

# Oracle® Tuxedo

Java Server

12c Release 2 (12.2.2)

April 2016

ORACLE®

---

Oracle Tuxedo Java Server, 12c Release 2 (12.2.2)

Copyright © 1996, 2016, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

This software or hardware and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

# Contents

## Oracle Tuxedo Java Server Configuration

Overview .....	1-5
Setting Up the JVM Library Environment .....	1-6
Configurations in UBBCONFIG. ....	1-6
Tuxedo Java Server Configuration File .....	1-7
Tuxedo Java Server Configuration File Version 2.1 .....	1-8
Tuxedo Java Server Configuration File Version 2.0 .....	1-23
Tuxedo Java Server Configuration File Version 1.0 .....	1-31
Configuration for Tuxedo Java Server Transaction Manager. ....	1-37
Create Spring Application Context .....	1-38
Transaction Management .....	1-39
AUTOTRAN .....	1-41
Supported Spring Application Transaction Modes .....	1-41
Limitations .....	1-43
See Also .....	1-43

## Oracle Tuxedo Java Server Administration

Java Server/Admin Interface. ....	2-1
Perform/Admin Request .....	2-1
tadmin Administration Console .....	2-2
See Also .....	2-2

## Appendix: Java Server Configuration File Schema

Java Server Configuration File Schema . . . . .	A-1
Java Server Configuration Schema File Version 2.1 . . . . .	A-1
Java Server Configuration Schema File Version 2.0 . . . . .	A-17
Java Server Configuration Schema File Version 1.0 . . . . .	A-30
See Also . . . . .	A-33

# Oracle Tuxedo Java Server Configuration

This topic contains the following sections:

- [Overview](#)
- [Setting Up the JVM Library Environment](#)
- [Tuxedo Java Server Configuration File](#)
- [Configuration for Tuxedo Java Server Transaction Manager](#)
- [Transaction Management](#)
- [Limitations](#)

## Overview

The ATMI Java server `TMJAVASVR`, is a new Oracle Tuxedo system server acting as a bridge between the Tuxedo system and Java-implemented services. Tuxedo Java server mainly performs the following tasks:

- Reading the Tuxedo Java server configuration file
- Advertising the services implemented with Java code according to the configuration file
- Launching JVM
- Forwarding requests to the Java world
- Getting and executing the results from java-implemented services

- Returning results to the Java world

## Setting Up the JVM Library Environment

The `TMJAVASVR` server depends on the `libjvm.so` (Unix)/`jvm.dll` (Windows) which is in the JRE package.

Before running `TMJAVASVR`, you must make sure the `libjvm.so` (Unix) or `jvm.dll` (Windows) is located in the library loading path of your platform.

On Unix, set `LD_LIBRARY_PATH` or `SHLIB_PATH` (HP-UX only) or `LIBPATH` (AIX only) accordingly.

Do one of the following according to the platform you are using:

- On Oracle Enterprise Linux (64 bit) and Java SE, make sure `$JRE_HOME/lib/amd64/server` is included in `$LD_LIBRARY_PATH`.
- On Oracle Enterprise Linux (64 bit) and Oracle JRockit JVM, make sure `$JRE_HOME/lib/amd64/jrockit` and `$JRE_HOME/lib/amd64` are included in `$LD_LIBRARY_PATH`.
- On AIX PPC64, we suggest that you include both `$JRE_HOME/lib/ppc64/classic` and `$JRE_HOME/lib/ppc64` in `$LIBPATH`.
- On Solaris sparc 64-bit, make sure `$JRE_HOME/lib/sparcv9/server` is included in `$LD_LIBRARY_PATH`
- On Solaris x64, make sure `$JRE_HOME/lib/amd64/server` is included in `$LD_LIBRARY_PATH`
- On Windows, make sure the `PATH` environment variable includes `%JRE_HOME%\bin\server`.

**Note:** The `$JRE_HOME` points to the JRE home directory. You can also refer to `$TUXDIR/tux.env` for the setting.

## Configurations in UBBCONFIG

You need to do the following configurations in `UBBCONFIG`.

- Configure the path where the Tuxedo Java server finds the configuration file for the Java-implemented services in `TMJAVASVR CLOPT`. For more information see, [TMJAVASVR \(5\)](#)

- Specify `MINDISPATCHTHREADS` and `MAXDISPATCHTHREADS` for Java server to decide its running mode: the multi-thread mode or the single-thread mode.

If neither `MINDISPATCHTHREADS` nor `MAXDISPATCHTHREADS` is specified or if `MAXDISPATCHTHREADS` is specified to 1, Java server runs in the single-thread mode; other than that, Java server runs in multi-thread mode.

[Listing 1-1](#) shows an example.

---

#### Listing 1-1 TMJAVASVR Configurations in UBBCONFIG Example

---

```
*SERVERS
TMJAVASVR SRVGRP=TJSVRGRP SRVID=3
CLOPT="-- -c /home/oracle/app/javaserver/TJSconfig.xml"
MINDISPATCHTHREADS=2 MAXDISPATCHTHREADS=3
```

---

## Tuxedo Java Server Configuration File

Tuxedo Java server configuration file is an XML file that defines the parameters necessary to run the Java-implemented services in JVM.

There are three Oracle Tuxedo Java server configuration file versions: 2.1, 2.0, and 1.0. (the corresponding schema files are: `TJSconfig_2.1.xsd`, `TJSconfig_2.0.xsd`, and `TJSconfig_1.0.xsd`). All are installed in the `$TUXDIR/udataobj/tuxj` (Unix), or `%TUXDIR%\udataobj\tuxj` (Windows) directory.

Version 2.1 introduces an element with the tag identifier "server-module" (similar to the previous `tux-server-config` element). A `server-module` element can own multiple `classpaths` and `server-class` child elements. These child elements are bundled in a separate server module that is easier to manage. Also, some new attributes are added in version 2.1 configuration file.

If the `version` attribute is specified, Java server searches the current built-in version table to validate if the `version` value is supported by Java server or not. If validation fails, Java server stops initializing, and exits. If the `version` is valid, then Java server loads the corresponding schema file for further configuration file validation.

**Note:** Though all versions are supported, it is strongly suggested to use version 2.1 (specify version attribute in `TJSconfig` to 2.1). If version attribute is not set, Java server treats the configuration file as version 1.0 by default.

- [Tuxedo Java Server Configuration File Version 2.1](#)
- [Tuxedo Java Server Configuration File Version 2.0](#)
- [Tuxedo Java Server Configuration File Version 1.0](#)

## Tuxedo Java Server Configuration File Version 2.1

- [Version 2.1 Configuration File Elements](#)
- [Advertising Services](#)
- [Version 2.1 Configuration File Example](#)
- [Mapping Java Services](#)

### Version 2.1 Configuration File Elements

Tuxedo Java server configuration file contains the following elements.

- Root Element
  - [TJSconfig](#)
- Main elements
  - [java-config](#)
  - [tux-config](#)
  - [classpath-config](#)
  - [tux-resources](#)
  - [jdbc-resources](#)
  - [tux-server-config](#)

### TJSconfig

It is the root element of Tuxedo Java server configuration file. Java server firstly does a pre-fetching operation to verify whether `TJSconfig` exists in the current configuration file. If



TJSconfig is not found, Java server logs an error and exits. [Table 1-1](#) lists the complete set of TJSconfig attribute.

**Table 1-1 TJSconfig Attribute**

Name	Type	Range	Descriptions
version	Enumeration	2.1	Optional. If it is not specified, Java server treats the configuration file as a version 1.0 configuration file.

## java-config

java-config specifies JVM options that are read by Java server and passed to JVM when Java server creates JVM. These options are used to tune JVM or pass extra properties to JVM. With this element, users can control JVM behaviors in terms of GC policy, stack size, heap size, and so on. [Table 1-2](#) lists the complete set of java-config nested elements.

**Table 1-2 java-config Nested Elements**

Name	Type	Range	Descriptions
jvm-options	String	1..255	Specifies JVM options that are read by Java server and passed to JVM when Java server creates JVM.

It is allowable to specify multiple JVM options in one jvm-options; these options should be separated by spaces.

**Note:** Java server doesn't validate any specified JVM options but only construct the standard JNI JVM initialization data structure with these options and pass them into JVM to handle; however, for -Xms and -Xmx, if users do not specify these two options, Oracle Tuxedo Java server replaces their built-in values with 256m and 256m, respectively, on IBM AIX 6.1 (32-bit) on IBM PowerPC, or with 256m and 512m, respectively, on Oracle Linux 5.6 (64-bit) on Exalotic 2.0.

## tux-config

tux-config specifies attributes related to Tuxedo applications. [Table 1-3](#) lists the complete set of tux-config nested elements.

**Table 1-3 tux-config Nested Elements**

Name	Type	Range	Descriptions
server-clopt	String	1..255	Specifies some startup options for the implemented methods, <code>tpsvrinit (String [])</code> and <code>tpsvrthrinit (String [])</code> , in Java server classes. More specifically, if users' Java server classes implement at least one of these two methods, Java server reads these options, constructs a string array, and passes the string array to the implemented method(s).
<server-propopt>	String	1..255	Specifies the KV pairs. Application can obtain the pairs by using the <code>getTuxConfigProperty (K)</code> interface.

It is allowable to specify multiple options in one `server-clopt`; these options should be separated by spaces.

### classpath-config

`classpath-config` specifies the packages used to look for user-defined classes. Differed from `ClassPaths` in version 1.0, `classpath` string in version 2.0 can be specified with a variable externally defined by an environment variable or a JVM option string specified in `jvm-options` with "-D" option. For example,

```
<classpath>${USER_JAVA_LIB}/lib1</classpath>
```

When adding this class path, Java server firstly searches `jvm-options` list to find whether `-DUSER_JAVA_LIB=...` is already specified. If it is found, Java server replaces the `USER_JAVA_LIB` with the value specified in `-DUSER_JAVA_LIB=` and passes the result to the class path searching list; otherwise, Java server finds whether the environment variable `USER_JAVA_LIB` is specified and passes the result to the class path searching list. If neither `-DUSER_JAVA_LIB=...` nor the environment variable `USER_JAVA_LIB` is found, Java server skips the unspecified class paths and logs an error in `ULOG`.

