initMAX

ZABBIX PREMIUM PARTNER

ZABBIX CERTIFIED TRAINER

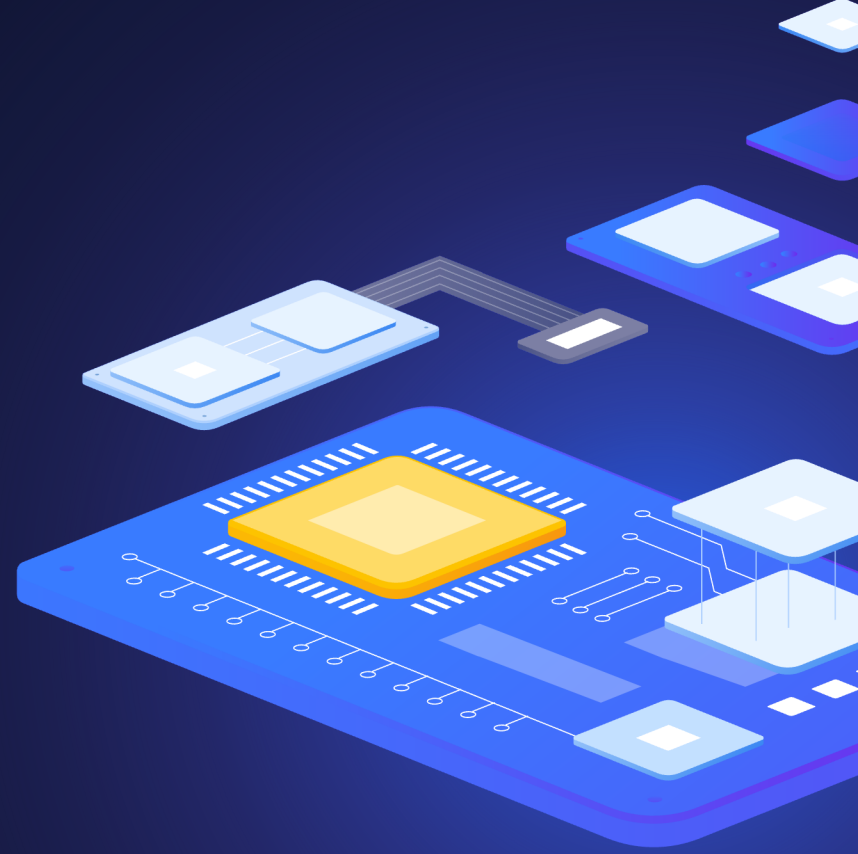
Webinar

Zabbix Java Gateway: Installation, Tips and Monitoring Tomcat and WildFly

all your microphones are muted

ask your questions in Q&A, not in the Chat

use Chat for discussion, networking or applause



1

Introduction to Zabbix Java Gateway

initMAX

Introduction to Zabbix Java Gateway

What is Zabbix Java Gateway?

- **Standalone component** of Zabbix for monitoring Java applications using **JMX** (Java Management Extensions).
- Enables Zabbix to **collect metrics** directly from Java applications.
- Must be connected to a **Zabbix Server** or **Zabbix Proxy**.
- **Written in Java** (requires Java Runtime Environment).

initMAX

Introduction to Zabbix Java Gateway

Deployment Considerations

- **Only one Java Gateway** per Zabbix Server or Proxy.
  - For multiple Java Gateways, deploy additional **Zabbix Proxies** (active/passive).
- **Communication between Zabbix components** and Java Gateway **cannot be encrypted**.
  - As of version 7.4, it's the last remaining internal Zabbix communication channel without encryption.
- Communication from Java Gateway to monitored Java applications **can be secured** (encrypted and/or authenticated).
- **Version alignment** is critical—Zabbix Java Gateway **must match the Zabbix server/proxy version**.
- Don't forget to explicitly define **allowed IP addresses** in the Java Gateway configuration (zabbix_java_gateway.conf).

initMAX

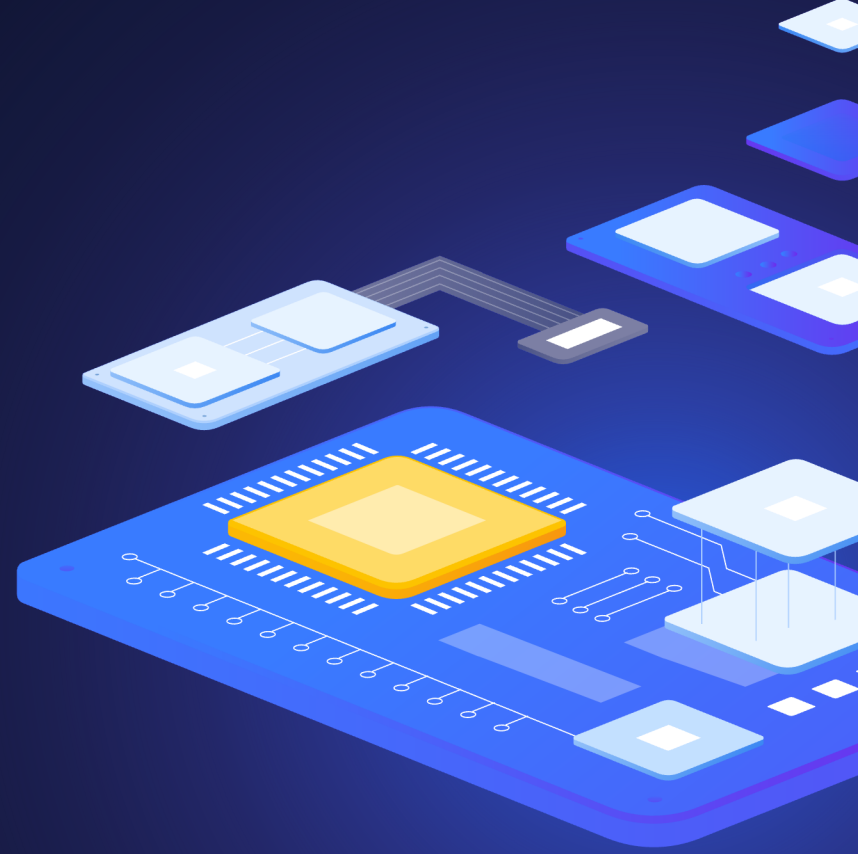
Introduction to Zabbix Java Gateway

Installation & Best Practices

- **Java Runtime Environment (JRE)** is automatically installed alongside Zabbix Java Gateway (it is Java-based).
- **No dedicated template** provided by Zabbix for the Java Gateway itself—use the generic Java template.
- **Custom libraries (JAR files)** might be needed for specific Java servers, typically located in the application's bin directory.
- Proper configuration of **Java pollers** is crucial—settings must be aligned on both:
  - Zabbix Server/Proxy (**Java pollers count**)
  - Zabbix Java Gateway (**start pollers**)
- **Recommended practice:** Deploy Zabbix Java Gateway alongside Zabbix Server or Proxy on the same machine for enhanced security.

