods:

### GET

Method returns example content which contains all required and optional parameters that may be passed.

### POST

Command requires fields `node` and `entry` to be filled

- `node` - field should contain id of node to publish to
- `item-id` - field may contain id of entry to publish
- `expire-at` - field may contain timestamp (in XEP-0082 [<http://xmpp.org/extensions/xep-0082.html>] format) after which item should not be delivered to user
- `entry` - field should contain multi-line entry content which should be valid XML value for an item

Example content to publish item with id `item-1` to node with id `example` and with content in `example` field. P

## Using XML

### with XML payload

In this example we will use following XML payload:

**Payload.**

```
<item-entry>
  <title>Example 1</title>
  <content>Example content</content>
</item-entry>
```

**Request in XML.**

```
<data>
  <node>example</node>
  <item-id>item-1</item-id>
  <expire-at>2015-05-13T16:05:00+02:00</expire-at>
  <entry>
    <item-entry>
      <title>Example 1</title>
      <content>Example content</content>
    </item-entry>
  </entry>
</data>
```

**Response in XML.**

```
<result>
  <Note type="fixed">
    <value>Operation successful</value>
  </Note>
</result>
```

**with JSON payload**

It is possible to publish JSON payload as value of published XML element. In example below we are publishing following JSON object:

**Payload.**

```
{ -"key-1" -: 2, -"key-2" -: -"value-2" -}
```

**Request in XML.**

```
<data>
  <node>example</node>
  <item-id>item-1</item-id>
  <expire-at>2015-05-13T16:05:00+02:00</expire-at>
  <entry>
    <payload>{ &quot;key-1&quot; -: 2, &quot;key-2&quot; -: &quot;value-2&quot; -}
  </entry>
</data>
```

**Response in XML.**

```
<result>
  <Note type="fixed">
    <value>Operation successful</value>
  </Note>
</result>
```

## Using JSON

### with XML payload

To publish XML using JSON you need to set serialized XML payload as value for entry key. In this example we will use following XML payload:

#### Payload.

```
<item-entry>
  <title>Example 1</title>
  <content>Example content</content>
</item-entry>
```

#### Request in JSON.

```
{
  -"node" -: -"example",
  -"item-id" -: -"item-1",
  -"expire-at" -: -"2015-05-13T16:05:00+02:00",
  -"entry" -: -"<item-entry>
    <title>Example 1</title>
    <content>Example content</content>
  </item-entry>"
}
```

#### Response in JSON.

```
{
  -"Note" -: -"Operation successful"
}
```

### with JSON payload

As JSON needs to be set as a value of an XML element it will be wrapped on server side as a value for <payload/> element.

#### Payload.

```
{ -"key-1" -: 2, -"key-2" -: -"value-2" -}
```

#### Request in JSON.

```
{
  -"node" -: -"example",
  -"item-id" -: -"item-1",
  -"expire-at" -: -"2015-05-13T16:05:00+02:00",
  -"entry" -: {
    -"key-1" -: 2,
    -"key-2" -: -"value-2"
  }
}
```

#### Response in JSON.

```
{
  -"Note" -: -"Operation successful"
}
```

```
}
```

**Published item.**

```
<payload>{ "key-1": 2, "key-2": "value-2" }
```

## Delete an item from a node

HTTP URI: `/rest/pubsub/pubsub.example.com/delete-item`

Available HTTP methods:

### GET

Method returns example content which contains all required and optional parameters that may be passed.

### POST

Command requires fields `node` and `item-id` to be filled

- `node` - field contains id of node to publish to
- `item-id` - field contains id of entry to publish

Example content to delete an item with id `item-1` from node with id `example`.

### Using XML

**Request in XML.**

```
<data>
  <node>example</node>
  <item-id>item-1</item-id>
</data>
```

**Response in XML.**

```
<result>
  <Note type="fixed">
    <value>Operation successful</value>
  </Note>
</result>
```

### Using JSON

**Request in JSON.**

```
{
  "node" : "example",
  "item-id" : "item-1"
}
```

**Response in JSON.**

```
{
  "Note" : "Operation successful"
}
```

```
}
```

## List available nodes

HTTP URI: `/rest/pubsub/pubsub.example.com/list-nodes`

Available HTTP methods:

### GET

Method returns list of available pubsub nodes for domain passed as part of URI (pubsub.example.com).

#### Example response in XML.

```
<result>
  <title>List of available nodes</title>
  <nodes label="Nodes" type="text-multi">
    <value>test</value>
    <value>node_54idf40037</value>
    <value>node_3ws5lz0037</value>
  </nodes>
</result>
```

in which we see nodes: test, node\_54idf40037 and node\_3ws5lz0037.

#### Example response in JSON.

```
{
  -"title" -: -"List of available nodes",
  -"nodes" -: [
    -"test",
    -"node_54idf40037",
    -"node_3ws5lz0037"
  -]
}
```

in which we see nodes: test, node\_54idf40037 and node\_3ws5lz0037.

## List published items on node

HTTP URI: `/rest/pubsub/pubsub.example.com/list-items`

Available HTTP methods:

### GET

Method returns example content which contains all required and optional parameters that may be passed.

### POST

Command requires field node to be filled

- node - field contains id of node which items we want to list

Example content to list of items published on node with id example.

## Using XML

### Request in XML.

```
<data>
  <node>example</node>
</data>
```

### Response in XML.

```
<result>
  <title>List of PubSub node items</title>
  <node label="Node" type="text-single">
    <value>example</value>
  </node>
  <items label="Items" type="text-multi">
    <value>item-1</value>
    <value>item-2</value>
  </items>
</result>
```

where item-1 and item-2 are identifiers of published items for node example.

## Using JSON

### Request in JSON.

```
{
  -"node" -: -"example"
}
```

### Response in JSON.

```
{
  -"title" -: -"List of PubSub node items",
  -"node" -: -"example",
  -"items" -: [
    -"item-1",
    -"item-2"
  ]
}
```

where item-1 and item-2 are identifiers of published items for node example.

## Retrieve item published on node

HTTP URI: /rest/pubsub/pubsub.example.com/retrieve-item

Available HTTP methods:

### GET

Method returns example content which contains all required and optional parameters that may be passed.

### POST

Command requires fields node and item-id to be filled

- node - field contains id of node which items we want to list
- item-id - field contains id of item to retrieve

Example content to list of items published on node with id `example`.

## Using XML

### Request in XML.

```
<data>
  <node>example</node>
  <item-id>item-1</item-id>
</data>
```

### Response in XML.

```
<result>
  <title>Retrieve PubSub node item</title>
  <node label="Node" type="text-single">
    <value>example</value>
  </node>
  <item-id label="Item ID" type="text-single">
    <value>item-1</value>
  </item-id>
  <item label="Item" type="text-multi">
    <value>
      <item expire-at="2015-05-13T14:05:00Z" id="item-1">
        <item-entry>
          <title>Example 1</title>
          <content>Example content</content>
        </item-entry>
      </item>
    </value>
  </item>
</result>
```

inside item element there is XML encoded element which is published on node `example` with id `item-1`.

## Using JSON

### Request in JSON.

```
{
  -"node" -: -"example",
  -"item-id" -: -"item-1"
}
```

### Response in JSON.

```
{
  -"title" -: -"Retrieve PubSub node item",
  -"node" -: -"example",
  -"item-id" -: -"item-1",
  -"item" -: [
    -"<item expire-at=\"2015-05-13T14:05:00Z\" id=\"item-1\">"
  ]
}
```



```
        <item-entry>
          <title>Example 1</title>
          <content>Example content</content>
        </item-entry>
      </item>"
    -]
  }
```

## Retrieve user subscriptions

HTTP URI: `/rest/pubsub/pubsub.example.com/retrieve-user-subscriptions`

Available HTTP methods:

### GET

Method returns example content which contains all required and optional parameters that may be passed.