**Note:** Java server supports the following three modes to use variables in class path specifications.

- `${RELATIVE_PATH_VAR}`
- `$(RELATIVE_PATH_VAR)`

- %RELATIVE\_PATH\_VAR% (Windows only)

Table 1-4 lists the complete set of `classpath-config` nested elements.

**Table 1-4 classpath-config Nested Elements**

Name	Type	Range	Descriptions
<code>classpath</code>	String	1..511	Specifies class path information, which is used by Java server to find and load a class.

It is allowable to specify multiple class paths in one `classpath`; these class paths should be separated by system path separators (";" in Windows platforms; ":" in Unix-like platforms).

Additionally, Java server wildcard mode is enhanced to support recursively searching all sub directories to find matched results.

## tux-resources

`tux-resources` specifies `view/view32` and `fml/fml32` typed buffer classes to instantiate buffer resource objects. Table 1-5 lists the complete set of `tux-resources` nested elements.

**Table 1-5 tux-resources Nested Elements**

Name	Type	Range	Descriptions
<code>classpath</code>	String	1..511	Optional. It specifies the class paths of Tuxedo typed buffer classes.  <b>Note:</b> Users can also specify the class paths of Tuxedo typed buffer classes in <code>classpath-config</code> element.
<code>f1d-tb116-class</code>	String	1..127	Class name list for FML typed buffer.
<code>f1d-tb132-class</code>	String	1..127	Class name list for FML32 typed buffer.
<code>view16-class</code>	String	1..127	Class name list for VIEW typed buffer.
<code>view32-class</code>	String	1..127	Class name list for VIEW32 typed buffer.

Users can specify multiple `FML/FML32/VIEW/VIEW32` classes in one element; these classes should be separated by commas.

Please refer to *Developing Oracle WebLogic Tuxedo Connector Applications for Oracle WebLogic Server* for more information about `jatmi` typed buffer.

Users can directly use `viewj/viewj32` compilers and `mkfldclass/mkfldclass32` utilities to generate corresponding `VIEW/VIEW32` classes and `FML/FML32` classes. Before running these compilers or utilities, specify the class path, where `$TUXDIR/udataobj/tuxj/com.bea.core.jatmi_12.2.2.0.jar` is included.

### **jdbc-resources**

`jdbc-resources` specifies the JDBC compatible data source parameters. [Table 1-6](#) lists the complete set of `jdbc-resources` nested elements.

**Table 1-6 jdbc-resources Nested Elements**

<b>Name</b>	<b>Descriptions</b>
<code>data-source</code>	It includes entries for JDBC data source to create database connection.

[Table 1-7](#) and [Table 1-8](#) list the complete set of `data-source` attributes and nested elements.

**Table 1-7 data-source Attributes**

<b>Name</b>	<b>Type</b>	<b>Range</b>	<b>Descriptions</b>
<code>vendor</code>	String	1..31	Mandatory. Specifies the driver vendor name.
<code>driver-type</code>	String	1..31	Mandatory. Specifies the driver type.
<code>enabled</code>	Boolean	true/false	Optional; the default value is true. Specifies whether the data source is enabled.

**Table 1-8 data-source Nested Elements**

Name	Type	Range	Descriptions
classpath	String	1..511	Optional. Specifies the data source driver package path. Users can also specify the driver class path in classpath-config element.
datasource-class name	String	1..255	Mandatory. Specifies the JDBC driver class name.
driver-params	It includes entries for the JDBC driver used to create database connection.		

[Table 1-9](#) lists the complete set of driver-params nested elements.

**Table 1-9 driver-params Nested Elements**

Name	Type	Range	Descriptions
connection-url	String	1..10239	Specifies the data source connection URL.

## tux-server-config

[Table 1-10](#) tux-server-config specifies Java server class attributes.

**Table 1-10 tux-server-config Attributes**

Name	Type	Range	Descriptions
maxmodules	short	1..1024	The maximum number of server modules in the Java server. Optional. The default value is 256

tux-server-config contains the nested element sever-module.

[Table 1-11](#) list the complete set of tux-server-config/server-module attributes.

**Table 1-11 tux-server-config/server-module Attributes**

Property	Type	Range	Description
name	string	1..63	The name of server-module element.  Mandatory.  The server-module name must be unique within all server-module names in the <tux-server-config> element section and cannot contain an asterisk (*), comma (,), or colon (:).
maxsvrclasses	short	1..8192	The maximum number of Java server classes in the server module.  Optional. The default value is 256.
maxadmclasses	short	1..4096	The maximum number of /Admin command classes in a server module.  It is optional. The default value is 128.
maxadmcmds	short	1..4096	The maximum number of /Admin commands in the Java server.  It is optional. The default value is 128

[Table 1-12](#) list the complete set of tux-server-config/server-module nested elements.

**Table 1-12 tux-server-config/server-module Nested Elements**

Property	Type	Range	Description
<classpath>	string	1..511	Optional. It specifies the class path for server classes. Users can also specify the class path in classpath-config section.
<server-clopt>	string	1..255	Specifies the argument parameters and pass them to tpsvrinit().
<server-propopt>	String	1..255	Specifies the KV pairs. Application can obtain the pairs by using the getTuxConfigProperty (K) interface.
<server-class>	Contains a nested element called services.		

tux-server-config/server-module/server-class attributes are listed in [Table 1-13](#)

**Table 1-13 tux-server-config/server-module/server-class Attributes**

Property	Type	Range	Description
name	String	1..255	Specifies the server class name.
autoadv	boolean	yes/no	Allows Java server to advertise service method as a Tuxedo service with same name as the method name when it is not explicitly specified in the configuration file.  Optional. The default value is "yes" to keep compatible with previous releases.

server-class also has a nested element called services. services has a nested element called service.

Table 1-14 list the complete set of tux-server-config/server-module/server-class/services/service attributes.

**Table 1-14 service Attributes**

Name	Type	Range	Descriptions
name	String	1..127	Mandatory. Specifies the service name that is advertised in Tuxedo application domain.
target	String	1..255	Specifies the method implemented in server class. The method must be declared with TPSVCINFO parameter and with a void type of return value.

## Advertising Services

Each Tuxedo Java server class that implements services must implement a set of methods, of which input argument parameter is the TPSVCINFO interface. The methods that are advertised as services must be the public method and have the return type set to void.



Tuxedo Java server advertises all of public methods as services into bulletin board. If `Services` is specified in configuration file, Tuxedo Java server uses the value of `name` property as the service name, otherwise, the method name is used as the service name.

Java server has three methods to advertise a Tuxedo service:

1. Advertise a Tuxedo service with the name specified in the configuration file

```
<tux-server-config>/<server-module>/<server-class>/<services>/<service>
:name and associate this Tuxedo service with the Java method specified in
<tux-server-config>/<server-module>/<server-class>/<services>/<service>
:target.
```

### Listing 1-2 Advertise a Tuxedo Service Specified in the Configuration File

---

```
<TJSconfig version="2.1">
...
  <tux-server-config maxmodules="100" maxadmcmts="100">
<server-module name="MyAppModule_1">
  <server-class name="MyTuxedoServerClass">
    <services>
      <service name="svc1" target="method1"></service>
      ...
    </services>
  </server-class>
</server-module>
  </tux-server-config>
</TJSconfig>
```

---

In this sample, the Java `method1` implemented in Java class `MyTuxedoServerClass` is associated with the Tuxedo service `svc1`. The exact service handling logic is implemented in `MyTuxedoServerClass.method1` written in Java language.

2. If a Java service method implemented in a Java server class specified in the configuration file is not referred, and the "autoadv" attribute of the Java server class is not set (default value is "yes" to keep compatible with previous releases) or explicitly set to "yes", then the service method is advertised to a Tuxedo service with the name same as the service method.

If the attribute value of "autoadv" is set to "no", then the rest of service methods implemented in this Java server class are not automatically advertised.

3. Invoking `TuxAppContext.tpadvertise()` to advertise a Tuxedo service. Using this method, the Java server application does not need to explicitly specify the Java server class in the configuration file, and the server class does not inherit from the abstract class `com.oracle.tuxedo.tjatmi.TuxedoJavaServer`.

## Version 2.1 Configuration File Example

Version 2.1 introduces a module entity used to separate a previous application into more modules in order to provide a way to manage the Java server application in fine-grained. Some new attributes are also added to the configuration file.

