Oracle® Tuxedo

Java Server Configuration and Administration Guide

Release 22c
F98206-01
July 2024

ORACLE

Oracle Tuxedo Java Server Configuration and Administration Guide, Release 22c

F98206-01

Primary Author: Preeti Gandhe

Contributing Authors: Tulika Das

Contributors: Maggie Li

Contents

List of Tables

Preface

This document describes how to read the Tuxedo Java server configuration file and its setup.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

1

Oracle Tuxedo Java Server Configuration

This topic contains the following sections:

1.1 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

1.2 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 ensure 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, ensure `$JRE_HOME/lib/amd64/server` is included in `$LD_LIBRARY_PATH`.
- On Oracle Enterprise Linux (64 bit) and Oracle JRockit JVM, ensure `$JRE_HOME/lib/amd64/jrockit` and `$JRE_HOME/lib/amd64` are included in `$LD_LIBRARY_PATH`.
- On AIX PPC64, we recommend you to include both `$JRE_HOME/lib/ppc64/classic` and `$JRE_HOME/lib/ppc64` in `$LIBPATH` library.

- On Solaris sparc 64-bit, ensure `$JRE_HOME/lib/sparcv9/server` is included in `$LD_LIBRARY_PATH`
- On Solaris x64, ensure `$JRE_HOME/lib/amd64/server` is included in `$LD_LIBRARY_PATH`
- On Windows, ensure 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.

1.3 Configurations in UBBCONFIG

You require 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.

The following code sample illustrates how to configure **TMJAVASVR** through the UBBCONFIG file.

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

1.4 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

1.4.1 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

1.4.1.1 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
- java-config
- tux-config
- classpath-config
- tux-resources
- jdbc-resources
- tux-server-config

1.4.1.1.1 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.

The following table 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.

1.4.1.1.2 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. The following table 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 must 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 Exalogic 2.0.

1.4.1.1.3 tux-config

`tux-config` specifies attributes related to Tuxedo applications. The following table 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, `tpsvrinit(String [])` and `tpsvrthrinit(String[])`, 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 `getTuxConfigProperty(K)` interface.

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

1.4.1.1.4 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)`

The following table lists the complete set of `classpath-config` nested elements.

Table 1-4 classpath-config Nested Elements

Name	Type	Range	Descriptions
`classpath`	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 must 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.

1.4.1.1.5 tux-resources

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

Table 1-5 tux-resources Nested Elements

Name	Type	Range	Descriptions
classpath	String	1..511	Optional. It specifies the class paths of Tuxedo typed buffer classes. Note: Users can also specify the class paths of Tuxedo typed buffer classes in classpath-config element.
fld-tbl16-class	String	1..127	Class name list for FML typed buffer.
fld-tbl32-class	String	1..127	Class name list for FML32 typed buffer.
view16-class	String	1..127	Class name list for VIEW typed buffer.
view32-class	String	1..127	Class name list for VIEW32 typed buffer.

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

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.

1.4.1.1.6 jdbc-resources

jdbc-resources specifies the JDBC compatible data source parameters. The following table lists the complete set of jdbc-resources nested elements.

Table 1-6 jdbc-resources Nested Elements

Name	Descriptions
data-source	It includes entries for JDBC data source to create database connection.

The following table *data-source Attributes* and table *data-source Nested Elements* lists the complete set of data-source attributes and nested elements.

Table 1-7 data-source Attributes

Name	Type	Range	Descriptions
vendor	String	1..31	Mandatory. Specifies the driver vendor name.
driver-type	String	1..31	Mandatory. Specifies the driver type.
enabled	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.		

The following table 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.

1.4.1.1.7 tux-server-config

The following table `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`.

The following table 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`.
`maxadmmcmds`	short	1..4096	The maximum number of /Admin commands in the Java server. It is optional. The default value is `128`.

The following table 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 the following table

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

Property	Type	Range	Descriptions
`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`.

The following table list the complete set of `tux-server-config/server-module/server-class/services/service` attributes.

Table 1-14 service Attributes

Property	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.