2

Installation and self monitoring



initMAX

Introduction to Zabbix Java Gateway

Installation and self monitoring

- Add official Zabbix repository:

```
rpm -Uvh https://repo.zabbix.com/zabbix/7.0/rocky/9/x86_64/zabbix-release-latest-7.0.el9.noarch.rpm
```

- Clean DNF cache:

```
dnf clean all
```

- Install Java Gateway:

```
dnf install zabbix-java-gateway
```

initMAX

Introduction to Zabbix Java Gateway

Configuration Files

- **Zabbix Java Gateway:**
  - zabbix_java_gateway.conf – main configuration
  - zabbix_java_gateway_logback.xml – logging configuration
- **Zabbix Server:**
  - zabbix_server.conf
- **Zabbix Proxy:**
  - zabbix_proxy.conf

initMAX

Introduction to Zabbix Java Gateway

Enable Java Gateway in Zabbix

- After installation, enable Java monitoring in your **server** or **proxy** by editing these parameters:
- **JavaGateway**: IP or DNS of Zabbix Java Gateway
  - Recommended setting (security best practice):

```
JavaGateway=127.0.0.1
```

- **JavaGatewayPort**: communication port
  - Default is recommended:

```
JavaGatewayPort=10052
```

  - (Not registered in IANA, may trigger IDS/IPS alerts if changed)

initMAX

Introduction to Zabbix Java Gateway

Enable Java Gateway in Zabbix

- **StartJavaPollers**: number of Java pollers started on Zabbix server/proxy
  - Default is disabled (0), Java monitoring off
  - Recommended starting value: 5 pollers
  - If changed, align this number also in Java Gateway configuration!
  - When using multiple Zabbix servers/proxies with one gateway, sum their poller counts:
  - Example: server1 has 5 pollers, server2 has 10 → set Java Gateway pollers to **15**.

```
StartJavaPollers=5
```

initMAX

Introduction to Zabbix Java Gateway

Important Java Gateway parameters

- After configuration, restart your Zabbix components. But wait—Java Gateway itself also needs monitoring!
- Critical configuration parameters (zabbix_java_gateway.conf):

```
LISTEN_IP="0.0.0.0"
LISTEN_PORT=10052
PID_FILE="/var/run/zabbix/zabbix_java_gateway.pid"
START_POLLERS=5
TIMEOUT=3
PROPERTIES_FILE=
```

- Special monitoring section (self-monitoring):
  Uncomment the following line to enable built-in JMX monitoring of the Gateway itself (JAVA_OPTIONS):

```
JAVA_OPTIONS="$JAVA_OPTIONS -Dcom.sun.management.jmxremote -Dcom.sun.management.jmxremote.port=12345
-Dcom.sun.management.jmxremote.authenticate=false -Dcom.sun.management.jmxremote.ssl=false -
Dcom.sun.management.jmxremote.registry.ssl=false"
```

- **Important:** Ensure each Java option (-Dcom...) starts immediately after space, with **no extra spaces inside**!

initMAX

Introduction to Zabbix Java Gateway

Verifying and monitoring setup

After uncommenting and configuring **monitoring lines, restart:**

- Zabbix Java Gateway
- Zabbix Server or Proxy (if configuration changed)

Add or edit your host in Zabbix frontend:

- Create a host for Zabbix Java Gateway itself or use an existing host.
- Add a new JMX-type interface.
- Apply built-in template: **Generic Java JMX**.

Monitoring status:

- After short delay, JMX icon turns **green**.
- Java Gateway is now ready and monitored.

initMAX

Introduction to Zabbix Java Gateway

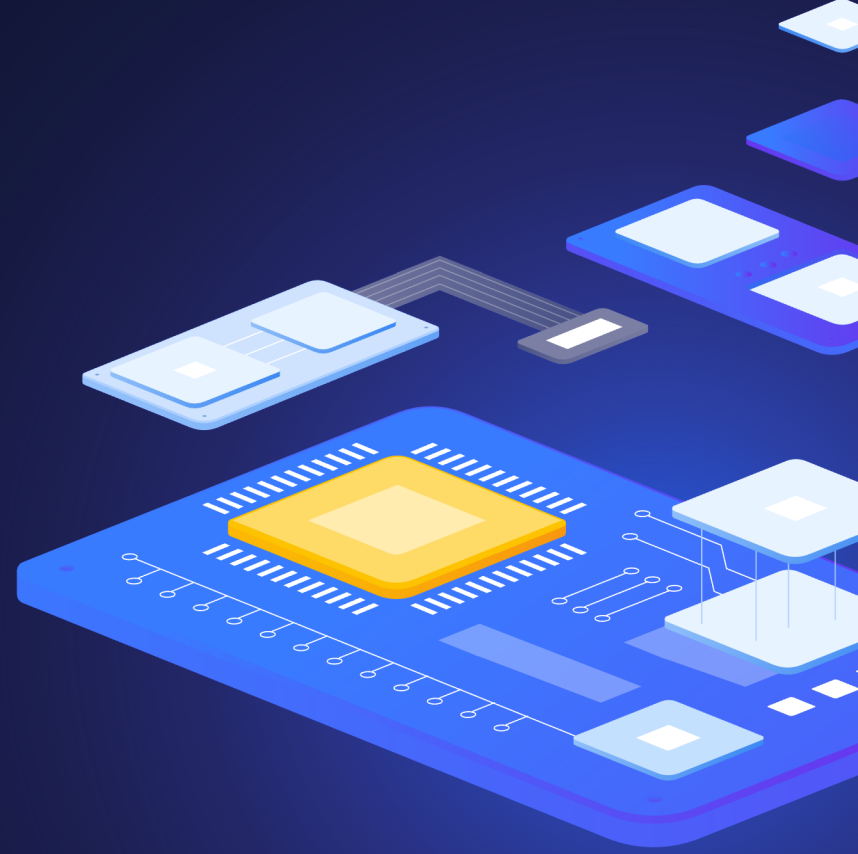
Why is this important?

- Zabbix by default monitors only Java poller availability.
- Additional self-monitoring via JMX ensures your Java Gateway is operational and detects common issues early, especially Java heap memory problems:

```
java.lang.OutOfMemoryError: Java heap space
```

3

Java keys and endpoints



initMAX

Java Keys – Introduction

Using Zabbix Java Gateway for Java Monitoring

Zabbix provides specialized keys for JMX monitoring:

- **jmx**: Standard JMX query to obtain specific values.
- **jmx.get**: Recommended method for **low-level discovery (LLD)**.
- **jmx.discovery**: Legacy method, **no longer recommended**.

- **Example of standard jmx key**:

```
jmx["Catalina:type=DataSource,host=localhost,context=/dbcheckapp,class=javax.sql.DataSource,name=\"jdbc/mydb\"","idle"]
```

initMAX

Java Keys – Discovery Examples

Using Zabbix Java Gateway for Java Monitoring

Recommended method (jmx.get):

- **jmx.get**: Recommended method for **low-level discovery (LLD)**.

```
jmx.get[beans,"Catalina:type=DataSource,host=localhost,context=*,class=javax.sql.DataSource,name=*"]
```

- **Legacy method (jmx.discovery, not recommended)**:

```
jmx.discovery[beans,"Catalina:type=Manager,host=*,context=*"]
```

- **Reason for recommendation:**
- jmx.get is more flexible, reliable, and easier for ongoing maintenance.

initMAX

Java Endpoint – URL Structure Explained

General syntax for JMX endpoints:

```
service:jmx:rmi:///jndi/rmi://<host>:<port>/jmxrmi
```

Explanation of each part:

- **service:** Specifies a service endpoint.
- **jmx:** Indicates Java Management Extensions service.
- **rmi:** Remote Method Invocation protocol **(default for JMX).**
- **/// (triple slash):** Separates the protocol specification from the object path.
- **jndi:** Java Naming and Directory Interface (JNDI), helps locate remote objects.
- **rmi://<host>:<port>:** Specifies host and port of the JMX service.
- **/jmxrmi**: Standard object name provided by Java applications.

initMAX

Java Endpoint – Practical Examples

Apache Tomcat example:

```
service:jmx:rmi:///jndi/rmi://tomcat-server:9004/jmxrmi
```

WildFly example (default setup):

```
service:jmx:remote+http://wildfly-server:9990
```

Default Java application example:

```
service:jmx:rmi:///jndi/rmi://localhost:12345/jmxrmi
```

For more details, please refer to the official documentation of your application server.

initMAX

Java Keys & Endpoints – Tips and Best Practices

- Always prefer **jmx.get** over jmx.discovery for discovery (LLD).
- Ensure **endpoint definitions match exactly** between Java applications and Zabbix configuration (port, protocol, security).
- When possible, enable **authentication and/or encryption** for endpoints.
- Regularly check Java Gateway logs for clear troubleshooting insights.