### Listing 1-3 Configuration File Version 2.1 Example

---

```
<TJSconfig version="2.1">
  <java-config>
    <jvm-options>-XX:MaxPermSize=192m -Xms256m </jvm-options>
    <jvm-options>-server</jvm-options>
    <jvm-options>-Xmx512m</jvm-options>
  </java-config>

  <tux-config>
    <server-clopt>-C dom=TMJSVRDOM -t 1000</server-clopt>
    <server-clopt>-n 200</server-clopt>
    <server-propopt> TUXCNF_propK1=propV1 </server-propopt>
    <server-propopt> TUXCNF_propK2=propV2 TUXCNF_propK3=propV3
  </server-propopt>
</tux-config>
```

```
<classpath-config>
  <classpath>${APPDIR}/classes</classpath>
  <classpath>${USERCLASS}/usr/libs/lib1.jar</classpath>
  <classpath>${USERCLASS}/usr/lib2/*</classpath>
  <classpath>${USERCLASS}/usr/lib3/*.jar</classpath>
</classpath-config>

<tux-resources>
  <classpath>${APPDIR}/tuxres</classpath>
  <classpath>${APPDIR}/testres</classpath>
  <fld-tbl16-class>test1.FML16TBL, test2.FML16TBL</fld-tbl16-class>
  <fld-tbl16-class>test3.FML16TBL</fld-tbl16-class>
  <fld-tbl16-class>test4.FML16TBL</fld-tbl16-class>
  <fld-tbl32-class>test1.FML32TBL</fld-tbl32-class>
  <fld-tbl32-class>test2.FML32TBL, test3.FML32TBL</fld-tbl32-class>
  <fld-tbl32-class>test4.FML32TBL</fld-tbl32-class>
  <view16-class>test1.VIEW16, test2.VIEW16</view16-class>
  <view16-class>test3.VIEW16</view16-class>
  <view16-class>test4.VIEW16</view16-class>
  <view32-class>test1.VIEW32</view32-class>
  <view32-class>test2.VIEW32, test3.VIEW32</view32-class>
  <view32-class>test4.VIEW32</view32-class>
</tux-resources>

<jdbc-resources>
  <data-source vendor="Oracle" driver-type="thin" enabled="true"
```

## Oracle Tuxedo Java Server Configuration

```
        object-type="usr" jndi-name="none" pool-name="none">
        <classpath>$(JDBC_DRIVER_ORA)/11gR2_ojdbc6_g.jar</classpath>
        <classpath>$(JDBC_DRIVER_ORA)/11gR2_xdb6.jar</classpath>
        <classpath>$(JDBC_DRIVER_ORA)/11gR2_orai18n.jar</classpath>

<datasource-classname>oracle.jdbc.xa.client.OracleXADataSource</datasource
-classname>

        <res-type>javax.sql.XADataSource</res-type>
        <driver-params>
                <connection-url>
                        jdbc:oracle:thin:@//10.182.54.144:1521/javaorcl
                </connection-url>
                <property name="user" value="tiger"/>
                <property name="password" value="scott"/>
                <property name="autocommit" value="true"/>
                <property name="autotran" value="yes"/>
                <property name="query" value=" "/>
                <property name="thread" value="true"/>
        </driver-params>
</data-source>
</jdbc-resources>

<jdbc-connection-pool>
</jdbc-connection-pool>

<tux-server-config maxmodules="100">
        <server-module name="MyAppModule_1" maxsvrclasses="100"
maxadmclasses="64" maxadmcmds="200">
```

```

<classpath>${APPDIR}/Java/module_1/classes</classpath>
<classpath>${APPDIR}/Java/module_1/lib</classpath>
<server-clopt>-C dom=TMJSVRDOM1 -g TJSSVRGRP1 -t
2000</server-clopt>
<server-clopt>-f application1.xml </server-clopt>
<server-propopt> K1=V1 K2=V2 </server-propopt>
<server-propopt> k3=v3 k4=v4</server-propopt>
<server-class name="MyTuxedoServerClass" autoadv="no">
    <services>
        <service name="mou1_svc1" target="method1"></service>
        <service name="mou1_svc2" target="method2"></service>
    </services>
</server-class>
<server-class name="MyTuxedoServerClass2" autoadv="yes">
    <services>
        <service name="mou1_svc3"
            target="JAVASTRINGSVC"></service>
        <service name="mou1_svc4"
            target="JAVAFML32SVC"></service>
    </services>
</server-class>
</server-module>
<server-module name="MyAppModule_2" maxsvrclasses="200"
maxadmcmds="20">
    <classpath>${APPDIR}/Java/module_2/classes</classpath>
    <classpath>${APPDIR}/Java/module_2/lib</classpath>
    <server-clopt>-C dom=TMJSVRDOM2 -g TJSSVRGRP2 -t
1000</server-clopt>
    <server-clopt>-f application2.xml </server-clopt>

```

```
<server-propopt> Kmou1=Vmou1 Kmou2=Vmou2 </server-propopt>
<server-propopt> kmou3=vmou3 kmou4=vmou4</server-propopt>
<server-class name="MyTuxedoServerClass">
    <services>
        <service name="mou2_svc1" target="method1"></service>
        <service name="mou2_svc2" target="method2"></service>
    </services>
</server-class>
<server-class name="MyTuxedoServerClass2" autoadv="yes">
    <services>
        <service name="mou2_svc3"
target="JAVASTRINGSVC"></service>
        <service name="mou2_svc4"
target="JAVAFML32SVC"></service>
    </services>
</server-class>
</server-module>
</tux-server-config>
</TJSconfig>
```

---

## Mapping Java Services

Java server retrieves all service methods implemented in application server classes and advertises services under the following rules.

- One method can be advertised for multiple times with different service names.
- The methods that are specified in the configuration file are advertised with the specified service name.
- The methods that are not specified in the configuration file are advertised with the method name as the service name.

- The methods that have been advertised with their specified service names in the configuration file will not be advertised any more with the same method name.

Only the methods that have been declared to public in the server class are retrieved by Java server to participate the advertising action.

## Tuxedo Java Server Configuration File Version 2.0

- [Version 2.0 Configuration File Elements](#)
- [Version 2.0 Configuration File Example](#)
- [Mapping Java Services](#)

### Version 2.0 Configuration File Elements

Tuxedo Java server configuration file contains the following elements.

- Root Element
  - [TJSconfig](#)
- Main elements
  - [java-config](#)
  - [tux-config](#)
  - [classpath-config](#)
  - [tux-resources](#)
  - [jdbc-resources](#)
  - [tux-server-config](#)

#### TJSconfig

It is the root element of Tuxedo Java server configuration file. Java server firstly does a pre-fetching operation to verify whether `TJSconfig` exists in the current configuration file. If `TJSconfig` is not found, Java server logs an error and exits. [Table 1-15](#) lists the complete set of `TJSconfig` attribute.

**Table 1-15 TJSconfig Attribute**

Name	Type	Range	Descriptions
version	Enumeration	2.0	It is optional. If it is not specified, Java server treats the configuration file as a version 1.0 configuration file.

### java-config

`java-config` specifies JVM options that are read by Java server and passed to JVM when Java server creates JVM. These options are used to tune JVM or pass extra properties to JVM. With this element, users can control JVM behaviors in terms of GC policy, stack size, heap size, and so on. [Table 1-16](#) lists the complete set of `java-config` nested elements.

**Table 1-16 java-config Nested Elements**

Name	Type	Range	Descriptions
<code>jvm-options</code>	String	1..255	It specifies JVM options that are read by Java server and passed to JVM when Java server creates JVM.

It is allowable to specify multiple JVM options in one `jvm-options`; these options should be separated by spaces.

**Note:** Java server does not validate any specified JVM options but only construct the standard JNI JVM initialization data structure with these options and pass them into JVM to handle; however, for `-Xms` and `-Xmx`, if users do not specify these two options, Oracle Tuxedo Java server replaces their built-in values with 256m and 256m, respectively, on IBM AIX 6.1 (32-bit) on IBM PowerPC, or with 256m and 512m, respectively, on Oracle Linux 5.6 (64-bit) on Exalogic 2.0.

### tux-config

`tux-config` specifies attributes related to Tuxedo applications. [Table 1-17](#) lists the complete set of `tux-config` nested elements.



**Table 1-17 tux-config Nested Elements**

Name	Type	Range	Descriptions
server-clopt	String	1 . . 255	It specifies some startup options for the implemented methods, <code>tpsvrinit (String [])</code> and <code>tpsvrthrininit (String [])</code> , in Java server classes. More specifically, if users' Java server classes implement at least one of these two methods, Java server reads these options, constructs a string array, and passes the string array to the implemented method(s).

It is allowable to specify multiple options in one `server-clopt`; these options should be separated by spaces.

### classpath-config

`classpath-config` specifies the packages used to look for user-defined classes. Differed from `ClassPaths` in version 1.0, `classpath` string in version 2.0 can be specified with a variable externally defined by an environment variable or a JVM option string specified in `jvm-options` with "-D" option. For example,

```
<classpath>${USER_JAVA_LIB}/lib1</classpath>
```

When adding this class path, Java server firstly searches `jvm-options` list to find whether `-DUSER_JAVA_LIB=...` is already specified. If it is found, Java server replaces the `USER_JAVA_LIB` with the value specified in `-DUSER_JAVA_LIB=` and passes the result to the class path searching list; otherwise, Java server finds whether the environment variable `USER_JAVA_LIB` is specified and passes the result to the class path searching list. If neither `-DUSER_JAVA_LIB=...` nor the environment variable `USER_JAVA_LIB` is found, Java server skips the unspecified class paths and logs an error in `ULOG`.

**Note:** Java server supports the following three modes to use variables in class path specifications.

- `${RELATIVE_PATH_VAR}`
- `$(RELATIVE_PATH_VAR)`
- `%RELATIVE_PATH_VAR%` (Windows only)

[Table 1-18](#) lists the complete set of `classpath-config` nested elements.

**Table 1-18 classpath-config Nested Elements**

Name	Type	Range	Descriptions
classpath	String	1..511	It specifies class path information, which is used by Java server to find and load a class.

It is allowable to specify multiple class paths in one `classpath`; these class paths should be separated by system path separators (";" in Windows platforms; ":" in Unix-like platforms).

Additionally, Java server wildcard mode is enhanced to support recursively searching all sub directories to find matched results.

### tux-resources

`tux-resources` specifies `view/view32` and `fml/fml32` typed buffer classes to instantiate buffer resource objects. [Table 1-19](#) lists the complete set of `tux-resources` nested elements.

**Table 1-19 tux-resources Nested Elements**

Name	Type	Range	Descriptions
classpath	String	1..511	It is optional. It specifies the class paths of Tuxedo typed buffer classes.  <b>Note:</b> Users can also specify the class paths of Tuxedo typed buffer classes in <code>classpath-config</code> element.
f1d-tb116-class	String	1..127	It is the class name list for FML typed buffer.
f1d-tb132-class	String	1..127	It is the class name list for FML32 typed buffer.
view16-class	String	1..127	It is the class name list for VIEW typed buffer.
view32-class	String	1..127	It is the class name list for VIEW32 typed buffer.