### POST

Command requires field `jid` to be filled.

- `jid` - field contains JID of a user for which we want to retrieve subscriptions
- `node-pattern` - field contains regex pattern to match. When field is not empty, request will return only subscribed nodes which match this pattern. If field should be empty it may be omitted in a request.

Example content to retrieve list of nodes to which user `test@example.com` is subscribed at `pubsub.example.com` which starts with `test-` (pattern `test-.*`)

### Using XML

#### Request in XML.

```
<data>
  <jid>test@example.com</jid>
  <node-pattern>test-.*</node-pattern>
</data>
```

#### Response in XML.

```
<result>
  <nodes label="Nodes" type="text-multi">
    <value>test-123</value>
    <value>test-342</value>
  </nodes>
</result>
```

### Using JSON

#### Request in JSON.

```
{
  -"jid" -: -"test@example.com",
  -"node-pattern" -: -"test-.*"
}
```

**Response in JSON.**

```
{
  -"nodes" -: [
    -"test-123",
    -"test-342"
  -]
}
```

## AdHoc Commands

Similar to the HTTP API, AdHoc commands based on groovy scripts can be sent to this component to do a number of tasks. All scripts for these Ad-hoc commands are found at `sec/main/groovy/tigase/admin` in source distributions, or at this link [<https://projects.tigase.org/projects/tigase-pubsub/repository/revisions/master/show/src/main/groovy/tigase/admin>]. To use them, the scripts need to be copied into the `scripts/admin/pubsub` folder in the Tigase installation directory. For all examples, the component address will be `pubsub.example.com`.

### Create a Node

Ad-hoc command node: `create-node` Required role: Service Administrator

Command requires fields `node` and `pubsub#node_type` to be filled with proper values for execution. - `node` Field containing id of node to create. - `pubsub#node_type` Contains one of two possible values. \* `leaf-node` Node that will be published. \* `collection` Node that will contain other nodes.

Other fields are optional fields that can be set to change configuration of newly create node to different configuration than default.

Example call using TCLMT:

```
bin/tclmt.sh --u admin@example.com --p admin123 remote pubsub.example.com create-n
```

### Delete a Node

Ad-hoc command node: `delete-node` Required role: Service Administrator

Command requires `node` field to be filled. - `node` Field containing id of node to delete.

Example call using TCLMT:

```
bin/tclmt.sh --u admin@example.com --p admin123 remote pubsub.example.com delete-n
```

### Subscribe to a Node

Ad-hoc command node: `subscribe-node` Required role: Service Administrator

Command requires `node` and `jids` nodes to be filled. - `node` Field containing node to subscribe to. - `jids` Field containing list of JIDs to subscribe to the node.

Example call using TCLMT:

```
bin/tclmt.sh --u admin@example.com --p admin123 remote pubsub.example.com subscrib
```

### Unsubscribe to a Node

Ad-hoc command node: `unsubscribe-node` Required role: Service Administrator

Command requires `node` and `jids` nodes to be filled. - `node` Field containing node to unsubscribe to.  
- `jids` Field containing list of JIDs to unsubscribe to the node.

Example call using TCLMT:

```
bin/tclmt.sh --u admin@example.com --p admin123 remote pubsub.example.com unsubscribe
```

## Publish an item to a Node

Ad-hoc command `node:publish-item` Required role: Service Administrator

Command requires fields `node` and `entry` to be filled. - `node` Field containing id of node to publish to. - `item-id` Field may contain id of entry to publish, can be empty. - `entry` Field should contain multi-line entry content that should be valid XML values for items.

This command due to it's complexity cannot be easily executed by TCLMT using default remote script which provides support for basic adhoc commands. Example call using TCLMT:

```
bin/tclmt.sh --u admin@example.com --p admin123 remote pubsub.example.com publish-
```

Example Groovy script to execute `create-node` command using JAXMPP2

```
import tigase.jaxmpp.j2se.Jaxmpp
import tigase.jaxmpp.core.client.AsyncCallback
import tigase.jaxmpp.core.client.exceptions.JaxmppException
import tigase.jaxmpp.core.client.xmlpp.stanzas.Stanza
import tigase.jaxmpp.core.client.SessionObject
import tigase.jaxmpp.j2se.ConnectionConfiguration
import tigase.jaxmpp.core.client.xml.Element
import tigase.jaxmpp.core.client.xml.DefaultElement
import tigase.jaxmpp.core.client.xmlpp.forms.JabberDataElement

Jaxmpp jaxmpp = new Jaxmpp();

jaxmpp.with {
    getConnectionConfiguration().setConnectionType(ConnectionConfiguration.CONNECT_TYPE_TCP);
    getConnectionConfiguration().setUserJID("admin@example.com");
    getConnectionConfiguration().setUserPassword("admin123");
}

jaxmpp.login(true);

def packet = IQ.create();
packet.setAttribute("to", -"pubsub.example.com");

Element command = new DefaultElement("command");
command.setXMLNS("http://jabber.org/protocol/commands");
command.setAttribute("node", -"create-node");
packet.addChild(command);

Element x = new DefaultElement("x");
x.setXMLNS("jabber:x:data");

command.addChild(x);

def data = new JabberDataElement(x);
```

```
data.addTextSingleField("node", "-example");
data.addListSingleField("pubsub#node_type", "-leaf");

jaxmpp.send(packet, new AsyncCallback() {
    void onError(Stanza responseStanza, tigase.jaxmpp.core.client.XMPPEException.Er
        println "-received error during processing request";
    -}

    void onSuccess(Stanza responseStanza) throws JaxmppException {
        x = responseStanza.getFirstChild("command").getFirstChid("x");
        data = new JabberDataElement(x);
        def error = data.getField("Error");
        println "-command executed with result = -" + (error -? "-failure, error =
    -}

    void onTimeout() {
        println "-command timed out"
    -}
});

Thread.sleep(30000);
jaxmpp.disconnect();
```

## PubSub Node Presence Protocol

### Occupant Use Case

#### Log in to Pubsub Node

To log in to PubSub Node user must send presence to PubSub component with additional information about node:

```
<presence
  from='hag66@shakespeare.lit/pda'
  id='n13mt31'
  to='pubsub.shakespeare.lit'>
  <pubsub xmlns='tigase:pubsub:1' node='princely_musings' />
</presence>
```

Component will publish this information in node:

```
<message from='pubsub.shakespeare.lit' to='francisco@denmark.lit' id='foo'>
  <event xmlns='http://jabber.org/protocol/pubsub#event'>
    <items node='princely_musings'>
      <item>
        <presence xmlns='tigase:pubsub:1' node='princely_musings' jid='hag66@shake
      </item>
    </items>
  </event>
</message>
<message from='pubsub.shakespeare.lit' to='bernardo@denmark.lit' id='bar'>
  <event xmlns='http://jabber.org/protocol/pubsub#event'>
    <items node='princely_musings'>
      <item>
        <presence xmlns='tigase:pubsub:1' node='princely_musings' jid='hag66@shake
```

```
    </item>
  </items>
</event>
</message>
```

And then will send notification with presences of all occupants to new occupant.

## Log out from PubSub Node

To logout from single node, user must send presence stanza with type unavailable:

```
<presence
  from='hag66@shakespeare.lit/pda'
  type='unavailable'
  to='pubsub.shakespeare.lit'>
  <pubsub xmlns='tigase:pubsub:1' node='princely_musings' />
</presence>
```

Component will send events to all occupants as described:

```
<message from='pubsub.shakespeare.lit' to='francisco@denmark.lit' id='foo'>
  <event xmlns='http://jabber.org/protocol/pubsub#event'>
    <items node='princely_musings'>
      <item>
        <presence xmlns='tigase:pubsub:1' node='princely_musings' jid='hag66@shake
      </item>
    </items>
  </event>
</message>
```

If component receives presence stanza with type unavailable without specified node, then component will log out user from all nodes he logged before and publish events.