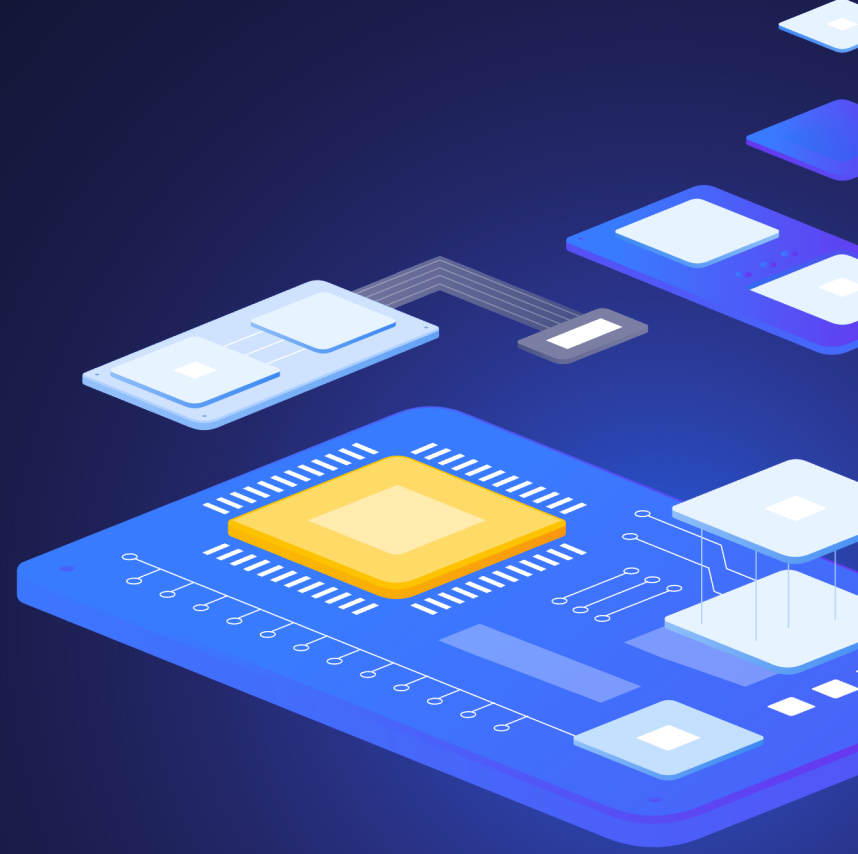
- **Typical error message:**

```
java.io.IOException: Failed to retrieve RMIServer stub
```

(Usually indicates network connectivity or configuration issues, like incorrect port or firewall restrictions.)

4

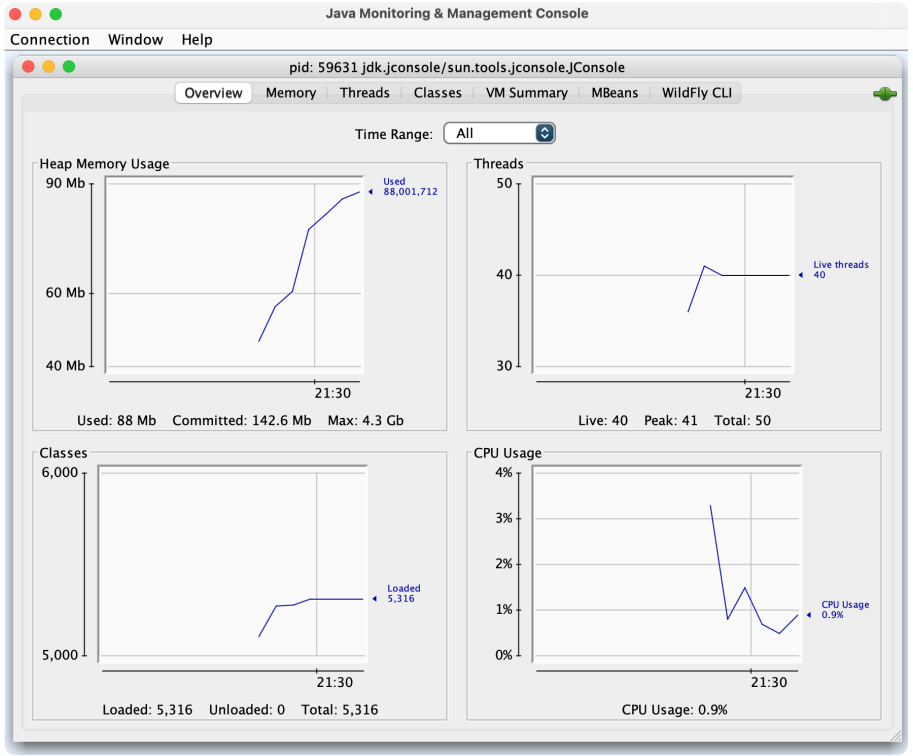
Java helpers



initMAX

Java Helpers – Overview

- JConsole:
  - Graphical interface for easy JMX monitoring
  - Simple and intuitive
  - Part of the Java Development Kit



- Jmxterm:
  - Command-line interface (CLI) for advanced JMX interaction
  - Flexible and script-friendly

```
wget https://github.com/jiaqi/jmxterm/releases/download/v1.0.4/jmxterm-1.0.4-uber.jar
```

```
root@student-postgresql-02 ~ # java -jar jmxterm-1.0.4-uber.jar
Delete /root/.jmxterm_history if you encounter error right after launching me.
Welcome to JMX terminal. Type "help" for available commands.
$>open localhost:9010 -u monitorRole -p Password1
#Connection to localhost:9010 is opened
$>domains
#following domains are available
Catalina
JMImplementation
Users
com.sun.management
java.lang
java.nio
java.util.logging
jdk.management.jfr
tomcat.jdbc
$>
```

initMAX

JConsole – Quick Introduction

Purpose:

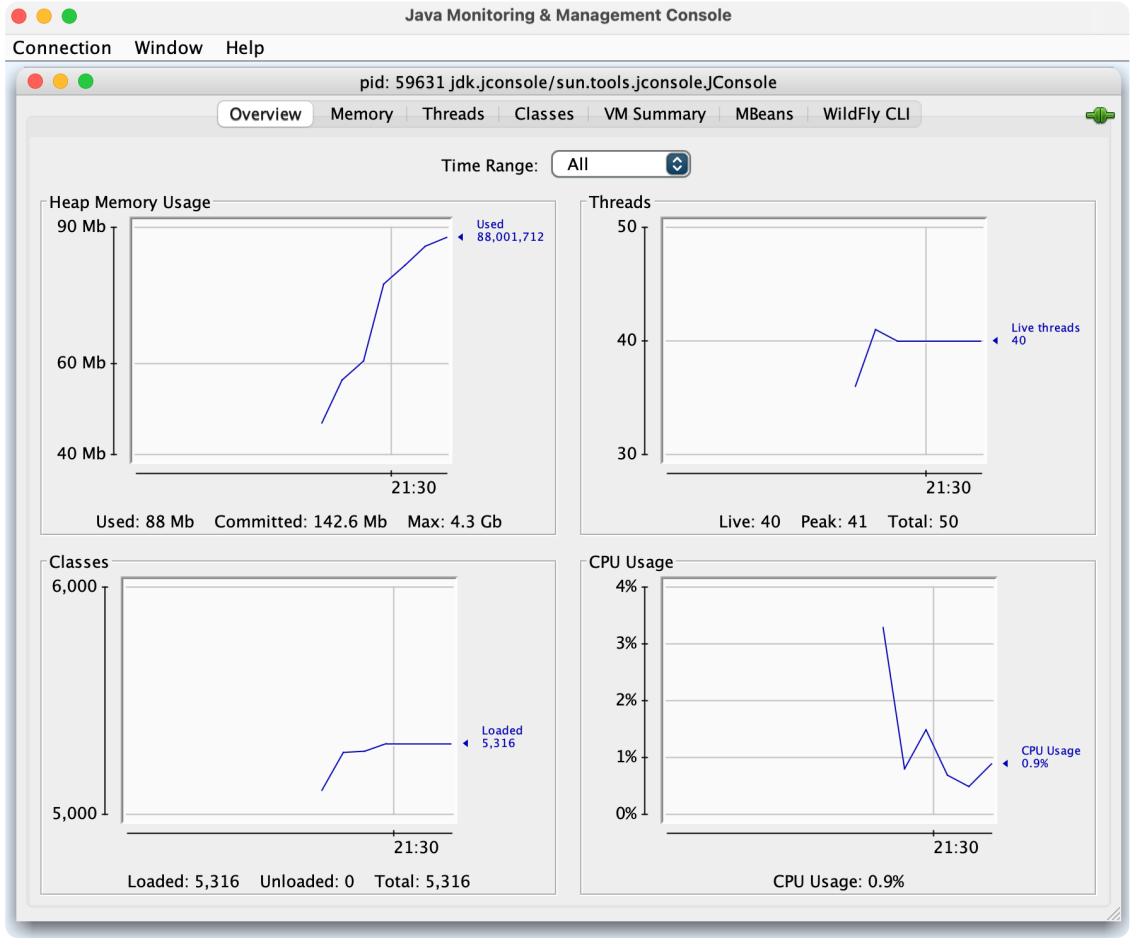
- Graphical Java management and monitoring tool.