Users can specify multiple `FML/FML32/VIEW/VIEW32` classes in one element; these classes should be separated by commas.

Please refer to [Developing Oracle WebLogic Tuxedo Connector Applications for Oracle WebLogic Server](#) for more information about `jatmi` typed buffer.

Users can directly use `viewj/viewj32` compilers and `mkfldclass/mkfldclass32` utilities to generate corresponding `VIEW/VIEW32` classes and `FML/FML32` classes. Before running these compilers or utilities, please specify the class path, where `$TUXDIR/udataobj/tuxj/com.bea.core.jatmi_2.0.0.0.jar` is included.

## **jdbc-resources**

`jdbc-resources` specifies the JDBC compatible data source parameters. [Table 1-20](#) lists the complete set of `jdbc-resources` nested elements.

**Table 1-20 jdbc-resources Nested Elements**

Name	Descriptions
<code>data-source</code>	It includes entries for JDBC data source to create database connection.

[Table 1-21](#) and [Table 1-22](#) list the complete set of `data-source` attributes and nested elements.

**Table 1-21 data-source Attributes**

Name	Type	Range	Descriptions
<code>vendor</code>	String	1..31	It is mandatory. It specifies the driver vendor name.
<code>driver-type</code>	String	1..31	It is mandatory. It specifies the driver type.
<code>enabled</code>	Boolean	true/false	It is optional; the default value is <code>true</code> . It specifies whether the data source is enabled.

**Table 1-22 data-source Nested Elements**

Name	Type	Range	Descriptions
classpath	String	1..511	It is optional. It specifies the data source driver package path. Users can also specify the driver class path in <code>classpath-config</code> element.
datasource-class name	String	1..255	It is mandatory. It specifies the JDBC driver class name.
driver-params	It includes entries for the JDBC driver used to create database connection.		

[Table 1-23](#)The following table lists the complete set of `driver-params` nested elements.

**Table 1-23 driver-params Nested Elements**

Name	Type	Range	Descriptions
connection-url	String	1..511	It specifies the data source connection URL.

## tux-server-config

`tux-server-config` specifies Java server class parameters. [Table 1-24](#) lists the complete set of `tux-server-config` nested elements.

**Table 1-24 tux-server-config Nested Elements**

Name	Type	Range	Descriptions
classpath	String	1..511	It is optional. It specifies the class path for server classes. Users can also specify the class path in <code>classpath-config</code> section.
server-class	It includes entries for the Java server class; users must specify server class names and can specify service alias for the methods.		

[Table 1-25](#)he following table lists the complete set of `server-class` attributes.

**Table 1-25 server-class Attributes**

Name	Type	Range	Descriptions
name	String	1..255	It specifies the server class name.

server-class also has a nested element called services. services has a nested element called service. [Table 1-26](#) list the complete set of service attributes.

**Table 1-26 service Attributes**

Name	Type	Range	Descriptions
name	String	1..127	It is mandatory. It specifies the service name that is advertised in Tuxedo application domain.
target	String	1..255	It specifies the method implemented in server class. The method must be declared with TPSVCINFO parameter and with a void type of return value.

## Version 2.0 Configuration File Example

[Listing 1-4](#) lists an example to create Tuxedo Java server configuration file.

**Listing 1-4 Example for Creating Tuxedo Java Server Configuration File**

```
<?xml version="1.0" encoding="UTF-8"?>
<TJSconfig version="2.0">
  <java-config>
    <jvm-options>-XX:MaxPermSize=192m</jvm-options>
    <jvm-options>-server</jvm-options>
    <jvm-options>-DMYVAR="var01" -DMYVAR02="var02"</jvm-options>
  </java-config>
  <tux-config>
```

## Oracle Tuxedo Java Server Configuration

```
<server-clopt>-C dom=TMJSVRDOM -t 1000</server-clopt>
<server-clopt>-n 200</server-clopt>
</tux-config>
<classpath-config>
    <classpath>${USER_JAVA_LIB}/lib1</classpath>
</classpath-config>
<tux-resources>
    <fld-tbl16-class>myfmltbl</fld-tbl16-class>
    <fld-tbl32-class>myfmltbl32,customerflds32</fld-tbl32-class>
    <view16-class>myView</view16-class>
    <view32-class>myView32,customerView32</view32-class>
</tux-resources>
<jdbc-resources>
    <data-source vendor="Oracle" driver-type="thin" enabled="true">
<datasource-classname>oracle.jdbc.xa.client.OracleXADataSource</datasource
-classname>
        <driver-params>
            <connection-url>
                jdbc:oracle:thin:@//Server:1521/orcl
            </connection-url>
        </driver-params>
    </data-source>
</jdbc-resources>
<tux-server-config>
    <classpath>$TUXJRE/jce.jar</classpath>
    <server-class name="MyTuxedoJavaServer">
        <services>
            <service name="JAVATOUPPER" target="JAVATOUPPER"></service>
```

```
        <service name="JAVATOUPPERFORWARD"  
target="JAVATOUPPERFORWARD"></service>  
    </services>  
</server-class>  
</tux-server-config>  
</TJSconfig>
```

---

## Mapping Java Services

Java server retrieves all service methods implemented in application server classes and advertises services under the following rules.

- One method can be advertised for multiple times with different service names.
- The methods that are specified in the configuration file are advertised with the specified service name.
- The methods that are not specified in the configuration file are advertised with the method name as the service name.
- The methods that have been advertised with their specified service names in the configuration file will not be advertised any more with the same method name.

Only the methods that have been declared to public in the server class are retrieved by Java server to participate the advertising action.

## Tuxedo Java Server Configuration File Version 1.0

- [Tuxedo Java Server Configuration File Properties](#)
- [Setting the Class Path Element](#)
- [Setting the FML/View Field Table Class](#)
- [Configuration Rules](#)
- [Advertising Services](#)
- [Version 1.0 Configuration File Example](#)

## Tuxedo Java Server Configuration File Properties

Table 1-27 lists available properties in a Java server configuration file. Please see the corresponding [Java Server Configuration File Schema](#), which checks the validation of the configuration XML file for more precise information.

**Table 1-27 Tuxedo Java Server Configuration File Properties**

Property Name	Type	Range	Description
ClassPath	String	Length:0..1024	Specifies the packages used to run the configured classes methods.
DriverClass	String	Length:1..512	Specifies the JDBC DataSource class name.
ConnectionUrl	String	Length:1..512	Specifies the JDBC connection URL string, which should be in accordance with the format required by the corresponding JDBC driver. It should not contain user name and password.
name (For TuxedoServer Class)	String	Length:1..512	Specifies the Tuxedo Java server class name. Mandatory in configuration.
name (For Service)	String	Length:1..127	Specifies the service name that will be advertised on Bulletin Board related with the method. In a configuration file, each "svcname" attribute value must be unique. If this property is not specified, the method name is used as service name.
target	String	Length:1.. 127	Specifies the class method for the corresponding service.
fieldTable16 Class	String	Length:0..1024	A comma-separated list of fully qualified FML classes. Any of FML16 field table classes should be defined here or else the Fldid() / Fname() of the FML will not work.



**Table 1-27 Tuxedo Java Server Configuration File Properties**

Property Name	Type	Range	Description
fieldTable32 class	String	Length:0..1024	A comma-separated list of fully qualified FML classes. All the FML32 field table classes should be defined here or else the Fldid()/ Fname() of the FML32 will not work.
viewFile16Cl ass	String	Length:0..1024	A comma-separated list of fully qualified FML classes. All the view classes that are used by programming must be defined here.
viewFile32Cl ass	String	Length:0..1024	A comma-separated list of fully qualified FML classes. All the view32 classes that are used by programming must be defined here.

## Setting the Class Path Element

Following <ClassPath> element settings are supported in Tuxedo Java configuration file:

<ClassPath>customer\_lib\_path/\*.jar</ClassPath> **or**

<ClassPath>customer\_lib\_path/\*.zip</ClassPath>

Adds all the .jar or .zip archives under customer\_lib\_path to JVM classpath.

<ClassPath>customer\_lib\_path</ClassPath>

Adds all the classes under customer\_lib\_path to JVM classpath.

<ClassPath>customer\_lib\_path/\*</ClassPath>

Adds all the .jar files, .zip files, and classes under customer\_lib\_path to JVM classpath.

<ClassPath>customer\_lib\_path/lib1.jar</ClassPath> **or**

<ClassPath>customer\_lib\_path/lib2.zip</ClassPath>

Adds customer\_lib\_path/lib1.jar or customer\_lib\_path/lib2.zip to JVM classpath.

### Notes:

- All the file searches in customer\_lib\_path are non-recursive.
- The customer\_lib\_path to \$APPDIR can be either absolute path or relative path.
- The \$APPDIR is added into classpath by default.

## Setting the FML/View Field Table Class

To use the `View/View32`, you need to specify the `View` class in `<Resources></Resources>` element. To use the `Fldid()/Fname()` properly, you need to specify the `FML/FML32` field table class in `<Resources></Resources>` element.

## Configuration Rules

You need to follow the following rules when creating Java server configuration file:

- Only `<TuxedoServerClasses>` is mandatory in Java server configuration file, and at least one `TuxedoServerClass` should be configured.
- Currently, only one `<DataSource>` element is allowed to be configured in `<DataSources>` in configuration file. The name property for `<DataSource>` does not take effect for the present.
- Only `<ClassPath>`, `<fieldTable16Class>`, `<fieldTable32class>`, `<viewFile16Class>`, `<viewFile32Class>`, and `<Service>` element can have multiple entities.
- In `<Resources>` section, you should configure `<fieldTable16Class>` first and then `<fieldTable32class>`, `<viewFile16Class>`, and `<viewFile32Class>` in sequence.