## Retrieving list of all Node Subscribers

To retrieve list of node subscribers, node configuration option `tigase#allow_view_subscribers` must be set to true:

```
<iq type='set'
  from='hamlet@denmark.lit/elsinore'
  to='pubsub.shakespeare.lit'
  id='config2'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub#owner'>
    <configure node='princely_musings'>
      <x xmlns='jabber:x:data' type='submit'>
        <field var='FORM_TYPE' type='hidden'>
          <value>http://jabber.org/protocol/pubsub#node_config</value>
        </field>
        <field var='tigase#allow_view_subscribers'><value>1</value></field>
      </x>
    </configure>
  </pubsub>
</iq>
```

When option is enabled, each subscriber may get list of subscribers the same way as owner [<http://xmpp.org/extensions/xep-0060.html#owner-subscriptions-retrieve>].

```
<iq type='get'
  from='hamlet@denmark.lit/elsinore'
  to='pubsub.shakespeare.lit'
  id='subman1'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub#owner'>
    <subscriptions node='princely_musings' />
  </pubsub>
</iq>
```

There is extension to filter returned list:

```
<iq type='get'
  from='hamlet@denmark.lit/elsinore'
  to='pubsub.shakespeare.lit'
  id='subman1'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub#owner'>
    <subscriptions node='princely_musings'>
      <filter xmlns='tigase:pubsub:1'>
        <jid contains='@denmark.lit' -/>
      </filter>
    </subscriptions>
  </pubsub>
</iq>
```

In this example will be returned all subscriptions of users from domain "denmark.lit".

## Offline Message Sink

Messages sent to offline users is published in pubsub node, from where that message is sent to all the node subscribers as a pubsub notification.

```
<message from='pubsub.coffeebean.local' to='bard@shakespeare.lit' id='foo'>
  <event xmlns='http://jabber.org/protocol/pubsub#event'>
    <items node='message_sink'>
      <item id='ae890ac52d0df67ed7cfd51b644e901'>
        <message type="chat" xmlns="jabber:client" id="x2ps6u0004"
          to="userB_h6x1bt0002@coffeebean.local"
          from="userA_uyxh8p0001@coffeebean.local/1149352695-tigase-20">
          <body>Hello</body>
        </message>
      </item>
    </items>
  </event>
</message>
```

## Configuration

The pubsub node must be created and configured beforehand:

### Create node

```
<iq type='set'
  to='pubsub.coffeebean.local'
  id='createl'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <create node='message_sink' />
  </pubsub>
</iq>
```

```
</pubsub>
</iq>
```

After that is done, you need to add SessionManager as a publisher:

### Add sess-man as publisher

```
<iq type='set'
  to='pubsub.coffeebean.local'
  id='ent2'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub#owner'>
    <affiliations node='message_sink'>
      <affiliation jid='sess-man@coffeebean.local' affiliation='publisher' />
    </affiliations>
  </pubsub>
</iq>
```

Finally, the 'msgoffline' offline messages processor must be configured as well

### config.tdsl configuration

```
sess-man {
  msgoffline () {
    msg-pubsub-jid = -'pubsub.coffeebean.local'
    msg-pubsub-node = -'message_sink'
    msg-pubsub-publisher = -'sess-man@coffeebean.local'
  }
}
```

### Usage

Because these sinks use a standard pubsub component, administration of the sink node is identical to any other pubsub node. XEP-0060 [<http://www.xmpp.org/extensions/xep-0060>] defines standard pubsub usage and management.

## Managing Subscriptions

### Add new Subscriber

```
<iq type='set'
  to='pubsub.coffeebean.local'
  id='subman2'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub#owner'>
    <subscriptions node='message_sink'>
      <subscription jid='bard@shakespeare.lit' subscription='subscribed' />
    </subscriptions>
  </pubsub>
</iq>
```

### Remove Subscriber

```
<iq type='set'
  to='pubsub.coffeebean.local'
  id='subman2'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub#owner'>
    <subscriptions node='message_sink'>
```

```
<subscription jid='bard@shakespeare.lit' subscription='none' />
</subscriptions>
</pubsub>
</iq>
```

## REST API

All example calls to pubsub REST API are prepared for pubsub component running at `pubsub.example.com`. It is required to replace this value with JID of pubsub component from your installation.

It is possible to provide parameters to requests as:

**XML** All parameters passed in content of HTTP request needs to be wrapped with `<data/>` tag as root tag of XML document, while returned parameters will be wrapped `<result/>` tag as root tag of XML document.

**JSON** Parameters must be passed as serialized JSON object. Additionally `Content-Type` header of HTTP request needs to be set to `application/json`.

### Create a node

HTTP URI: `/rest/pubsub/pubsub.example.com/create-node`

Available HTTP methods:

#### GET

Method returns example content which contains all required and optional parameters that may be passed to newly created node.

#### POST

Command requires fields `node` and `pubsub#node_type` to be filled with proper values for execution.

- `node` - field should contain id of node to create
- `owner` - field may contains jid which should be used as jid of owner of newly created node (will use jid of Tigase HTTP API Component if not passed)
- `pubsub#node_type` - should contain type of node to create (two values are possible: `leaf` - node to which items will be published, `collection` - node which will contain other nodes)

Example content to create node of id `example` and of type `leaf` and with owner set to `admin@example.com`.

### Using XML

#### Request in XML.

```
<data>
  <node>example</node>
  <owner>admin@example.com</owner>
  <pubsub prefix="true">
    <node_type>leaf</node_type>
  </pubsub>
</data>
```



**Response in XML.**

```
<result>
  <Note type="fixed">
    <value>Operation successful</value>
  </Note>
</result>
```

**Using JSON****Request in JSON.**

```
{
  -"node" -: -"example",
  -"owner" -: -"admin@example.com",
  -"pubsub#node_type" -: -"leaf"
}
```

**Response in JSON.**

```
{
  -"Note": -"Operation successful"
}
```

**Delete a node**

HTTP URI: /rest/pubsub/pubsub.example.com/delete-node

Available HTTP methods:

**GET**

Method returns example content which contains all required and optional parameters that may be passed.

**POST**

Command requires field node to be filled.

- node - field should contain id of node to delete

Example content to delete node with id example

**Using XML****Request in XML.**

```
<data>
  <node>example</node>
</data>
```

**Response in XML.**

```
<result>
  <Note type="fixed">
    <value>Operation successful</value>
  </Note>
</result>
```

## Using JSON

### Request in JSON.

```
{
  -"node" -: -"example"
}
```

### Response in JSON.

```
{
  -"Note" -: -"Operation successful"
}
```

## Subscribe to a node

HTTP URI: `/rest/pubsub/pubsub.example.com/subscribe-node`

Available HTTP methods:

### GET

Method returns example content which contains all required and optional parameters that may be passed.

### POST

Command requires fields `node` and `jids` to be filled.

- `node` - field should contain id of node to subscribe to
- `jids` - field should contain list of jids to be subscribed to node

Example content to subscribe to node with id `example` users with `jid test1@example.com` and `test2@example.com`

## Using XML

### Request in XML.

```
<data>
  <node>example</node>
  <jids>
    <value>test1@example.com</value>
    <value>test2@example.com</value>
  </jids>
</data>
```

### Response in XML.

```
<result>
  <Note type="fixed">
    <value>Operation successful</value>
  </Note>
</result>
```

## Using JSON

### Request in JSON.

```
{
  -"node" -: -"example",
  -"jids" -: [
    -"test1@example.com",
    -"test2@example.com"
  -]
}
```

**Response in JSON.**

```
{
  -"Note" -: -"Operation successful"
}
```

**Unsubscribe from a node**

HTTP URI: `/rest/pubsub/pubsub.example.com/unsubscribe-node`

Available HTTP methods:

**GET**

Method returns example content which contains all required and optional parameters that may be passed.