Benefits:

- Easy-to-use GUI.
- Live metrics overview: heap memory, threads, CPU, loaded classes.
- Quick access to JMX Beans for troubleshooting.

Typical use cases:

- Initial verification of JMX configuration.
- Checking real-time performance and resource consumption.



initMAX

Jmxterm – Quick Introduction

Purpose:

- Command-line tool to interact with JMX beans.

Benefits:

- Ideal for scripting and automation.
- Flexible, precise querying of specific MBeans.

Typical use cases:

- Troubleshooting complex JMX queries.
- Automation and validation in deployment scripts.

Download:

- github.com/jiaqi/jmxterm

```
$>get -b java.lang:type=Memory HeapMemoryUsage
#mbean = java.lang:type=Memory:
HeapMemoryUsage = {
  committed = 52428800;
  init = 60817408;
  max = 960495616;
  used = 37248656;
 };

$>
```

initMAX

Comparison – JConsole vs Jmxterm

Feature	JConsole	Jmxterm
Interface:	GUI	CLI
Complexity:	User-friendly, simple	Advanced, technical
Installation:	Bundled with JDK	Separate download
Best for:	Initial checks, quick monitoring	Automation, advanced debugging

Recommendation:

- **JConsole** for quick checks and beginners.
- **Jmxterm** for advanced troubleshooting and scripting.

initMAX

Common Issues & Troubleshooting Tips

Common problems when using JConsole or Jmxterm:

- Missing libraries causing incomplete metrics visibility.
- Connection issues due to firewall restrictions or incorrect port configuration.

Recommendations:

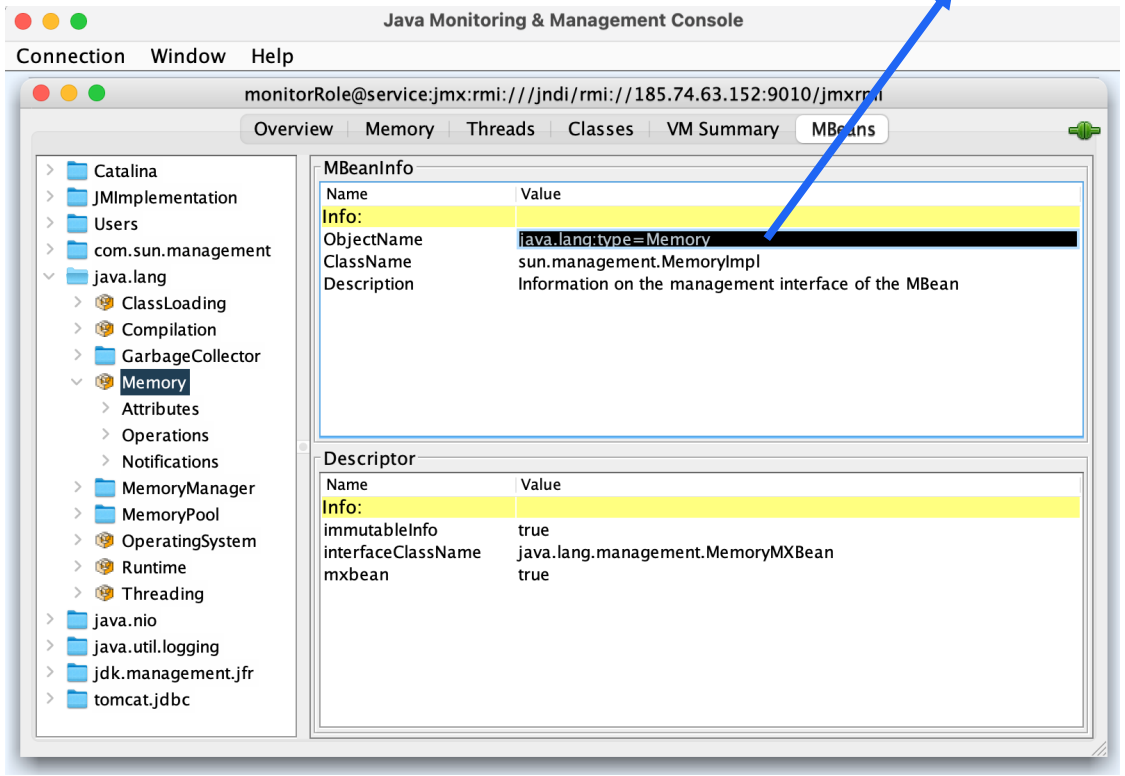
- Always verify connectivity first using simple tools (telnet, nc).
- Ensure necessary Java libraries (.jar) are included in classpath for JConsole/Jmxterm if required by the monitored application.

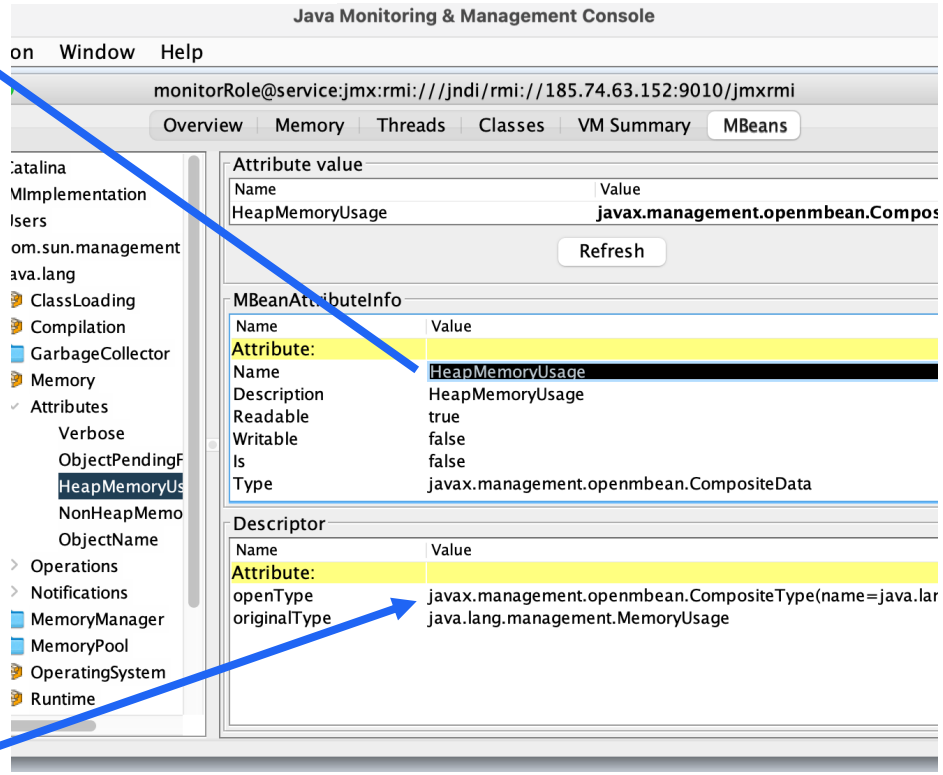
initMAX

How to use it?