**Note:** The configuration file (e.g. `TJSconfig.xml`) will be checked against the Schema file `TJSconfig.xsd` which is located in `$TUXDIR/udataobj/tuxj/TJSconfig.xsd` (Unix) or `$TUXDIR\udataobj\tuxj/TJSconfig.xsd` (Windows). Oracle recommends you refer to `TJSconfig.xsd` for more information about configuration rules.

## Advertising Services

Each Tuxedo Java server class that implements services must implement a set of methods, of which input argument parameter is the `TPSVINFO` interface. The methods that are advertised as services must be the public method and have the return type set to `void`.

Tuxedo Java server advertises all of public methods as services into bulletin board. If `Services` is specified in configuration file, Tuxedo Java server uses the value of name property as the service name, otherwise, the method name is used as the service name.

## Version 1.0 Configuration File Example

[Listing 1-5](#) shows a Tuxedo Java server configuration file example.

Suppose the `MyTuxedoServerClass1` Java class defines `JAVASTRINGSVC`, `JAVAFML32SVC`, and `svc3` methods, and the `MyTuxedoServerClass2` Java class defines `svc4` method, this example initiates the connection to Oracle database with JDBC connection and exports the services: `svc1`, `svc2`, `svc3`, and `svc4`.

Note that the method name `svc3` and `svc4` are exported as the service name because there is no `<Service>` property specified for them in the configuration file.

### Listing 1-5 Tuxedo Java Server Configuration Example

---

```
<?xml version="1.0" encoding="UTF-8"?>
<TJSconfig>
<ClassPaths>
<ClassPath>/home/oracle/app/javaserver/MyTuxedoServerClass.jar</ClassPath>
<ClassPath>/home/oracle/app/oracle/product/11.2.0/dbhome_2/ucp/lib/ucp.jar
</ClassPath>
<ClassPath>/home/oracle/app/oracle/product/11.2.0/dbhome_2/jdbc/lib/ojdbc6
.jar</ClassPath>
</ClassPaths>
<Resources>
    <FieldTable16Classes>test1.FML16TBL,
test2.FML16TBL</FieldTable16Classes>
    <FieldTable16Classes>test3.FML16TBL,
test4.FML16TBL</FieldTable16Classes>
    <FieldTable32Classes>test1.FML32TBL1,
test2.FML32TBL</FieldTable32Classes>
    <FieldTable32Classes>test3.FML32TBL1,
test4.FML32TBL</FieldTable32Classes>
    <ViewFile16Classes>test1.VIEW16,
test2.VIEW16</ViewFile16Classes>
    <ViewFile16Classes>test3.VIEW16,
test4.VIEW16</ViewFile16Classes>
```

## Oracle Tuxedo Java Server Configuration

```
        <ViewFile32Classes>test1.VIEW32,
test2.VIEW32</ViewFile32Classes>
        <ViewFile32Classes>test3.VIEW32,
test4.VIEW32</ViewFile32Classes>
</Resources>
<DataSources>
        <DataSource name="oracle">
<DriverClass>oracle.jdbc.xa.client.OracleXADataSource</DriverClass>
        <JdbcDriverParams>
<ConnectionUrl>jdbc:oracle:thin:@//10.182.54.144:1521/javaorcl</Connection
Url>
        </JdbcDriverParams>
        </DataSource>
</DataSources>
<TuxedoServerClasses>
<TuxedoServerClass name="MyTuxedoServerClass1">
        <Services>
                <Service name="svc1" target="JAVASTRINGSV" ></Service>
                <Service name="svc2" target="JAVAFML32SVC" ></Service>
        </Services>
</TuxedoServerClass>
<TuxedoServerClass name="MyTuxedoServerClass2" ></TuxedoServerClass>
</TuxedoServerClasses>
</TJSconfig>
```

---

**Notes:**

- Because a service name is used to map to specified class + method in the configuration file, it must be unique in one Tuxedo Java server and among the different Java classes in it.
- You need to provide all the libraries (except standard JRE library) used by your Java applications, such as user-defined classes and DataSource driver library.

## Configuration for Tuxedo Java Server Transaction Manager

Java server provides a getter method to create a new or acquire an existing JTA compliant transaction manager. [Listing 1-6](#) demonstrates how to configure a Tuxedo transaction manager in Spring application context configuration file.

For more information about Spring application context configuration file, see [Create Spring Application Context](#).

### Listing 1-6 Sample of Tuxedo Transaction Manager Configuration

---

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:context="http://www.springframework.org/schema/context"
xmlns:aop="http://www.springframework.org/schema/aop"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd
http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop.xsd
http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx.xsd">
```

```
<bean id="TuxTransactionManager"
class="com.oracle.tuxedo.tjatmi.TuxTransactionManager"
factory-method="getTransactionManager">
</bean>

<bean id="springAppTransactionManager"
class="org.springframework.transaction.jta.JtaTransactionManager">
  <property name="userTransaction" ref="TuxTransactionManager"/>
  <property name="transactionManager" ref="TuxTransactionManager"/>
</bean>

</beans>
```

---

## Create Spring Application Context

Java server does not create, destroy, or maintain the Spring application context; Spring applications do.

The Spring Framework libraries and all dependent libraries need to be configured in the Java server configuration file. [Listing 1-7](#) demonstrates how to specify Spring Framework libraries in Java server configuration file.

For more information about Java server configuration file, see [Tuxedo Java Server Configuration File](#).

### Listing 1-7 Sample of Specifying Spring Framework Libraries

---

```
<?xml version="1.0" encoding="UTF-8"?>
<TJSconfig version="2.0">
<java-config>
```

```

    <jvm-options>-DSPRING_LIB=/home/tuxdev/Spring/3.2.4</jvm-options>
    <jvm-options>-DSPRING_DEP=/home/tuxdev/Spring/deps</jvm-options>
</java-config>
<classpath-config>
    <classpath>${SPRING_LIB}/*.jar</classpath>
    <classpath>${SPRING_DEP}/*.jar</classpath>
</classpath-config>
</TJSconfig>

```

---

Java server loads Spring Framework and all dependent libraries according to the configuration information in the Java server configuration file at the booting stage. Spring applications can directly import the Spring Framework classes as they want.

## Transaction Management

The Tuxedo Java-based transaction is implemented based on the existing Tuxedo transaction management framework. As before, the configuration you need to make in the UBB configuration file for transaction is, define the attributes of transaction manager servers (TMSs) and resource managers (RMs) for a particular group. The Tuxedo Java Server (TMJAVASVR) should belong to this group.

### Listing 1-8 UBB Configurations for Java-Based Transaction

---

```

*GROUP

ORASVRGRP  LMID=simple GRPNO=1

OPENINFO="Oracle_XA:Oracle_XA+Acc=P/system/oracle+SesTm=120+MaxCur=5+LogDi
r=..+SqlNet=orcl"

          TMSNAME=TMSORA TMSCOUNT=2

*SERVERS

TMJAVASVR SRVGRP=ORASVRGRP SRVID=3 CLOPT="-- -c TJSconfig_ORA.xml"

          MINDISPATCHTHREADS=2 MAXDISPATCHTHREADS=2

```

For more information, see [Configuring Your ATMI Application to Use Transactions](#).

Since the Java-implemented services need to connect the data source in Java world, it is necessary to provide parameters that define how to connect the data source in Tuxedo Java server configuration file.

In this release, the Java-implemented services can only connect the database with JDBC connection, so you need to specify JDBC parameters in Tuxedo Java server configuration file using `ConnectionUrl` property.

Because the Java server retrieves user name and password from `OPENINFO`, you should provide the user name and password in `OPENINFO` in UBB configuration file. This way encrypts the password and ensures the high security.

Different `DataSource` have different `ConnectionURL` format.

For Oracle database, the format is:

```
jdbc:oracle:thin:@//hostname:portnumber/servicename
```

For DB2 database, the format is:

```
jdbc:db2://hostname:portnumber/DatabaseName
```

[Listing 1-9](#) is a configuration example of Oracle database.

### Listing 1-9 Oracle Database Configuration Example

---

```
<DataSources>
    <DataSource name="oracle">
        <DriverClass>oracle.jdbc.xa.client.OracleXADataSource</DriverClass>
        <JdbcDriverParams>
            <ConnectionUrl>jdbc:oracle:thin:@//10.182.54.144:1521/javaorcl</Connection
            Url>
            </JdbcDriverParams>
        </DataSource>
```