**POST**

Command requires fields `node` and `jids` to be filled.

- `node` - field should contain id of node to unsubscribe from
- `jids` - field should contain list of jids to be unsubscribed from node

Example content to unsubscribe from node with id `example` users `test1@example.com` and `test2@example.com`

**Using XML****Request in XML.**

```
<data>
  <node>example</node>
  <jids>
    <value>test@example.com</value>
    <value>test2@example.com</value>
  </jids>
</data>
```

**Response in XML.**

```
<result>
  <Note type="fixed">
    <value>Operation successful</value>
  </Note>
</result>
```

**Using JSON****Request in JSON.**

```
{
  -"node" -: -"example.com",
  -"jids" -: [
    -"test@example.com",
    -"test2@example.com"
  ]
}
```

**Response in JSON.**

```
{
  -"Note" -: -"Operation successful"
}
```

**Publish an item to a node**

HTTP URI: `/rest/pubsub/pubsub.example.com/publish-item`

Available HTTP methods:

**GET**

Method returns example content which contains all required and optional parameters that may be passed.

**POST**

Command requires fields `node` and `entry` to be filled

- `node` - field should contain id of node to publish to
- `item-id` - field may contain id of entry to publish
- `expire-at` - field may contain timestamp (in XEP-0082 [<http://xmpp.org/extensions/xep-0082.html>] format) after which item should not be delivered to user
- `entry` - field should contain multi-line entry content which should be valid XML value for an item

Example content to publish item with id `item-1` to node with id `example` and with content in `example` field. P

**Using XML****with XML payload**

In this example we will use following XML payload:

**Payload.**

```
<item-entry>
  <title>Example 1</title>
  <content>Example content</content>
</item-entry>
```

**Request in XML.**

```
<data>
  <node>example</node>
```

```
<item-id>item-1</item-id>
<expire-at>2015-05-13T16:05:00+02:00</expire-at>
<entry>
  <item-entry>
    <title>Example 1</title>
    <content>Example content</content>
  </item-entry>
</entry>
</data>
```

**Response in XML.**

```
<result>
  <Note type="fixed">
    <value>Operation successful</value>
  </Note>
</result>
```

**with JSON payload**

It is possible to publish JSON payload as value of published XML element. In example below we are publishing following JSON object:

**Payload.**

```
{ -"key-1" -: 2, -"key-2" -: -"value-2" -}
```

**Request in XML.**

```
<data>
  <node>example</node>
  <item-id>item-1</item-id>
  <expire-at>2015-05-13T16:05:00+02:00</expire-at>
  <entry>
    <payload>{ &quot;key-1&quot; -: 2, &quot;key-2&quot; -: &quot;value-2&quot; -}
  </entry>
</data>
```

**Response in XML.**

```
<result>
  <Note type="fixed">
    <value>Operation successful</value>
  </Note>
</result>
```

**Using JSON****with XML payload**

To publish XML using JSON you need to set serialized XML payload as value for entry key. In this example we will use following XML payload:

**Payload.**

```
<item-entry>
  <title>Example 1</title>
```

```
<content>Example content</content>
</item-entry>
```

**Request in JSON.**

```
{
  -"node" -: -"example",
  -"item-id" -: -"item-1",
  -"expire-at" -: -"2015-05-13T16:05:00+02:00",
  -"entry" -: -"<item-entry>
    <title>Example 1</title>
    <content>Example content</content>
  </item-entry>"
}
```

**Response in JSON.**

```
{
  -"Note" -: -"Operation successful"
}
```

**with JSON payload**

As JSON needs to be set as a value of an XML element it will be wrapped on server side as a value for `<payload/>` element.

**Payload.**

```
{ -"key-1" -: 2, -"key-2" -: -"value-2" -}
```

**Request in JSON.**

```
{
  -"node" -: -"example",
  -"item-id" -: -"item-1",
  -"expire-at" -: -"2015-05-13T16:05:00+02:00",
  -"entry" -: {
    -"key-1" -: 2,
    -"key-2" -: -"value-2"
  }
}
```

**Response in JSON.**

```
{
  -"Note" -: -"Operation successful"
}
```

**Published item.**

```
<payload>{ &quot;key-1&quot;; -: 2, &quot;key-2&quot;; -: &quot;value-2&quot;; -}</pa
```

**Delete an item from a node**

HTTP URI: `/rest/pubsub/pubsub.example.com/delete-item`

Available HTTP methods:

**GET**

Method returns example content which contains all required and optional parameters that may be passed.

**POST**

Command requires fields `node` and `item-id` to be filled

- `node` - field contains id of node to publish to
- `item-id` - field contains id of entry to publish

Example content to delete an item with id `item-1` from node with id `example`.

**Using XML****Request in XML.**

```
<data>
  <node>example</node>
  <item-id>item-1</item-id>
</data>
```

**Response in XML.**

```
<result>
  <Note type="fixed">
    <value>Operation successful</value>
  </Note>
</result>
```

**Using JSON****Request in JSON.**

```
{
  -"node" -: -"example",
  -"item-id" -: -"item-1"
}
```

**Response in JSON.**

```
{
  -"Note" -: -"Operation successful"
}
```

**List available nodes**

HTTP URI: `/rest/pubsub/pubsub.example.com/list-nodes`

Available HTTP methods:

**GET**

Method returns list of available pubsub nodes for domain passed as part of URI (`pubsub.example.com`).

**Example response in XML.**

```
<result>
  <title>List of available nodes</title>
  <nodes label="Nodes" type="text-multi">
    <value>test</value>
    <value>node_54idf40037</value>
    <value>node_3ws5lz0037</value>
  </nodes>
</result>
```

in which we see nodes: test, node\_54idf40037 and node\_3ws5lz0037.

#### **Example response in JSON.**

```
{
  -"title" -: -"List of available nodes",
  -"nodes" -: [
    -"test",
    -"node_54idf40037",
    -"node_3ws5lz0037"
  -]
}
```

in which we see nodes: test, node\_54idf40037 and node\_3ws5lz0037.

## **List published items on node**

HTTP URI: /rest/pubsub/pubsub.example.com/list-items

Available HTTP methods:

### **GET**

Method returns example content which contains all required and optional parameters that may be passed.

### **POST**

Command requires field node to be filled

- node - field contains id of node which items we want to list

Example content to list of items published on node with id example.

## **Using XML**

### **Request in XML.**

```
<data>
  <node>example</node>
</data>
```

### **Response in XML.**

```
<result>
  <title>List of PubSub node items</title>
  <node label="Node" type="text-single">
    <value>example</value>
  </node>
  <items label="Items" type="text-multi">
```



```
<value>item-1</value>
<value>item-2</value>
</items>
</result>
```

where item-1 and item-2 are identifiers of published items for node example.

## Using JSON

### Request in JSON.

```
{
  -"node" -: -"example"
}
```

### Response in JSON.

```
{
  -"title" -: -"List of PubSub node items",
  -"node" -: -"example",
  -"items" -: [
    -"item-1",
    -"item-2"
  ]
}
```

where item-1 and item-2 are identifiers of published items for node example.

## Retrieve item published on node

HTTP URI: /rest/pubsub/pubsub.example.com/retrieve-item

Available HTTP methods:

### GET

Method returns example content which contains all required and optional parameters that may be passed.