Type: JMX agent

* Key: jmx["java.lang:type=Memory","HeapMemoryUsage.used"]

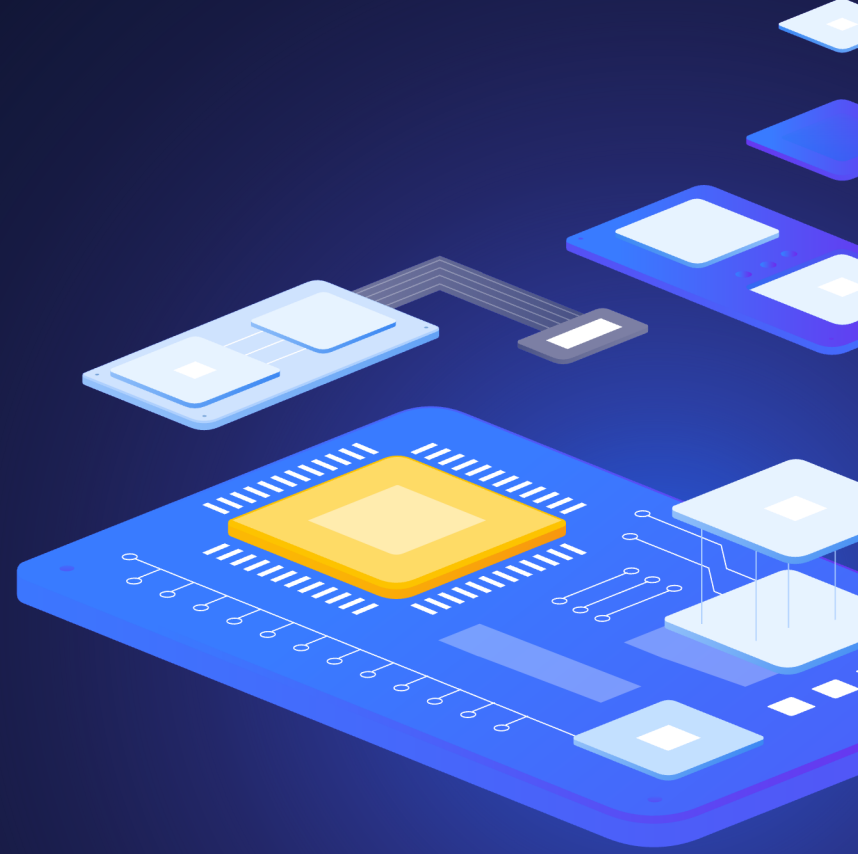





```
javax.management.openmbean.CompositeType(name=java.lang.management.MemoryUsage,items=((itemName=committed,itemType=javax.management.openmbean.SimpleType(name=java.lang.
Long)),(itemName=init,itemType=javax.management.openmbean.SimpleType(name=java.lang.Long)),(itemName=max,itemType=javax.management.openmbean.SimpleType(name=java.lang.L
ong)),(itemName=used,itemType=javax.management.openmbean.SimpleType(name=java.lang.Long))))
```

5

Tomcat monitoring



initMAX

Tomcat Monitoring – Installation & Initial Setup

Default installation (Rocky Linux):

```
dnf install tomcat tomcat-webapps tomcat-admin-webapps tomcat-docs-webapp
```

Enable JMX monitoring with authentication:

- Edit /etc/tomcat/tomcat.conf and add: (Don't forget to change your IP – monitored host not Zabbix!)

```
JAVA_OPTS="-Dcom.sun.management.jmxremote \
-Dcom.sun.management.jmxremote.port=9010 \
-Dcom.sun.management.jmxremote.rmi.port=9010 \
-Dcom.sun.management.jmxremote.ssl=false \
-Dcom.sun.management.jmxremote.authenticate=true \
-Dcom.sun.management.jmxremote.password.file=/etc/tomcat/jmxremote.password \
-Dcom.sun.management.jmxremote.access.file=/etc/tomcat/jmxremote.access \
-Djava.rmi.server.hostname=185.74.63.152 \
-Djava.net.preferIPv4Stack=true"
```

initMAX

JMX Authentication Configuration

Create JMX access roles and passwords:

- Edit or create /etc/tomcat/jmxremote.access:

```
monitorRole readonly
controlRole readwrite
```

- Edit or create /etc/tomcat/jmxremote.password:

```
monitorRole Password1
controlRole Password2
```

- Set correct permissions for security:

```
chmod 600 /etc/tomcat/jmxremote.*
chown tomcat:tomcat /etc/tomcat/jmxremote.*
```

initMAX

Finalizing JMX Setup & Firewall

Restart Tomcat service:

```
systemctl restart tomcat.service
```

- Open firewall port (9010/tcp):

```
firewall-cmd --permanent --add-port=9010/tcp
firewall-cmd --reload
```

Test your JMX connection:

- JMX Endpoint:

```
service:jmx:rmi:///jndi/rmi://185.74.63.152:9010/jmxrmi
```

Credentials:

- Username: monitorRole
- Password: Password1

initMAX

Extend Tomcat Monitoring (Database)

Deploy application and JDBC driver:

```
cp /root/dbcheckapp.war /var/lib/tomcat/webapps/
cp /root/postgresql-42.7.3.jar /usr/share/tomcat/lib/
```

- Open Correct file ownership:

```
chown tomcat:tomcat /var/lib/tomcat/webapps/dbcheckapp.war
chown tomcat:tomcat /usr/share/tomcat/lib/postgresql-42.7.3.jar
```

- Configure DB connection pool in:

```
nano /etc/tomcat/context.xml
```

initMAX

Example DB Resource Configuration

Sample configuration (context.xml):

```
<Resource name="jdbc/mydb"
          auth="Container"
          type="javax.sql.DataSource"
          factory="org.apache.tomcat.jdbc.pool.DataSourceFactory"
          driverClassName="org.postgresql.Driver"
          url="jdbc:postgresql://185.74.63.151:5432/tomcatdb"
          username="tomcat"
          password="2HDc74fE3Kdjw2gY"
          initialSize="10"
          maxActive="100"
          maxIdle="30"
          minIdle="10"
          jmxEnabled="true"
          testOnBorrow="true"
          testWhileIdle="true"
          testOnReturn="true"
          validationQuery="SELECT 1"
          validationInterval="30000"
          timeBetweenEvictionRunsMillis="30000"/>
```

initMAX

Example DB Resource Configuration

Restart Tomcat and verify:

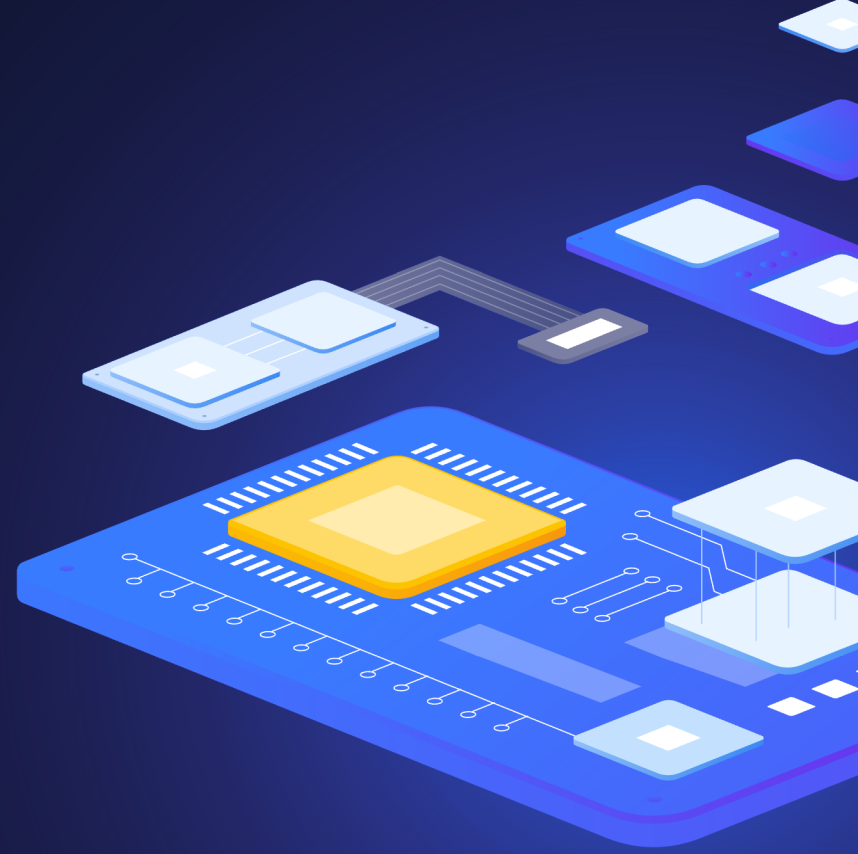
```
systemctl restart tomcat
less /usr/share/tomcat/logs/catalina.$(date '+%Y-%m-%d').log
```