```
</DataSources>
```

---

## AUTOTRAN

The `AUTOTRAN` can also be configured on the services advertised by Tuxedo Java server. When the Tuxedo Java server service is configured in `*SERVICES` section and its `AUTOTRAN` attribute is specified as `Y`, the service starts a transaction automatically.

## Supported Spring Application Transaction Modes

Spring Framework supports two transaction management modes.

- Programmatic transaction mode
- Declarative transaction mode

Declarative transaction can support seven transaction propagation modes that can be mapped to corresponding container management transaction modes in EJB as listed in .

**Table 1-28 Supported Transaction Propagation Modes**

Propagation Constants	Description
<code>PROPAGATION_REQUIRED</code>	Supports a current transaction, or creates a new one if no transaction exists. It is analogous to EJB transaction attribute of the same name. This mode is typically the default setting of a transaction definition.
<code>PROPAGATION_SUPPORTS</code>	Supports a current transaction, or execute non-transactionally if no transaction exists. It is analogous to EJB transaction attribute of the same name.
<code>PROPAGATION_MANDATORY</code>	Supports a current transaction, or throws an exception if no transaction exists. It is analogous to EJB transaction attribute of the same name.
<code>PROPAGATION_REQUIRES_NEW</code>	Creates a new transaction, or suspends the current transaction if exists. It is analogous to EJB transaction attribute of the same name

**Table 1-28 Supported Transaction Propagation Modes**

Propagation Constants	Description
PROPAGATION_NOT_SUPPORTED	Executes non-transactionally, or suspends the current transaction if exists. It is analogous to EJB transaction attribute of the same name.
PROPAGATION_NEVER	Executes non-transactionally, or throws an exception if a transaction exists. It is analogous to EJB transaction attribute of the same name.
PROPAGATION_NESTED*	Executes within a nested transaction if a current transaction exists, or behaves like PROPAGATION_REQUIRED.

**Note:** \*In declarative transaction management mode, the PROPAGATION\_NESTED propagation behavior is not support by Java server transaction manager.

Java server supports two initiated transaction.

- Transaction initiated from Java server, created by Spring Framework
- Transaction initiated from a service requester

The transaction attribute timeout, rollback-for, and no-rollback-for are supported.

**Note:** To process Oracle Database operations with non-transaction model by Tuxedo Java Server Spring Transaction Manager, you should explicitly add the transaction begin/end sentences in your scenario.

## Java Server Transaction Manager Default Timeout

If the timeout value is specified for one bean method, the timeout is used for a new transaction. If neither an explicit transaction timeout value nor a Java server transaction manager timeout is specified, the default built-in transaction timeout value is used as 30 to create a new transaction. Java server supports to specify a default transaction timeout value in the Java server configuration file. shows a sample.

### Listing 1-10 Sample of Specifying a Default Transaction Timeout Value

```
<?xml version="1.0" encoding="UTF-8"?>
<TJSconfig version="2.0">
```

```
<java-config>
  <jvm-options>-Dtuxedo.tjatmi.defaultTransactionTimeout=60</jvm-options>
</java-config>
</TJSconfig>
```

---

## Limitations

- The user cannot launch a new Java thread in the service of Java server.
- The Java Server is a multithreaded process that dispatches the services in multiple threads. It cannot run in the single thread mode.
- Only supports the connection to database through JDBC driver. For DB2, only the Type 4 JDBC driver is supported.
- Conversation is not supported by Oracle Tuxedo Java server.
- MSSQ is not supported by Oracle Tuxedo Java server.
- All Oracle Tuxedo resources specified in `<TJSconfig>/<tux-resources>` are managed and maintained by Java server itself; they should be accessible and visible to all server modules deployed in Java server. These of resource classes must be deployed as public.

## See Also

[TMJAVASVR \(5\)](#)

[Java Server Javadoc](#)

[Oracle Tuxedo Java Server Administration](#)

[Appendix: Java Server Configuration File Schema](#)

[Programming an Oracle Tuxedo Application Using Java](#)



# Oracle Tuxedo Java Server Administration

This topic contains the following sections:

- [Java Server/Admin Interface](#)
- [See Also](#)

## Java Server/Admin Interface

Java server supports a MIB like interface to handle some of administrative operation requests. T\_TJS class can be used to perform some of administrative operations on a specific Java server. T\_TJS is a general MIB class as the /Admin entry invoked from TM\_MIB component. A T\_TJS request may has a sub-class field which can be used to define the exact MIB request at Java server side. For more information., see [TMIB \(5\)](#) .

## Perform/Admin Request

There are methods to perform the /Admin request by calling TMIB service or Java server /Admin service: using TMIB, or using Java Server /Admin Interface.

If invoking the /Admin request via TMIB the service name is .TMIB, and the TA\_SRVGRP and TA\_SRVID are mandatory.

---

### Listing 2-1 Invoking the /Admin Request Using TMIB

---

```
SRVCNM .TMIB
```

```
TA_CLASS      T_TJS
TA_OPERATION  GET/SET
TA_SRVGRP    JSVRGRP
TA_SRVID     1001
TA_TJS_CLASS T_TJSXXX
...
```

---

If invoking the /Admin request via Java server /Admin interface the service name is ..TJSADM\_GRPID\_SRVID, the TA\_SRVGRP and TA\_SRVID are optional.

#### **Listing 2-2 Invoking the /Admin Request Using Java Server /Admin Interface**

---

```
SRVCNM ..TJSADM_9200_1001
TA_CLASS      T_TJS
TA_OPERATION  GET/SET
TA_TJS_CLASS  T_TJSXXX
...
```

---

## **tmadmin Administration Console**

A `tmadmin` sub-command, `tjsreload`, is provided so that an administrator can perform a full reload, and reload a server module to a Java server. `tjsreload` uses the group name and server identifier to locate the target Java server. For more information, see *tmadmin Commands* in [tmadmin \(1\)](#).

## **See Also**

[TMJAVASVR\(5\)](#)

[Java Server Javadoc](#)

[Oracle Tuxedo Java Server Configuration](#)

[Appendix: Java Server Configuration File Schema](#)  
[Programming an Oracle Tuxedo Application Using Java](#)





# Appendix: Java Server Configuration File Schema

## Java Server Configuration File Schema

- [Java Server Configuration Schema File Version 2.1](#)
- [Java Server Configuration Schema File Version 2.0](#)
- [Java Server Configuration Schema File Version 1.0](#)

## Java Server Configuration Schema File Version 2.1

[Listing A-1](#) shows the Java Server configuration XML schema version 2.1.

### Listing A-1 Java Server Configuration XML Schema File Version 2.1

---

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified">
  <!-- TJSconfig Configuration Root Element -->
  <xs:element name="TJSconfig">
    <xs:complexType>
      <xs:annotation>
```

## Appendix: Java Server Configuration File Schema

```

    <xs:documentation>Tuxedo Java Server Configuration Root
Element</xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element minOccurs="0" maxOccurs="1" ref="java-config"/>
    <xs:element minOccurs="0" maxOccurs="1" ref="tux-config"/>
    <xs:element minOccurs="0" maxOccurs="1" ref="classpath-config"/>
    <xs:element minOccurs="0" maxOccurs="1" ref="tux-resources"/>
    <xs:element minOccurs="0" maxOccurs="1" ref="jdbc-resources"/>
    <xs:element minOccurs="0" maxOccurs="1" ref="jdbc-connection-pool"/>
    <xs:element minOccurs="1" maxOccurs="1" ref="tux-server-config"/>
  </xs:sequence>
  <xs:attribute name="version" use="required">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="2.1"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
</xs:complexType>
</xs:element>
<!-- ***** -->
<!--           java-config section           -->
<!-- ***** -->
<xs:element name="java-config">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="unbounded" ref="jvm-options"/>

```

```

    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="jvm-options">
  <xs:simpleType>
    <xs:annotation>
      <xs:documentation>255-chars string only</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:minLength value="2" />
      <xs:maxLength value="255" />
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<!-- ***** -->
<!--           tux-config section           -->
<!-- ***** -->
<xs:element name="tux-config">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="unbounded" ref="server-clopt" />
      <xs:element minOccurs="0" maxOccurs="unbounded"
ref="server-propopt" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
<!-- ***** -->
<!--           classpath-config section           -->

```

## Appendix: Java Server Configuration File Schema

```
<!-- ***** -->
<xs:element name="classpath-config">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="unbounded" ref="classpath"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<!-- ***** -->
<!--          tux-resources section          -->
<!-- ***** -->
<xs:element name="tux-resources">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="unbounded" ref="classpath"/>
      <xs:element minOccurs="0" maxOccurs="unbounded"
ref="fld-tbl16-class"/>
      <xs:element minOccurs="0" maxOccurs="unbounded"
ref="fld-tbl32-class"/>
      <xs:element minOccurs="0" maxOccurs="unbounded" ref="view16-class"/>
      <xs:element minOccurs="0" maxOccurs="unbounded" ref="view32-class"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="fld-tbl16-class">
  <xs:simpleType>
    <xs:annotation>
      <xs:documentation>127-chars string only</xs:documentation>
    </xs:annotation>
  </xs:simpleType>
</xs:element>
```

```

</xs:annotation>
<xs:restriction base="xs:string">
  <xs:minLength value="1" />
  <xs:maxLength value="127" />
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="fld-tbl32-class">
  <xs:simpleType>
    <xs:annotation>
      <xs:documentation>127-chars string only</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:minLength value="1" />
      <xs:maxLength value="127" />
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="view16-class">
  <xs:simpleType>
    <xs:annotation>
      <xs:documentation>127-chars string only</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:minLength value="1" />
      <xs:maxLength value="127" />
    </xs:restriction>
  </xs:simpleType>
</xs:element>

```

## Appendix: Java Server Configuration File Schema

```
</xs:simpleType>
</xs:element>
<xs:element name="view32-class">
  <xs:simpleType>
    <xs:annotation>
      <xs:documentation>127-chars string only</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:minLength value="1" />
      <xs:maxLength value="127" />
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<!-- ***** -->
<!--           jdbc-resources section           -->
<!-- ***** -->
<xs:element name="jdbc-resources">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="1" ref="data-source"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="data-source">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="unbounded" ref="classpath" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

```

<xs:element minOccurs="1" maxOccurs="1" ref="datasource-classname"/>
  <xs:element minOccurs="1" maxOccurs="1" ref="res-type"/>
  <xs:element minOccurs="1" maxOccurs="1" ref="driver-params"/>
</xs:sequence>
<xs:attribute name="vendor" use="required">
  <xs:simpleType>
    <xs:annotation>
      <xs:documentation>31-chars string only</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:minLength value="1" />
      <xs:maxLength value="31"/>
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>
<xs:attribute name="pool-name" use="optional">
  <xs:simpleType>
    <xs:annotation>
      <xs:documentation>63-chars string only</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:minLength value="1" />
      <xs:maxLength value="63"/>
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>
<xs:attribute name="object-type" use="optional">

```

## Appendix: Java Server Configuration File Schema