### POST

Command requires fields node and item-id to be filled

- node - field contains id of node which items we want to list
- item-id - field contains id of item to retrieve

Example content to list of items published on node with id example.

## Using XML

### Request in XML.

```
<data>
  <node>example</node>
  <item-id>item-1</item>
</data>
```

### Response in XML.

```
<result>
  <title>Retrieve PubSub node item</title>
  <node label="Node" type="text-single">
    <value>example</value>
  </node>
  <item-id label="Item ID" type="text-single">
    <value>item-1</value>
  </item-id>
  <item label="Item" type="text-multi">
    <value>
      <item expire-at="2015-05-13T14:05:00Z" id="item-1">
        <item-entry>
          <title>Example 1</title>
          <content>Example content</content>
        </item-entry>
      </item>
    </value>
  </item>
</result>
```

inside item element there is XML encoded element which is published on node example with id item-1.

## Using JSON

### Request in JSON.

```
{
  -"node" -: -"example",
  -"item-id" -: -"item-1"
}
```

### Response in JSON.

```
{
  -"title" -: -"Retrieve PubSub node item",
  -"node" -: -"example",
  -"item-id" -: -"item-1",
  -"item" -: [
    -"<item expire-at\"2015-05-13T14:05:00Z\" id=\"item-1\">
      <item-entry>
        <title>Example 1</title>
        <content>Example content</content>
      </item-entry>
    </item>"
  ]
}
```

## Retrieve user subscriptions

HTTP URI: /rest/pubsub/pubsub.example.com/retrieve-user-subscriptions

Available HTTP methods:

### GET

Method returns example content which contains all required and optional parameters that may be passed.

## POST

Command requires field `jid` to be filled.

- `jid` - field contains JID of a user for which we want to retrieve subscriptions
- `node-pattern` - field contains regex pattern to match. When field is not empty, request will return only subscribed nodes which match this pattern. If field should be empty it may be omitted in a request.

Example content to retrieve list of nodes to which user `test@example.com` is subscribed at `pubsub.example.com` which starts with `test-` (pattern `test-.*`)

## Using XML

### Request in XML.

```
<data>
  <jid>test@example.com</jid>
  <node-pattern>test-.*</node-pattern>
</data>
```

### Response in XML.

```
<result>
  <nodes label="Nodes" type="text-multi">
    <value>test-123</value>
    <value>test-342</value>
  </nodes>
</result>
```

## Using JSON

### Request in JSON.

```
{
  -"jid" -: -"test@example.com",
  -"node-pattern" -: -"test-.*"
}
```

### Response in JSON.

```
{
  -"nodes" -: [
    -"test-123",
    -"test-342"
  ]
}
```

# Limitations

## Addressing

Within Tigase, all pubsub component address **MUST** be domain-based address and not a JID style address. This was made to simplify communications structure. Tigase will automatically set component names to `pubsub.domain`, however any messages send to `pubsub@domain` will result in a `SERVICE_UNAVAILABLE` error.

Pubsub nodes within Tigase can be found as a combination of JID and node where nodes will be identified akin to service discovery. For example, to address a friendly node, use the following structure:

```
<iq to='pubsub.domain'>  
  <query node='friendly node' />  
</iq>
```

---

# Chapter 19. Tigase Socks5 Proxy

Welcome to Tigase Socks5 Proxy guide

Tigase SOCKS5 component allows for file transfers to be made over a SOCKS5 proxy in accordance with XEP-0065 SOCKS5 Bytestreams [<http://xmpp.org/extensions/xep-0065.html>]. This allows for some useful features such as: - transfer limits per user, domain, or global - recording transfers between users - quotas and credits system implementation

## Overview

Tigase Socks5 Proxy is implementation of Socks5 proxy described in XEP-0065: SOCKS5 Bytestreams, in section 6. Mediated Connection [<https://xmpp.org/extensions/xep-0065.html#mediated:>] which provides support for Socks5 proxy for file transfers between XMPP client behind NATs to Tigase XMPP Server.

## Installation

Tigase SOCKS5 component comes built into the dist-max archives for Tigase XMPP server, and requires the component to be listed in config.tdsl file:

```
proxy {}
```

You will also need to decide if you wish to use database-based features or not. If you wish to simply run the socks5 proxy without features such as quotas, limits add the following line:

```
proxy {  
    -'verifier-class' = -'tigase.socks5.verifiers.DummyVerifier'  
}
```

This will enable the SOCKS5 Proxy without any advanced features. If you wish to use those features, see the configuration section below.

## Database Preparation

In order to use the more advanced features of the SOCKS5 Proxy Component, your database needs to be prepared with the proper schema prior to running the server.

You may either edit an existing database, or create a new database for specific use with the Proxy.

## Edit Existing Database

You can add the proper schema to your existing database using the DBSchemaLoader utility included with Tigase. The database folder contains the schema file for your type of database.

First, backup your database before performing any actions and shut down Tigase XMPP Server.

Then from the Tigase installation directory run the following command:

```
java --cp -"jars/*" tigase.db.util.DBSchemaLoader --dbType {derby,mysql,postgresql}
```

You should see the following dialogue

```
LogLevel: CONFIG
```

tigase.db.util.DBSchemaLoader	<init>	CONFIG	Properties: [{dbH
tigase.db.util.DBSchemaLoader	validateDBConnection	INFO	Validating D
tigase.db.util.DBSchemaLoader	validateDBConnection	CONFIG	DriverManage
tigase.db.util.DBSchemaLoader	validateDBConnection	INFO	Connection O
tigase.db.util.DBSchemaLoader	validateDBExists	INFO	Validating wheth
tigase.db.util.DBSchemaLoader	validateDBExists	INFO	Exists OK
tigase.db.util.DBSchemaLoader	loadSchemaFile	INFO	Loading schema fr
tigase.db.util.DBSchemaLoader	loadSchemaFile	INFO	completed OK
tigase.db.util.DBSchemaLoader	shutdownDerby	INFO	Validating DBConn
tigase.db.util.DBSchemaLoader	shutdownDerby	WARNING	Database - 'tigase
tigase.db.util.DBSchemaLoader	printInfo	INFO	

One this process is complete, you may begin using SOCKS5 proxy component.

## Create New Database

If you want to create a new database for the proxy component and use it as a separate socks5 database, create the database using the appropriate schema file in the database folder. Once this is created, add the following line to your config.tdsl folder.

```
proxy {}
```

For example, a mysql database will have this type of URL: jdbc:mysql://localhost/SOCKS?user=root&password=root to replace database URL. For more options, check the database section of this documentation.

# Configuration

## Enabling proxy

To enable Tigase Socks5 Proxy component for Tigase XMPP Server, you need to activate socks5 component in Tigase XMPP Server configuration file (etc/config.tdsl). In simples solution it will work without ability to enforce any limits but will also work without a need of database to store informations about used bandwidth.

### Simple configuration.

```
socks5 () {
  repository {
    default () {
      cls = -'dummy'
    }
  }
}
```

## remote-addresses

```
proxy {
  -'remote-addresses' = -'192.168.1.205,20.255.13.190'
}
```

Comma separated list of IP addresses that will be accessible VIA the Socks5 Proxy. This can be useful if you want to specify a specific router address to allow external traffic to transfer files using the proxy to users on an internal network.

## Port settings

If socks5 is being used as a proxy, you may configure a specific ports for the proxy using the following line in config.tdsl:

```
proxy {
  -'connections' {
    -'ports' = [ 1080 -]
  }
}
```

## Enabling limits

To enable limits you need to import schema files proper for your database and related to Tigase Socks5 Proxy component from database directory. To do this, refer to the previous section.

With that setup, it is possible to enable limits verifier by replacing entries related to Tigase Socks5 Proxy component configuration with following entries. This will use default database configured to use with Tigase XMPP Server.