- Verify via JConsole:
  - Use JMX connection details provided earlier.
- You are now ready to use the jmx.get key for discovery and standard jmx keys for monitoring your database connection pools.
- Application verification URL: (**only in our demo** – own test app for db connection)

```
http://185.74.63.152:8080/dbcheckapp/dbcheck
```

6

WildFly monitoring



initMAX

WildFly Monitoring – Initial Setup

Enable remote management

Since we are using a systemd unit file, edit the WildFly service configuration:

```
nano /etc/systemd/system/wildfly.service
```

Add or modify the following line to enable remote management: (add **-bmanagement=0.0.0.0**)

```
ExecStart=/opt/wildfly/bin/standalone.sh -c standalone-full.xml -b=0.0.0.0 -bmanagement=0.0.0.0
```

```
[Unit]
Description=WildFly Application Server
After=syslog.target network.target

[Service]
User=wildfly
Group=wildfly
ExecStart=/opt/wildfly/bin/standalone.sh -c standalone-full.xml -b=0.0.0.0 -bmanagement=0.0.0.0
ExecStop=/opt/wildfly/bin/jboss-cli.sh --connect command=:shutdown
TimeoutStartSec=300
TimeoutStopSec=30
Restart=always
RestartSec=10

[Install]
WantedBy=multi-user.target
```

initMAX

WildFly Monitoring – Initial Setup

Create management user

Run user creation script:

```
/opt/wildfly/bin/add-user.sh
```

Type: (**Example!**)

- a
- admin
- a
- Password-123
- Password-123
- Management User

```
root@student-postgresql-03 ~ # /opt/wildfly/bin/add-user.sh

What type of user do you wish to add?
 a) Management User (mgmt-users.properties)
 b) Application User (application-users.properties)
(a): a

Enter the details of the new user to add.
Using realm 'ManagementRealm' as discovered from the existing property files.
Username : admin
User 'admin' already exists and is disabled, would you like to...
 a) Update the existing user password and roles
 b) Enable the existing user
 c) Type a new username
(a): a
Password recommendations are listed below. To modify these restrictions edit the add-user.properties configuration file.
 - The password should be different from the username
 - The password should not be one of the following restricted values {root, admin, administrator}
 - The password should contain at least 8 characters, 1 alphabetic character(s), 1 digit(s), 1 non-alphanumeric symbol(s)
Password :
Re-enter Password :
What groups do you want this user to belong to? (Please enter a comma separated list, or leave blank for none)[  ]: Management User
Updated user 'admin' to file '/opt/wildfly/standalone/configuration/mgmt-users.properties'
Updated user 'admin' to file '/opt/wildfly/domain/configuration/mgmt-users.properties'
Updated user 'admin' with groups Management User to file '/opt/wildfly/standalone/configuration/mgmt-groups.properties'
Updated user 'admin' with groups Management User to file '/opt/wildfly/domain/configuration/mgmt-groups.properties'
root@student-postgresql-03 ~ #
```

initMAX

WildFly Monitoring – Verify & Firewall

Since we are modifying the systemd unit file, we must reload systemd before restarting WildFly:

```
systemctl daemon-reload
systemctl restart wildfly
```

Open firewall port for management interface:

```
firewall-cmd --permanent --add-port=9990/tcp
firewall-cmd --reload
```

Verify user via Web Management Console

- Open in browser:

```
http://185.74.63.153:9990/console/
```

JMX Endpoint URL for WildFly:

```
service:jmx:remote+http://185.74.63.153:9990
```

initMAX

Deploy JDBC Driver for WildFly

Navigate to /root and download PostgreSQL driver:

```
cd /root
wget https://jdbc.postgresql.org/download/postgresql-42.7.3.jar
```

Create module directory and move driver:

```
mkdir -p /opt/wildfly/modules/org/postgresql/main
cp postgresql-42.7.3.jar /opt/wildfly/modules/org/postgresql/main/
```

Create module configuration (module.xml):

```
nano /opt/wildfly/modules/org/postgresql/main/module.xml
```

```
<module xmlns="urn:jboss:module:1.0" name="org.postgresql">
  <resources>
    <resource-root path="postgresql-42.7.3.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

initMAX

Configure JDBC Driver via CLI

Start WildFly CLI:

```
/opt/wildfly/bin/jboss-cli.sh --connect
```

Register the JDBC driver:

```
/subsystem=datasources/jdbc-driver=postgresql:add(driver-name=postgresql,driver-module-
name=org.postgresql,driver-class-name=org.postgresql.Driver)
```

initMAX

Configure WildFly DataSource

Add PostgreSQL DataSource via CLI:

```
data-source add \
--name=PostgreSQLDS \
--jndi-name=java:/jdbc/mydb \
--driver-name=postgresql \
--connection-url=jdbc:postgresql://185.74.63.151:5432/tomcatdb \
--user-name=tomcat \
--password=2HDc74fE3Kdjw2gY \
--use-ccm=true \
--initial-pool-size=10 \
--min-pool-size=10 \
--max-pool-size=100 \
--pool-prefill=true \
--blocking-timeout-wait-millis=5000 \
--validate-on-match=true \
--background-validation=true \
--background-validation-millis=30000 \
--valid-connection-checker-class-name=org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLValidConnectionChecker \
--check-valid-connection-sql="SELECT 1"
```

Reload WildFly configuration to apply changes:

```
reload
```

initMAX

Deploy Test Application

Deploy the test application (wildflydbcheck.war):

```
cp /root/wildflydbcheck.war /opt/wildfly/standalone/deployments/
chown wildfly:wildfly /opt/wildfly/standalone/deployments/wildflydbcheck.war
chmod 644 /opt/wildfly/standalone/deployments/wildflydbcheck.war
```

Verify application via browser:

```
http://185.74.63.153:8080/wildflydbcheck/dbcheck
```

initMAX

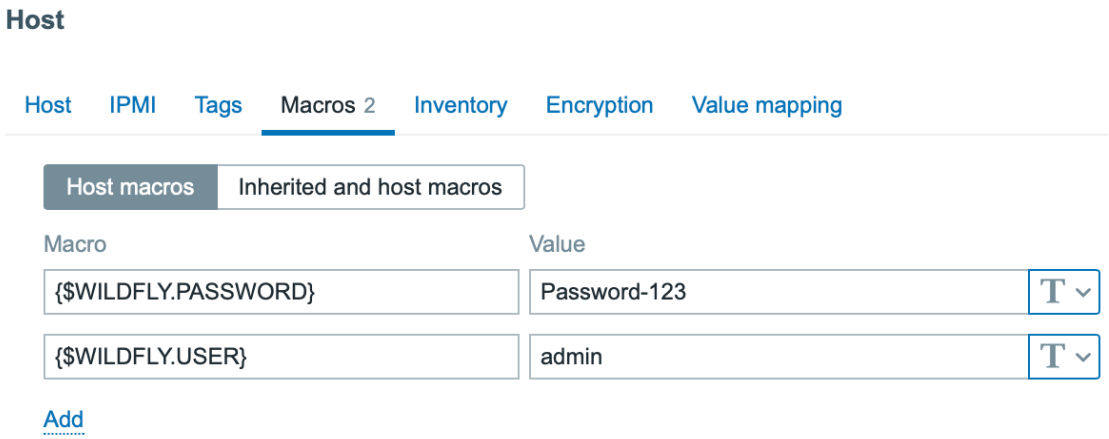
Create new WildFly host in Zabbix

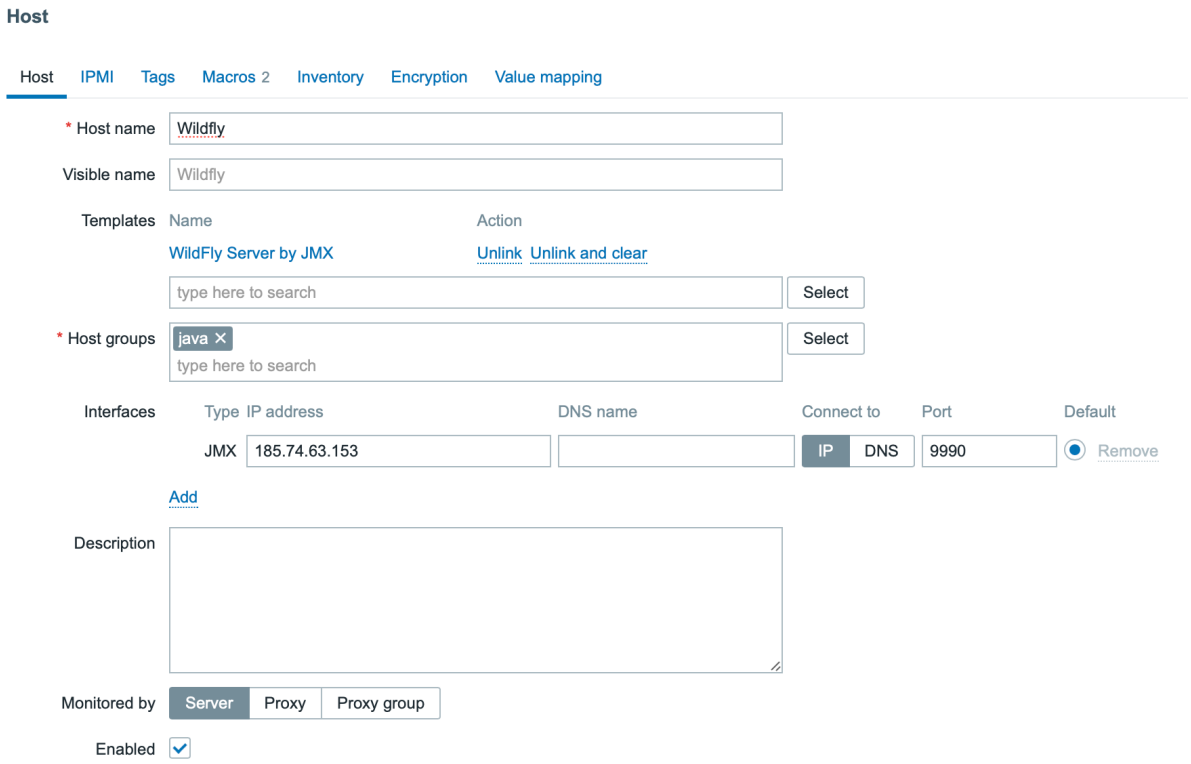
Create a new host in Zabbix using the WildFly template and adjust the macros

Set required macros:

- {$WILDFLY.USER} → **admin**
- {$WILDFLY.PASSWORD} → **Password-123**

Use correct JMX interface settings (port 9990)

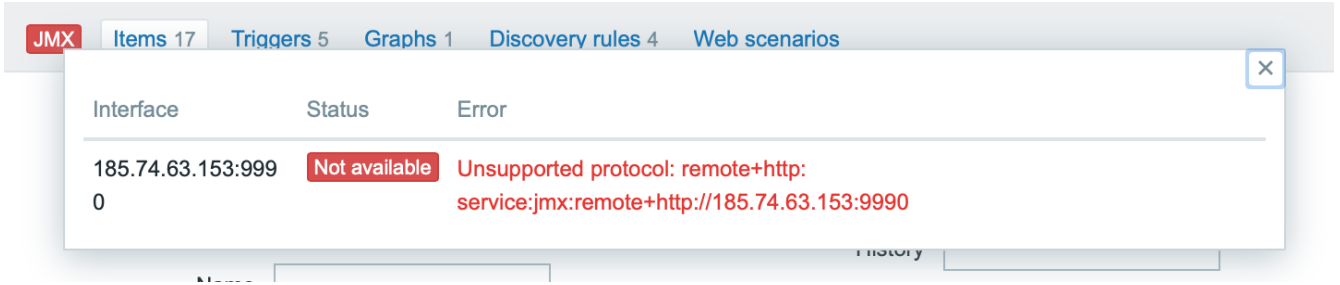




initMAX

Fix issue with missing library

If Zabbix reports the following error: (Unsupported protocol: remote+http)



It means Zabbix Java Gateway lacks the required WildFly client library.

Fix: Manually upload the WildFly client library to the Zabbix Java Gateway server:

```
cd /tmp
wget https://github.com/wildfly/wildfly/releases/download/31.0.1.Final/wildfly-31.0.1.Final.tar.gz
tar xzf wildfly-31.0.1.Final.tar.gz
cp wildfly-31.0.1.Final/bin/client/jboss-client.jar /usr/share/zabbix-java-gateway/lib/
systemctl restart zabbix-java-gateway
```

initMAX

Verify the library is loaded correctly

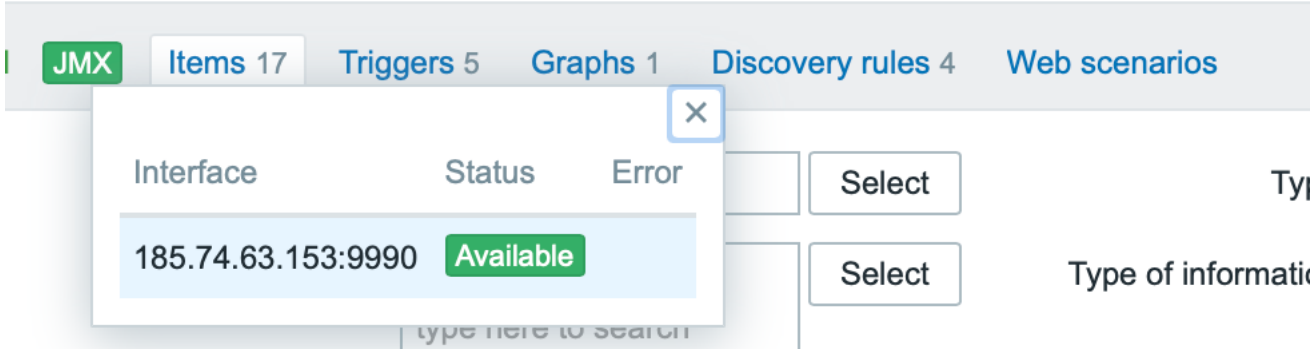
Check whether the library has loaded into Zabbix Java Gateway's process:

```
ps xauf | grep jboss
```

If you see jboss-client.jar in the Zabbix Java Gateway's process command-line, the library has been successfully loaded.