```
<xs:simpleType>
  <xs:restriction base="xs:string">
    <xs:enumeration value="usr" />
    <xs:enumeration value="sys" />
  </xs:restriction>
</xs:simpleType>
</xs:attribute>
<xs:attribute name="jndi-name" use="optional">
  <xs:simpleType>
    <xs:annotation>
      <xs:documentation>63-chars string only</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:minLength value="1" />
      <xs:maxLength value="63" />
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>
<xs:attribute name="enabled" type="xs:boolean" use="optional"
default="true" />
<xs:attribute name="driver-type" use="required">
  <xs:simpleType>
    <xs:annotation>
      <xs:documentation>31-chars string only</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:minLength value="1" />
      <xs:maxLength value="31" />
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>
```



```

        </xs:restriction>
    </xs:simpleType>
</xs:attribute>
</xs:complexType>
</xs:element>
<xs:element name="datasource-classname">
    <xs:simpleType>
        <xs:annotation>
            <xs:documentation>255-chars string only</xs:documentation>
        </xs:annotation>
        <xs:restriction base="xs:string">
            <xs:minLength value="1" />
            <xs:maxLength value="255" />
        </xs:restriction>
    </xs:simpleType>
</xs:element>
<xs:element name="res-type">
    <xs:simpleType>
        <xs:annotation>
            <xs:documentation>255-chars string only</xs:documentation>
        </xs:annotation>
        <xs:restriction base="xs:string">
            <xs:minLength value="1" />
            <xs:maxLength value="255" />
        </xs:restriction>
    </xs:simpleType>
</xs:element>

```

## Appendix: Java Server Configuration File Schema

```
<xs:element name="driver-params">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="1" maxOccurs="1" ref="connection-url"/>
      <xs:element minOccurs="0" maxOccurs="unbounded" ref="property"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="connection-url">
  <xs:simpleType>
    <xs:annotation>
      <xs:documentation>4095-chars string only</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:minLength value="1" />
      <xs:maxLength value="4095" />
    </xs:restriction>
  </xs:simpleType>
</xs:element>

<xs:element name="property">
  <xs:complexType>
    <xs:attribute name="name" use="optional">
      <xs:simpleType>
        <xs:annotation>
          <xs:documentation>63-chars string only</xs:documentation>
        </xs:annotation>
        <xs:restriction base="xs:string">
```

```

        <xs:minLength value="1" />
        <xs:maxLength value="31"/>
    </xs:restriction>
</xs:simpleType>
</xs:attribute>
<xs:attribute name="value" use="optional">
    <xs:simpleType>
        <xs:annotation>
            <xs:documentation>127-chars string only</xs:documentation>
        </xs:annotation>
        <xs:restriction base="xs:string">
            <xs:minLength value="1" />
            <xs:maxLength value="127"/>
        </xs:restriction>
    </xs:simpleType>
</xs:attribute>
</xs:complexType>
</xs:element>
<!-- ***** -->
<!--          jdbc-connection-pool section          -->
<!-- ***** -->
<xs:element name="jdbc-connection-pool" type="xs:string"/>
<!-- ***** -->
<!--          tux-server-config section          -->
<!-- ***** -->
<xs:element name="tux-server-config">
    <xs:complexType>

```

## Appendix: Java Server Configuration File Schema

```
<xs:sequence>
  <xs:element minOccurs="0" maxOccurs="unbounded"
ref="server-module" />
</xs:sequence>
<xs:attribute name="maxmodules" use="optional">
  <xs:simpleType>
    <xs:annotation>
      <xs:documentation>Greater or equal 1 and less or equal
1024</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:short">
      <xs:minInclusive value="1" />
      <xs:maxInclusive value="1024" />
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>
</xs:complexType>
</xs:element>
<xs:element name="server-module">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="unbounded" ref="classpath" />
      <xs:element minOccurs="0" maxOccurs="unbounded" ref="server-clopt" />
      <xs:element minOccurs="0" maxOccurs="unbounded"
ref="server-propopt" />
      <xs:element minOccurs="1" maxOccurs="unbounded" ref="server-class" />
    </xs:sequence>
    <xs:attribute name="name" use="required">
```

```

<xs:simpleType>
  <xs:annotation>
    <xs:documentation>63-chars string only</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:minLength value="1" />
    <xs:maxLength value="63"/>
  </xs:restriction>
</xs:simpleType>
</xs:attribute>
<xs:attribute name="maxsvrclasses" use="optional">
  <xs:simpleType>
    <xs:annotation>
      <xs:documentation>Greater or equal 1 and less or equal
8192</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:short">
      <xs:minInclusive value="1" />
      <xs:maxInclusive value="8192"/>
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>
<xs:attribute name="maxadmclasses" use="optional">
  <xs:simpleType>
    <xs:annotation>
      <xs:documentation>Greater or equal 1 and less or equal
4096</xs:documentation>
    </xs:annotation>

```

## Appendix: Java Server Configuration File Schema

```
<xs:restriction base="xs:short">
  <xs:minInclusive value="1" />
  <xs:maxInclusive value="4096"/>
</xs:restriction>
</xs:simpleType>
</xs:attribute>
<xs:attribute name="maxadmcmds" use="optional">
  <xs:simpleType>
    <xs:annotation>
      <xs:documentation>Greater or equal 1 and less or equal
4096</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:short">
      <xs:minInclusive value="1" />
      <xs:maxInclusive value="4096"/>
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>
</xs:complexType>
</xs:element>
<xs:element name="server-class">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="unbounded" ref="services"/>
    </xs:sequence>
    <xs:attribute name="name" use="required">
      <xs:simpleType>
        <xs:annotation>
```

```

        <xs:documentation>255-chars string only</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
        <xs:minLength value="1" />
        <xs:maxLength value="255"/>
    </xs:restriction>
</xs:simpleType>
</xs:attribute>
<xs:attribute name="autoadv" use="optional">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:enumeration value="no" />
            <xs:enumeration value="yes" />
        </xs:restriction>
    </xs:simpleType>
</xs:attribute>
</xs:complexType>
</xs:element>
<xs:element name="services">
    <xs:complexType>
        <xs:sequence>
            <xs:element minOccurs="0" maxOccurs="unbounded" ref="service" />
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="service">
    <xs:complexType>

```

## Appendix: Java Server Configuration File Schema

```
<xs:attribute name="name" use="required">
  <xs:simpleType>
    <xs:annotation>
      <xs:documentation>127-chars string only</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:minLength value="1" />
      <xs:maxLength value="127"/>
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>
<xs:attribute name="target" use="required">
  <xs:simpleType>
    <xs:annotation>
      <xs:documentation>255-chars string only</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:minLength value="1" />
      <xs:maxLength value="255"/>
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>
</xs:complexType>
</xs:element>
<!-- ***** -->
<!--          common classpath element          -->
<!-- ***** -->
```



```

<xs:element name="classpath">
  <xs:simpleType>
    <xs:annotation>
      <xs:documentation>10239-chars string only</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:minLength value="0" />
      <xs:maxLength value="10239" />
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<!-- ***** -->
<!--           common server-clopt element           -->
<!-- ***** -->
<xs:element name="server-clopt">
  <xs:simpleType>
    <xs:annotation>
      <xs:documentation>255-chars string only</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">

```

---

## Java Server Configuration Schema File Version 2.0

[Listing A-2](#) shows the Java Server configuration XML schema version 2.0.

### Listing A-2 Java Server Configuration XML Schema File Version 2.0

---

```
<?xml version="1.0" encoding="UTF-8"?>
```

## Appendix: Java Server Configuration File Schema

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
  <xs:element name="TJSconfig">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" maxOccurs="1" ref="java-config"/>
        <xs:element minOccurs="0" maxOccurs="1" ref="tux-config"/>
        <xs:element minOccurs="0" maxOccurs="1" ref="classpath-config"/>
        <xs:element minOccurs="0" maxOccurs="1" ref="tux-resources"/>
        <xs:element minOccurs="0" maxOccurs="1" ref="jdbc-resources"/>
        <xs:element minOccurs="0" maxOccurs="1" ref="jdbc-connection-pool"/>
        <xs:element minOccurs="1" maxOccurs="1" ref="tux-server-config"/>
      </xs:sequence>
      <xs:attribute name="version" use="optional">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="2.0"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:complexType>
  </xs:element>
  <xs:element name="java-config">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" maxOccurs="unbounded" ref="jvm-options"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

```

</xs:element>
<xs:element name="jvm-options">
  <xs:simpleType>
    <xs:annotation>
      <xs:documentation>255-chars string only</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:minLength value="2" />
      <xs:maxLength value="255" />
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="tux-config">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="unbounded" ref="server-clopt" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="server-clopt">
  <xs:simpleType>
    <xs:annotation>
      <xs:documentation>255-chars string only</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:minLength value="2" />
      <xs:maxLength value="255" />
    </xs:restriction>
  </xs:simpleType>
</xs:element>

```

## Appendix: Java Server Configuration File Schema

```
        </xs:restriction>
    </xs:simpleType>
</xs:element>
<xs:element name="classpath-config">
    <xs:complexType>
        <xs:sequence>
            <xs:element minOccurs="0" maxOccurs="unbounded" ref="classpath"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="classpath">
    <xs:simpleType>
        <xs:annotation>
            <xs:documentation>511-chars string only</xs:documentation>
        </xs:annotation>
        <xs:restriction base="xs:string">
            <xs:minLength value="0" />
            <xs:maxLength value="511"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
<xs:element name="tux-resources">
    <xs:complexType>
        <xs:sequence>
            <xs:element minOccurs="0" maxOccurs="unbounded" ref="classpath"/>
            <xs:element minOccurs="0" maxOccurs="unbounded"
ref="fld-tbl16-class"/>
```