### DummyVerifier

- Class Name: `tigase.socks5.verifiers.DummyVerifier`

This accepts file transfers VIA SOCKS5 proxy from any user and does not check limitations against the database.

```
socks5 () {
  verifier (class: tigase.socks5.verifiers.DummyVerifier) {
  }
}
```

### LimitsVerifier

- Class Name: `tigase.socks5.verifiers.LimitsVerifier`

Uses the database to store limits and record the amount of data transferred VIA the proxy.

### Configuring limits

Following properties are possible to be set for LimitsVerifier:

```
proxy {
  -'verifier-class' = -'tigase.socks5.verifiers.LimitsVerifier'
  tigase.socks5.verifiers.LimitsVerifier {
    -'transfer-update-quantization' = -'1000'
    -'instance-limit' = -'3000'
  }
}
```

Parameters for `LimitsVerifier` which will override the defaults. All of these limits are on a per calendar month basis. For example, a user is limited to 10MB for all transfers. If he transfers 8MB between the 1st and the 22nd, he only has 2MB left in his limit. On the 1st of the following month, his limit is reset to 10MB.

Available parameters:

- `transfer-update-quantization` which value is used to quantize value to check if value of transferred bytes should be updated in database or not. By default it is 1MB. (Low value can slow down file transfer while high value can allow to exceed quota)
- `global-limit` - Transfer limit for all domains in MB per month.
- `instance-limit` - Transfer limit for server instance in MB per month.
- `default-domain-limit` - The Default transfer limit per domain in MB per month.
- `default-user-limit` - The default transfer limit per user in MB per month.
- `default-file-limit` - The default transfer limit per file in MB per month.

## Note

Low values can slow down file transfers, while high values can allow for users to exceed quotas.

## Individual Limits

Using the default database schema in table `tig_socks5_users` limits can be specified for individual users.

Value of the field `user_id` denotes the scope of the limitation:

- `domain_name` defines limits for users which JIDs are within that domain;
- `JID` of the user defines limit for this exact user.

Value of the limit bigger than 0 defines an exact value. If value is equal 0 limit is not override and more global limit is used. If value equals -1 proxy will forbid any transfer for this user. If there is no value for user in this table new row will be created during first transfer and limits for domain or global limits will be used.

Socks5 database is setup in this manner:

**Table 19.1. `tig_socks5_users`**

uid	user_id	sha1_user_id	domain	sha1_domain	filesize_limit	transfer_limit	transfer_limit_per_domain
1	user@domain.com [mailto:user@domain.com]	c35f2956d804e01a12092100da23039d33f02733040fe8005f05a257a	domain.com	9100da23039d33f02733040fe8005f05a257a	500	3000	3000
2	domain.com	e1000db219f3a68b0f02735000f8009f306257a	domain.com	9100da23039d33f02733040fe8005f05a257a	500	3000	3000

This example table shows that `user@domain.com [mailto:user@domain.com]` is limited to 3000MB per transfer whereas all users of `domain.com` are limited to a max file size of 500MB. This table will populate as users transfer files using the SOCKS5 proxy, once it begins population, you may edit it as necessary. A second database is setup `tig_socks5_connections` that records the connections and transmissions being made, however it does not need to be edited.

## Using a separate database

To use separate database with Tigase Socks5 Proxy component you need to configure new `DataSource` in `dataSource` section. Here we will use `socks5-store` as name of newly configured data source. Additionally you need to pass name of newly configured data source to `dataSourceName` property of default repository of Tigase Socks5 Proxy component.

```
dataSource {
```



```
socks5-store () {  
    uri = -'jdbc:db_server_type://server/socks5-database'  
    -}  
}  
  
socks5 () {  
    repository {  
        default () {  
            dataSourceName = -'socks5-store'  
            -}  
        -}  
        -....  
    }  
}
```

## Performance

Tigase Socks5 Proxy component was tested with 100 concurrent transfers. Maximal traffic processed by component was 21,45MB/s on loopback interface. All XMPP clients and Tigase XMPP Server used in test were running on the single machine.

---

# Chapter 20. Tigase Push Component

Tigase Team :toc: :numbered: :website: <http://tigase.net>

Welcome to Tigase Push component guide

## Tigase Push Component

Tigase Push component is a Push notifications component implementing XEP-0357: Push Notifications [<https://xmpp.org/extensions/xep-0357.html>]. It is a gateway between Push Notification services and XMPP servers. It is configured by default to run under name of `push`.

### Note

Tigase Push component requires at the minimum version 8.0.0 of Tigase XMPP Server.

Push notifications enable messages and pertinent information for clients, even if they are offline as long as they are registered with the push service. Tigase Messenger for iOS and Tigase Messenger for Android both have support for this feature.

## Workflow

The workflow for enabling and using push notifications works as follows:

### Enabling notifications

In order to receive notifications, clients will require registration with a push service. Although this process is mainly invisible to the user, the steps in registration are listed here:

- The client registers and bootstraps too it's associated push service. This is done automatically.
- The client registers itself with the push service server which then will dedicate a node for the device.
- Node information is passed back to the client and is shared with necessary components.

### Receiving notifications

Notifications sent from the server are received the following (simplified) way:

- A message is published on the XMPP node which is then sent to the push service on the same server.
- The push service will then inform the user agent (an application on the device running in the background) that a push notification has been sent.
- The user agent will then publish the notification to the client for a user to see, waking up or turning on the client if it is not running or suspended.

## Configuration

### Enabling component

Push notifications may be sent by Tigase XMPP Server with or without use of Push component. Push Component is only required if you have your own application for mobile devices for which you want to send push notifications.

This component is not loaded and enabled by default as it requires implementations of Push notifications providers and additional configuration (including credentials required to authorize to push services). Following entries will activate component:

```
push () {  
}
```

### Note

You need to enable and configure push providers implementations before it will be possible to send push notifications. For more details about this process, please check documentations of push service provider projects.

## Enabling push notifications for offline messages

Push notifications may be sent by Tigase XMPP Server with or without use of Push component. Push Component is only required if you have your own application for mobile devices for which you want to send push notifications.

If you are using existing application, then its maker have to provide you with push server (push component) which will send notification in format understandable by its application.

However, on Tigase XMPP Server side you need to enable processor which will generate notifications about offline messages sent to accounts on this server. To do so, you need to add following lines withing `sess-man` configuration block to enable `urn:xmpp:push:0` processor:

```
'sess-man' () {  
  - 'urn:xmpp:push:0' () {}  
}
```

## Enabling push notifications for messages received when all resources are AWAY/XA/DND

Push notifications may also be sent by Tigase XMPP Server when new message is received and all resources of recipient are in AWAY/XA/DND state. To enable this notifications you need to enable `urn:xmpp:push:0:ext` processor instead of default Push processor.

```
'sess-man' () {  
  - 'urn:xmpp:push:0:ext' () {}  
}
```

### Warning

This is an extended version of default processor, so be sure not to enable both of them as you may received duplicated push notifications.

As this behaviour may not be expected by users and users need a compatible XMPP client to properly handle this notifications (XMPP client needs to retrieve message history to get actual message), even with this processor enable XMPP clients need to enable push notifications and in enable element need to have `away` attribute with value of `true` as in following example"

**Enabling Push notifications for away/xa/dnd account.**

```
<iq type='set' id='x43'>
```

```
<enable xmlns='urn:xmpp:push:0' away='true' jid='push-5.client.example' node='yx'
  <x xmlns='jabber:x:data' type='submit'>
    -....
  </x>
</enable>
</iq>
```

If later on, user decides to disable notification for account in away/xa/dnd state, it may disable push notifications or once again send stanza to enable push notification but without away attribute being set:

```
<iq type='set' id='x43'>
  <enable xmlns='urn:xmpp:push:0' away='true' jid='push-5.client.example' node='yx'
    <x xmlns='jabber:x:data' type='submit'>
      -....
    </x>
  </enable>
</iq>
```

## Usage

### Sending notifications

When you will register a device for a Push Notifications, you will receive name of the PubSub node where you should publish items. Publishing items to this node, as specified in XEP-0357: Push Notifications [<https://xmpp.org/extensions/xep-0357.html>] will result in push notifications being delivered to the registered device.