```
root@student-postgresql-01 /tmp # ps xauf | grep jboss
root       12515  0.0  0.0   6404  2176 pts/1    S+   23:23   0:00          \_ grep --color=auto jboss
zabbix     12419  2.5  2.3 3562284 88380 ?       Sl   23:22   0:01 java -server -Dlogback.configurationFile=/etc/zabbix/zabbix_java_gateway_logback.xml -classpath lib:lib/
android-json-4.3_r3.1.jar:lib/jboss-client.jar:lib/logback-classic-1.5.16.jar:lib/logback-core-1.5.16.jar:lib/slf4j-api-2.0.16.jar:bin/zabbix-java-gateway-7.0.13.jar -Dzab
bix.pidFile=/var/run/zabbix/zabbix_java_gateway.pid -Dsun.rmi.transport.tcp.responseTimeout=3000 com.zabbix.gateway.JavaGateway
root@student-postgresql-01 /tmp #
```

Now Zabbix should show the JMX interface as "Available" (green icon).



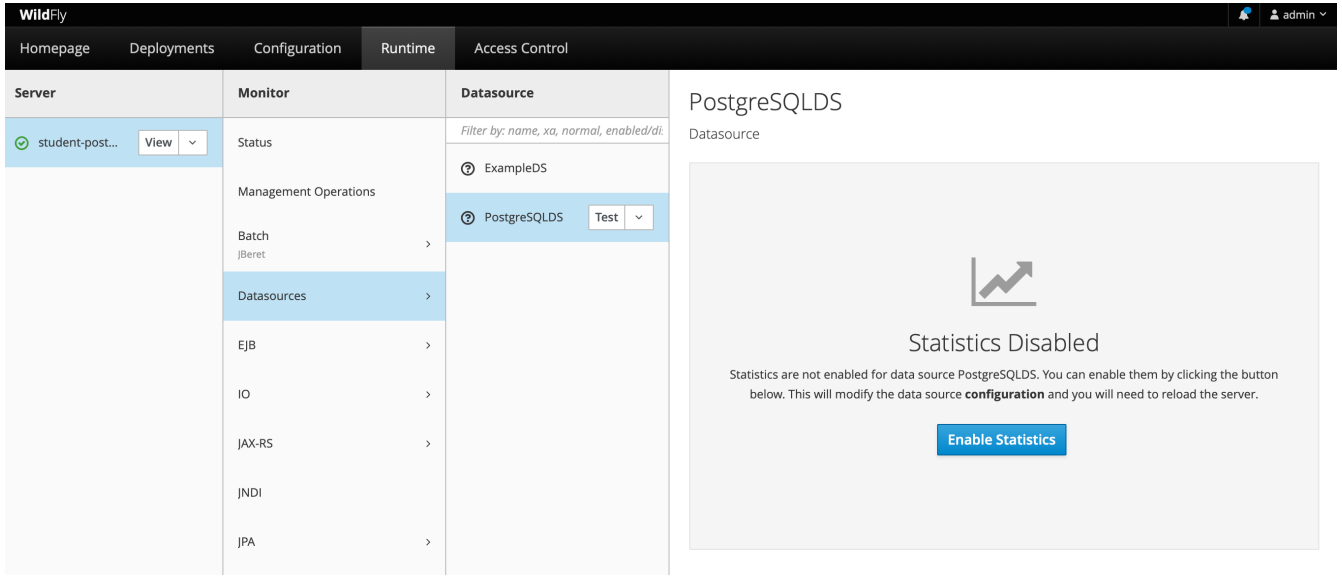
initMAX

Enable WildFly pool statistics

If Zabbix shows a warning like:

```
Pools monitoring statistic is not enabled
```

it means database connection pool statistics are disabled on your WildFly server.



To fix this:

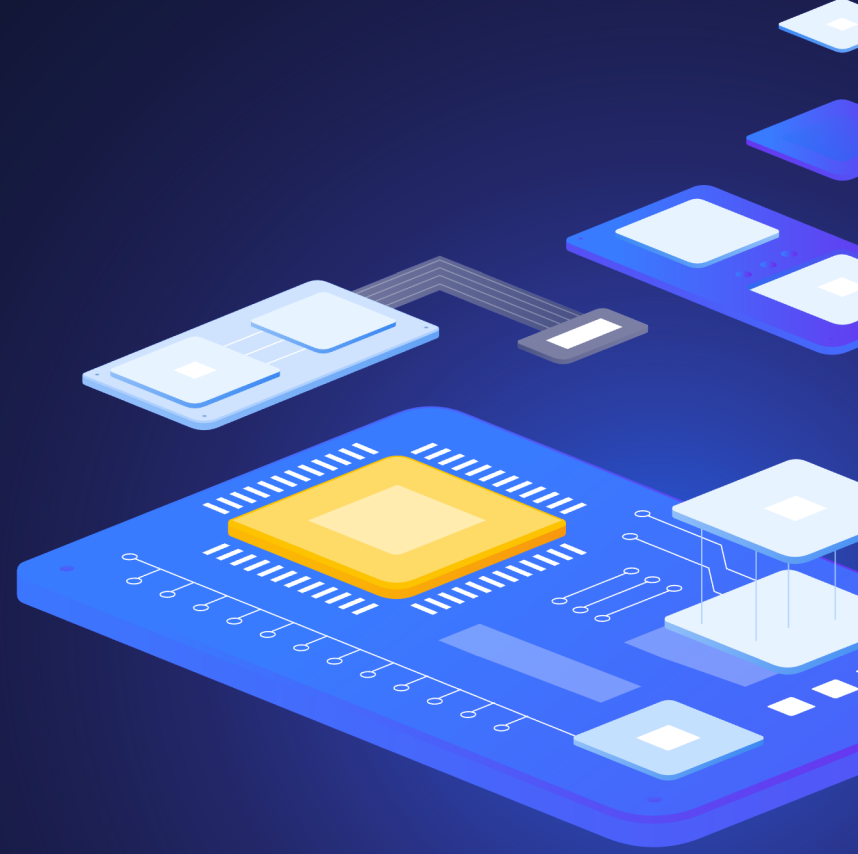
- Log into **WildFly management web interface**
- Navigate to **Runtime → Datasources**
- Select your datasource (e.g., **PostgreSQLDS**)
- Click the button **“Enable Statistics”**

After enabling statistics:

- Check again in the Zabbix frontend. The warning should disappear, and Zabbix will now collect database connection pool metrics correctly.
- You can verify metrics and graphs within Zabbix for your WildFly datasource to ensure everything is functioning as expected.

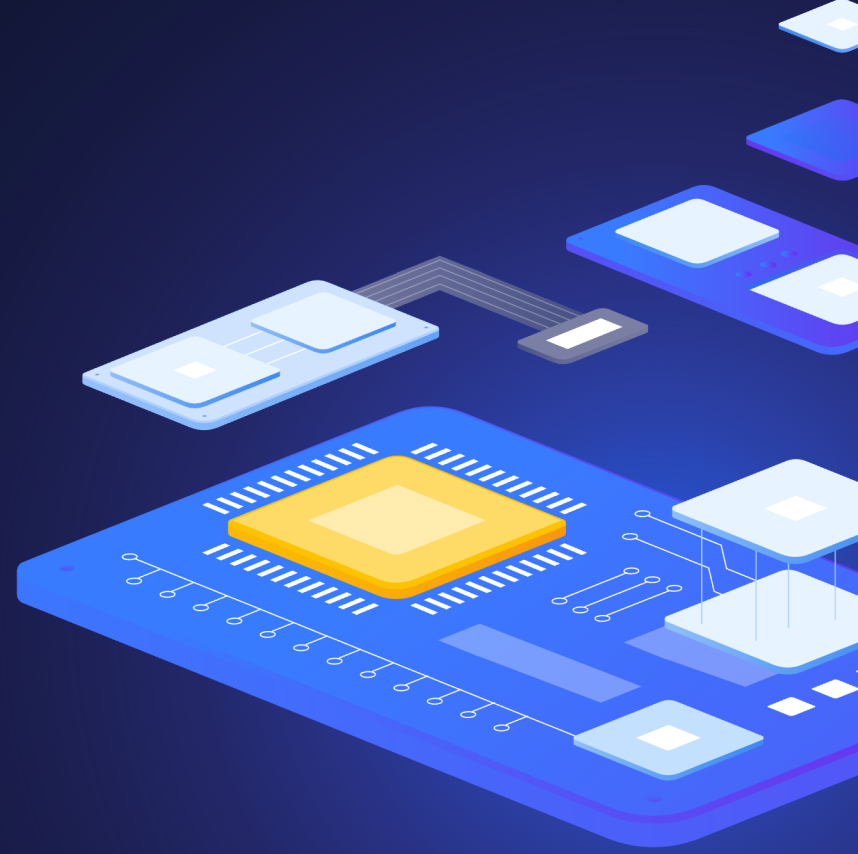
7

Demonstration



initMAX

Questions?



initMAX

Contact us:

Phone:	+420 800 244 442
Web:	https://www.initmax.cz
Email:	tomas.hermanek@initmax.cz
LinkedIn:	https://www.linkedin.com/company/initmax
Twitter:	https://twitter.com/initmax
Tomáš Heřmánek:	+420 732 447 184