```

        <xs:element minOccurs="0" maxOccurs="unbounded"
ref="fld-tbl32-class"/>
        <xs:element minOccurs="0" maxOccurs="unbounded" ref="view16-class"/>
        <xs:element minOccurs="0" maxOccurs="unbounded" ref="view32-class"/>
    </xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="view32-class">
    <xs:simpleType>
        <xs:annotation>
            <xs:documentation>127-chars string only</xs:documentation>
        </xs:annotation>
        <xs:restriction base="xs:string">
            <xs:minLength value="1" />
            <xs:maxLength value="127" />
        </xs:restriction>
    </xs:simpleType>
</xs:element>
<xs:element name="view16-class">
    <xs:simpleType>
        <xs:annotation>
            <xs:documentation>127-chars string only</xs:documentation>
        </xs:annotation>
        <xs:restriction base="xs:string">
            <xs:minLength value="1" />
            <xs:maxLength value="127" />
        </xs:restriction>
    </xs:simpleType>

```

## Appendix: Java Server Configuration File Schema

```
</xs:element>
<xs:element name="fld-tbl32-class">
  <xs:simpleType>
    <xs:annotation>
      <xs:documentation>127-chars string only</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:minLength value="1" />
      <xs:maxLength value="127" />
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="fld-tbl16-class">
  <xs:simpleType>
    <xs:annotation>
      <xs:documentation>127-chars string only</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:minLength value="1" />
      <xs:maxLength value="127" />
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="jdbc-resources">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="1" ref="data-source"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

```

    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="data-source">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="unbounded" ref="classpath"/>
      <xs:element minOccurs="1" maxOccurs="1" ref="datasource-classname"/>
      <xs:element minOccurs="1" maxOccurs="1" ref="res-type"/>
      <xs:element minOccurs="1" maxOccurs="1" ref="driver-params"/>
    </xs:sequence>
    <xs:attribute name="vendor" use="required">
      <xs:simpleType>
        <xs:annotation>
          <xs:documentation>31-chars string only</xs:documentation>
        </xs:annotation>
        <xs:restriction base="xs:string">
          <xs:minLength value="1" />
          <xs:maxLength value="31"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="pool-name" use="optional">
      <xs:simpleType>
        <xs:annotation>
          <xs:documentation>63-chars string only</xs:documentation>
        </xs:annotation>

```

## Appendix: Java Server Configuration File Schema

```
<xs:restriction base="xs:string">
  <xs:minLength value="1" />
  <xs:maxLength value="63"/>
</xs:restriction>
</xs:simpleType>
</xs:attribute>
<xs:attribute name="object-type" use="optional">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="usr"/>
      <xs:enumeration value="sys"/>
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>
<xs:attribute name="jndi-name" use="optional">
  <xs:simpleType>
    <xs:annotation>
      <xs:documentation>63-chars string only</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:minLength value="1" />
      <xs:maxLength value="63"/>
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>
<xs:attribute name="enabled" use="optional" type="xs:boolean"
default="true"/>
<xs:attribute name="driver-type" use="required">
```



```

<xs:simpleType>
  <xs:annotation>
    <xs:documentation>31-chars string only</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:minLength value="1" />
    <xs:maxLength value="31"/>
  </xs:restriction>
</xs:simpleType>
</xs:attribute>
</xs:complexType>
</xs:element>
<xs:element name="datasource-classname">
  <xs:simpleType>
    <xs:annotation>
      <xs:documentation>255-chars string only</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:minLength value="1" />
      <xs:maxLength value="255"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="res-type">
  <xs:simpleType>
    <xs:annotation>
      <xs:documentation>255-chars string only</xs:documentation>

```

## Appendix: Java Server Configuration File Schema

```
</xs:annotation>
<xs:restriction base="xs:string">
  <xs:minLength value="1" />
  <xs:maxLength value="255"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="driver-params">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="1" maxOccurs="1" ref="connection-url"/>
      <xs:element minOccurs="0" maxOccurs="unbounded" ref="property"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="connection-url">
  <xs:simpleType>
    <xs:annotation>
      <xs:documentation>511-chars string only</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:minLength value="1" />
      <xs:maxLength value="511"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="property">
```

```

<xs:complexType>
  <xs:attribute name="name" use="optional">
    <xs:simpleType>
      <xs:annotation>
        <xs:documentation>63-chars string only</xs:documentation>
      </xs:annotation>
      <xs:restriction base="xs:string">
        <xs:minLength value="1" />
        <xs:maxLength value="31"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="value" use="optional">
    <xs:simpleType>
      <xs:annotation>
        <xs:documentation>127-chars string only</xs:documentation>
      </xs:annotation>
      <xs:restriction base="xs:string">
        <xs:minLength value="1" />
        <xs:maxLength value="127"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
</xs:complexType>
</xs:element>
<xs:element name="jdbc-connection-pool" type="xs:string"/>
<xs:element name="tux-server-config">

```

## Appendix: Java Server Configuration File Schema

```
<xs:complexType>
  <xs:sequence>
    <xs:element minOccurs="0" maxOccurs="unbounded" ref="classpath"/>
    <xs:element minOccurs="1" maxOccurs="unbounded" ref="server-class"/>
  </xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="server-class">
  <xs:complexType mixed="true">
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="unbounded" ref="services"/>
    </xs:sequence>
    <xs:attribute name="name" use="required">
      <xs:simpleType>
        <xs:annotation>
          <xs:documentation>255-chars string only</xs:documentation>
        </xs:annotation>
        <xs:restriction base="xs:string">
          <xs:minLength value="1" />
          <xs:maxLength value="255"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
</xs:element>
<xs:element name="services">
  <xs:complexType>
```

```

<xs:sequence>
  <xs:element minOccurs="0" maxOccurs="unbounded" ref="service" />
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="service">
  <xs:complexType>
    <xs:attribute name="name" use="required">
      <xs:simpleType>
        <xs:annotation>
          <xs:documentation>127-chars string only</xs:documentation>
        </xs:annotation>
        <xs:restriction base="xs:string">
          <xs:minLength value="1" />
          <xs:maxLength value="127"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="target" use="required">
      <xs:simpleType>
        <xs:annotation>
          <xs:documentation>255-chars string only</xs:documentation>
        </xs:annotation>
        <xs:restriction base="xs:string">
          <xs:minLength value="1" />
          <xs:maxLength value="255"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
</xs:element>

```

```
        </xs:simpleType>
    </xs:attribute>
</xs:complexType>
</xs:element>
</xs:schema>
```

---

## Java Server Configuration Schema File Version 1.0

[Listing A-3](#) shows the Java Server configuration XML schema version 1.0.

### Listing A-3 Java Server Configuration XML Schema File Version 1.0

---

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified">
  <xs:element name="TJSconfig">
    <xs:complexType>
      <xs:all>
        <xs:element minOccurs="0" maxOccurs="1" ref="ClassPaths"/>
        <xs:element minOccurs="0" maxOccurs="1" ref="DataSources"/>
        <xs:element minOccurs="0" maxOccurs="1" ref="Resources"/>
        <xs:element minOccurs="1" maxOccurs="1" ref="TuxedoServerClasses"/>
      </xs:all>
    </xs:complexType>
  </xs:element>
  <xs:element name="ClassPaths">
    <xs:complexType>
      <xs:sequence>
```

```

        <xs:element minOccurs="0" maxOccurs="unbounded" ref="ClassPath"/>
    </xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="ClassPath" type="xs:string"/>
<xs:element name="DataSources">
    <xs:complexType>
        <xs:sequence>
            <xs:element minOccurs="0" maxOccurs="1" ref="DataSource"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="DataSource">
    <xs:complexType>
        <xs:all>
            <xs:element minOccurs="1" maxOccurs="1" ref="DriverClass"/>
            <xs:element minOccurs="1" maxOccurs="1" ref="JdbcDriverParams"/>
        </xs:all>
        <xs:attribute name="name" use="required" type="xs:string"/>
    </xs:complexType>
</xs:element>
<xs:element name="DriverClass" type="xs:string"/>
<xs:element name="JdbcDriverParams">
    <xs:complexType>
        <xs:sequence>
            <xs:element minOccurs="1" maxOccurs="1" ref="ConnectionUrl"/>
        </xs:sequence>
    </xs:complexType>

```

## Appendix: Java Server Configuration File Schema

```
</xs:complexType>
</xs:element>
<xs:element name="ConnectionUrl" type="xs:string"/>
<xs:element name="Resources">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="unbounded"
ref="FieldTable16Classes"/>
      <xs:element minOccurs="0" maxOccurs="unbounded"
ref="FieldTable32Classes"/>
      <xs:element minOccurs="0" maxOccurs="unbounded"
ref="ViewFile16Classes"/>
      <xs:element minOccurs="0" maxOccurs="unbounded"
ref="ViewFile32Classes"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="FieldTable16Classes" type="xs:string"/>
<xs:element name="FieldTable32Classes" type="xs:string"/>
<xs:element name="ViewFile16Classes" type="xs:string"/>
<xs:element name="ViewFile32Classes" type="xs:string"/>
<xs:element name="TuxedoServerClasses">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="1" maxOccurs="unbounded"
ref="TuxedoServerClass"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```



```

<xs:element name="TuxedoServerClass">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="unbounded" ref="Services"/>
    </xs:sequence>
    <xs:attribute name="name" use="required" type="xs:string"/>
  </xs:complexType>
</xs:element>

<xs:element name="Services">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="unbounded" ref="Service"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="Service">
  <xs:complexType mixed="true">
    <xs:attribute name="name" use="required" type="xs:string"/>
    <xs:attribute name="target" use="required" type="xs:string"/>
  </xs:complexType>
</xs:element>
</xs:schema>

```

---

## See Also

[TMJAVASVR\(5\)](#)

[Java Server Javadoc](#)

## Appendix: Java Server Configuration File Schema

[Oracle Tuxedo Java Server Administration](#)

[Oracle Tuxedo Java Server Configuration](#)

[Programming an Oracle Tuxedo Application Using Java](#)