### Registering device

To register a device you need to execute the adhoc command `register-device` available at Push Notification component. This command will return a form which needs to be filled.

Form consists of following fields:

provider	ID of a provider for which you want to register a device. It contains a list of available providers and you need to select a proper one.
device-token	Unique token which your application retrieved from a device or client library and which should be used to identify device you want to register for push notifications.

When you submit this form, it will be processed and will respond with a `result` type form. Within this form you will find a `node` field which will contain a PubSub node name created by the Push Notifications component, to which you should publish notification items. This returned node with `jid` of the Push Notifications Component should be passed to your XMPP server as the address of the XMPP Push Service.

### Unregistering device

To unregister a device, you need to execute the adhoc command `unregister-device` available within the Push Notification component. This command will return a form which needs to be filled out.

This form consists of the following fields:

provider	ID of a provider for which your devices was registered.
----------	---

device-token      Unique token which your application retrieved from a device or client library and was registered at this push notifications component.

When you submit this form, it will be processed and will respond with a `result` form to notify you that device was successfully unregistered from the push notifications component.

## Providers

Providers availability depends on the deployed binaries, by default Tigase includes following providers:

## Tigase Push Component - FCM provider

### Overview

Tigase Push Component - FCM provider is an implementation of FCM provider for Tigase Push Component. It allows Tigase Push Component to connect to Firebase Cloud Messaging and send notifications using this service.

## Configuration

### Enabling provider

To enable this provider, you need to enable `fcm-xmpp-api` bean within push component configuration scope.

**Example.**

```
push () {
    -'fcm-xmpp-api' () {
        # FCM configuration here
    -}
}
```

### Note

You need to pass FCM configuration parameters to make it work, see below.

### Setting FCM credentials

FCM XMPP API provider will not work properly without API key and project id as this values are required for authorization by FCM. You need to get information from FCM account.

When you have this data, you need to pass sender id as `sender-id` property and server key as `server-key` property.

**Example.**

```
push () {
    -'fcm-xmpp-api' () {
        -'sender-id' = -'your-sender-id'
        -'server-key' = -'your-server-key'
    -}
}
```

```
}
```

## Connection pool

By default this provider uses single client to server connection to FCM for sending notifications. If in your use case it is too small (as you need better performance), you should adjust value of pool-size configuration property. Setting it to 5 will open five connections to FCM for better performance.

### Example.

```
push () {  
  - 'fcm-xmpp-api' () {  
    - 'pool-size' = 5  
  -}  
}
```

## Tigase Push Component - APNs provider

### Overview

Tigase Push Component - APNs provider is an implementation of APNs provider for Tigase Push Component. It allows Tigase Push Component to connect to Apple Push Notification service and send notifications using this service.

## Configuration

### Enabling provider

To enable this provider, you need to enable apns-binary-api bean within push component configuration scope.

### Example.

```
push () {  
  - 'apns-binary-api' () {  
    # APNs configuration here  
  -}  
}
```

### Note

You need to pass APNs configuration parameters to make it work, see below.

### Setting APNs credentials

APNs binary API provider will not work properly without certificate file required for authorization by APNs and password to decrypt this certificate file. You need to get certificate using Apple Developer Account.

When you have this certificate, you need to pass path to certificate file as cert-file property and password as cert-password

**Example for /etc/apns-cert.p12 and Pa\$\$word.**

```
push () {  
  -'apns-binary-api' () {  
    -'cert-file' = -'/etc/apns-cert.p12'  
    -'cert-password' = -'Pa$$w0rd'  
  -}  
}
```

## Connection pool

By default this provider uses many connection to APNs for sending notifications which equals to number of available CPU cores. If in your use case it is too small or too big number, you can adjust it by setting value of pool-size configuration property. Setting it to 5 will make sure to use only five connections to APNs.

### Example.

```
push () {  
  -'apns-binary-api' () {  
    -'pool-size' = 5  
  -}  
}
```

---

# Chapter 21. Tigase STUN Component

Tigase Team <team@tigase.net [mailto:team@tigase.net]> :toc: :numbered: :website: http://tigase.net

Welcome to Tigase STUN component guide

## Tigase STUN Component

Tigase STUN Component allows for the use of a STUN server to handle XMPP and related communications to allow for smoother server operations behind a NAT.

### What is STUN?

STUN stands for Simple Traversal of UDP[User Datagram Protocol] Through NAT[Network Address Translators]. It allows for computers behind a NAT router to host and provide UDP information without having to create rule exceptions on the router, or provide specific information to the NAT service. When specified within Tigase, XMPP and UDP communications can be directed to a specific STUN server which will then handle incoming requests to your network. You may use a public, or your own STUN server with Tigase.

### Requirements

The only requirement (aside from configuration) is that you are operating on a network that is not a Symmetric NAT as STUN by itself will not function correctly.

## Configuration

Below is an example configuration for STUN component. Note that the 2 `stun-primary` and 2 `stun-secondary` settings are required, where external settings are not.

```
stun (class: tigase.stun.StunComponent) {  
  -'stun-primary-ip' = -'10.0.0.1'  
  -'stun-primary-port' = 3478  
  -'stun-secondary-ip' = -'10.0.0.2'  
  -'stun-secondary-port' = 7001  
  -'stun-primary-external-ip' = -'172.16.0.22'  
  -'stun-primary-external-port' = 3479  
  -'stun-secondary-external-ip' = -'172.16.0.23'  
  -'stun-secondary-external-port' = 7002  
}
```

### Note

Primary port should be set to 3478 as it is default port for STUN servers.

### Setting descriptions

1. `stun-primary-ip` - primary IP address of STUN server used for binding (and sending to client if `stun-primary-external-ip`)



2. `stun-primary-port` - primary port of STUN server used for binding (and sending to client if `stun-primary-external-port`)
3. `stun-secondary-ip` - secondary IP address of STUN server used for binding (and sending to client if `stun-secondary-external-ip`)
4. `stun-secondary-port` - secondary port of STUN server used for binding (and sending to client if `stun-secondary-external-port`)

If you wish to have a secondary STUN server as a backup, or to provide multiple addresses for STUN services, the following may be used.

1. `stun-primary-external-ip` - primary external IP address of STUN server used for sending to client if set
2. `stun-primary-external-port` - primary external port of STUN server used for sending to client if set
3. `stun-secondary-external-ip` - secondary external IP address of STUN server used for sending to client if set
4. `stun-secondary-external-port` - secondary external port of STUN server used for sending to client if set

## Logback configuration

You may want to use logback for STUN server to append normal server logs. To do this, specify the logback xml file within java options in the `tigase.conf` file.

```
JAVA_OPTIONS="-Dlogback.configurationFile=etc/logback.xml"
```

You may configure the logback by editing the xml included with distributions at `logback.xml`.

What is included is a basic logback configuration that just adds the stun logging.

```
<configuration scan="true">

  <appender name="STDOUT"
    class="ch.qos.logback.core.ConsoleAppender">
    <encoder>
      <pattern>
        %d{HH:mm:ss.SSS} [%thread] %-5level %logger{36} -- %msg%n
      </pattern>
    </encoder>
  </appender>

  <logger name="de.javawi.jstun.header.MessageHeader" level="INFO" -/>

  <root level="DEBUG">
    <appender-ref ref="STDOUT" -/>
  </root>

</configuration>
```

---

# Chapter 22. Tigase SPAM Filter

Tigase Team <team@tigase.net [mailto:team@tigase.net]> :toc: :toclevels: 3 :numbered: :website: http://tigase.net :Date: 2017-04-09

Welcome to Tigase SPAM Filter guide.

## Overview

This Tigase SPAM Filter project contains additional features provided for Tigase XMPP Server to reduce number of sent/received SPAM messages.

## Configuration

To enable default set of SPAM filters with default settings you need to enable SessionManager processor spam-filter:

**Enabling default SPAM filters.**

```
'sess-man' () {  
    - 'spam-filter' () {}  
}
```

## Changing active SPAM filters

You can configure active SPAM filters by setting enabling and disabling SPAM filters (subbeans of spam-filter processor bean).

**Enabling message-same-long-body filter.**

```
'sess-man' () {  
    - 'spam-filter' () {  
        - 'message-same-long-body' () {}  
    -}  
}
```

## Sending error when packet is dropped

By default, due to nature of SPAM, you do not want to send error packet when SPAM packet is dropped as sending error back will:

- increase traffic on a server (which in rare cases may lead to overload of a XMPP server)
- notify spammer that it was not possible to delivery message

It is possible to configure spam-filter to send error back, by setting true to spam-filter return-error property:

**Allow sending error.**

```
'sess-man' () {  
    - 'spam-filter' () {  
        return-error = true
```

```
    -}  
}
```

## Enabling logging of dropped messages

It is possible to enable logging of dropped messages by adding `spam` to comma separate list of values for `--debug` property.

```
--debug=spam
```

## Filters

In this section there is a list of available filters and detailed description of each filtering algorithm.

### Same long message body

When there is a SPAM being sent using XMPP server in most cases number of messages with longer body size increases and in most cases every SPAM message contains same body part. This filter is identified by following id `message-same-long-body`.

Detection is based on:

- message body being longer than particular value
- multiple messages being sent with same long body

Below is list of possible settings which may be modified to adjust this filter behaviour.

### Message body length

SPAM messages are usually longer messages (over 100 chars). To reduce overhead of filtering and memory required for filtering we check length of message body and process it further only if message exceeds declared message body length (*default: 100 chars*).

You can also check messages with smaller body (ie. only 50 chars) by setting `body-size` property to 50.

**Setting filter to check message with body bigger than 50 chars.**

```
'sess-man' () {  
  - 'spam-filter' () {  
    - 'message-same-long-body' () {  
      - 'body-size' = 50  
    -}  
  -}  
}
```

### Number of allowed message with same body

In most cases message with same body is sent to multiple users. Filter will count messages with same body (which is bigger than declared message body length) and if it exceeds message number limit then any further message with same body will be detected and marked as SPAM. By default we allow 20 messages with same body to be processed by SessionManager. If you wish to change this limit set `number-limit` to appropriate value.

**Setting number of allowed message to 10.**

```
'sess-man' () {
  -'spam-filter' () {
    -'message-same-long-body' () {
      -'number-limit' = 10
    -}
  -}
}
```

## Size of counters cache

We process every message and for every body of message which body length exceeds body length limit we need to keep counter. These counters are kept in cache which size is configurable and by default equals 10000. To change size of counters cache assign proper value to `counter-size-limit`.

**Increasing cache size to 1000000.**

```
'sess-man' () {
  -'spam-filter' () {
    -'message-same-long-body' () {
      -'counter-size-limit' = 1000000
    -}
  -}
}
```

## Error message and missing `<error/>` child

Some of SPAM messages are sent as stanzas which are invalid if we compare them with XMPP specification, ie. `<message/>` stanza with `type` attribute set to `error` are sent without child element `<error/>` which is required for all packets of type `error`. This filter detects this kind of messages and marks them as SPAM.

This filter is identified by following id `message-error-ensure-error-child`.

## Groupchat messages sent to bare JID

In some cases SPAM messages are being sent as groupchat messages (messages with `type` attribute set to `groupchat`). With this type of messages we cannot use filtering based on number of message sent with same body as in case of MUC messages we must accept a lot of messages with same body, because there may be many users which are participants of same MUC room and should receive same message.

To address this issue we decided to drop all groupchat messages which are sent to our server XMPP users with `to` attribute set to bare jid, as real MUC component is aware of user resources which joined particular room and will send messages only to this particular resource by addressing message with full jid. This filter is identified by following id `muc-message-ensure-to-full-jid`.

## Known spammers

To deal with spam it is required to filter every message to verify if it is spam or not. Usually spammers are using same accounts to send bigger number of messages. This filter takes it as an advantage of this to reduce time required for filtering spam messages as when any other filter marks message as spam this filter will be notified and will mark sender's jid as a spammer. This will result in a ban for any packet exchange with this user for configured ban time.

If user will send a burst of spam messages then he will be banned for configured ban time for every spam message, ie. if user would send 20 messages and ban time will be set to 15 minutes then users will be banned for 300 minutes (5 hours).

This filter is identified by following id known-spammers.

<ban-time>

<title>Ban time</title>

Time in minutes for which user marked as spammer will not be able to exchange packets with any other users. By default this value is set to 15 minutes and if you would like to increase it to 30 minutes just add following line to `etc/init.properties` file:

```
'sess-man' () {  
    -'spam-filter' () {  
        -'known-spammers' () {  
            ban-time = 30  
        -}  
    -}  
}
```

</ban-time>

## Cache time

Time in minutes for which user will be remembered as a spammer. It will be able to exchange messages with other users (after ban time passes), but if the situation repeats within this time and our algorithm will be sure that user is a spammer - it may disable local user account.

```
'sess-man' () {  
    -'spam-filter' () {  
        -'known-spammers' () {  
            cache-time = 10080  
        -}  
    -}  
}
```

## Disabling account

If filter, depending on other filter reports, will establish that user is for sure a spammer it may not only ban user for some time, but it may disable that user account. This is done by default, if you wish to disable account deactivation add following line to `etc/init.properties` file:

```
'sess-man' () {  
    -'spam-filter' () {  
        -'known-spammers' () {  
            disable-account = false  
        -}  
    -}  
}
```

## Print list of detected spammers

It is possible to request filter to print full list of known spammer which are currently banned every minute. To do so, you need to set `print-spammers` property to `true`.

```
'sess-man' () {
  -'spam-filter' () {
    -'known-spammers' () {
      print-spammers = true
    -}
  -}
}
```

## Frequency of printing list of spammers

By default, list of detected spammers is printed to logs every day. If you wish you can adjust this value to 1 hour, then add following entry to `etc/init.properties` file:

```
'sess-man' () {
  -'spam-filter' () {
    -'known-spammers' () {
      print-spammers-frequency = 60
    -}
  -}
}
```

## Presence subscription filter

When there is a presence-based SPAM being sent using XMPP server in most cases there is a lot of presence of type subscribe being sent from the single JID. This behavior is annoying and has negative impact on the XMPP server as according to the XMPP specification each presence of type subscribe sent from JID which is not in the users roster causes adding this JID to the user's roster until user declines subscription request.

Detection is based on counting subscription request being sent from the same bare JID within a period of time.

Below is list of possible settings which may be modified to adjust this filter behaviour.

## Number of allowed subscription requests per minute

By default filter allows 5 subscription requests to be sent from the single JID per minute. If some client will send more than 5 subscription requests it will be marked as a spammer.

**Setting filter to allow 7 subscription requests per minute.**

```
'sess-man' () {
  -'spam-filter' () {
    -'presence-subscribe' () {
      -'limit-per-minute' = 7
    -}
  -}
}
```