1.4.1.2 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`.
   The following code sample illustrates how to advertise a Tuxedo Service specified in the configuration file.

```
<TJSconfig version="2.1">
  ...
  <tux-server-config maxmodules="100" maxadmcmds="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 require 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`.

1.4.1.3 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.

The following code sample illustrates an example to create Tuxedo Java server configuration file version 2.1.

```
<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>$USERCLASS/usrlib2/*</classpath>
    <classpath>$USERCLASS/usrlib3/*.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" 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>
```

1.4.1.4 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.

1.4.2 Tuxedo Java Server Configuration File Version 2.0

- Version 2.0 Configuration File Elements
- Version 2.0 Configuration File Example
- Mapping Java Services

1.4.2.1 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
- java-config
- tux-config
- classpath-config
- tux-resources
- jdbc-resources
- tux-server-config

1.4.2.1.1 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. The following table 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.

1.4.2.1.2 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. the following table lists the complete set of `java-config` nested elements.

Table 1-16 java-config Nested Elements

Name	Type	Range	Descriptions
`jvm-optiona`	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 must 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.

1.4.2.1.3 tux-config

`tux-config` specifies attributes related to Tuxedo applications. The following table 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, `tpsvrinit(String [])` and `tpsvrthrinit(String[])`, 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 must be separated by spaces.

1.4.2.1.4 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)`

The following table 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 must 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.

1.4.2.1.5 tux-resources

tux-resources specifies `view/view32` and `fml/fml32` typed buffer classes to instantiate buffer resource objects. The following table 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. Note: Users can also specify the class paths of Tuxedo typed buffer classes in `classpath-config` element.
`fld-tbl16-class`	String	`1..127`	It is the class name list for FML typed buffer.
`fld-tbl32-class`	String	`1..127`	It is the class name list for FML 32 typed buffer.
`view16-class`	String	`1..127`	It is the class name list for VIEW typed buffer.

Table 1-19 (Cont.) tux-resources Nested Elements

Name	Type	Range	Descriptions
view32-class	String	1..127	It is the class name list for VIEW 32 typed buffer.

Users can specify multiple FML/FML32/VIEW/VIEW32 classes in one element; these classes must 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.

1.4.2.1.6 jdbc-resources

jdbc-resources specifies the JDBC compatible data source parameters. The following table lists the complete set of jdbc-resources nested elements.

Table 1-20 jdbc-resources Nested Elements

Name	Descriptions
data-source	It includes entries for JDBC data source to create database connection.

The following table *data-source Attributes* and table *data-source Nested Elements* list the complete set of data-source attributes and nested elements.

Table 1-21 data-source Attributes

Name	Type	Range	Descriptions
vendor	String	1..31	It is mandatory. It specifies the driver vendor name.
driver-type	String	1..31	It is mandatory. It specifies the driver type.
enabled	Boolean	true/false	It is optional; the default value is true. 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 classpath-config 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.		

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.

1.4.2.1.7 tux-server-config

tux-server-config specifies Java server class parameters. The following table 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 classpath-config 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.		

The 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. The following table 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.

1.4.2.2 Version 2.0 Configuration File Example

The following code sample illustrates an example to create Tuxedo Java server configuration file version 2.0.

```
<?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>
    <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>
```

1.4.2.3 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.

1.4.3 Tuxedo Java Server Configuration File Version 1.0

1.4.3.1 Tuxedo Java Server Configuration File Properties

The following table 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 must be in accordance with the format required by the corresponding JDBC driver. It must 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.

Table 1-27 (Cont.) Tuxedo Java Server Configuration File Properties

Property Name	Type	Range	Description
fieldTable16Class	String	Length:0..1024	A comma-separated list of fully qualified FML classes. Any of FML16 field table classes must be defined here or else the Fldid()/ Fname() of the FML will not work.
fieldTable32Class	String	Length:0..1024	A comma-separated list of fully qualified FML classes. All the FML32 field table classes must be defined here or else the Fldid()/ Fname() of the FML32 will not work.
viewFile16Class	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.
viewFile32Class	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.

1.4.3.2 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.

> **Note:**
>
> - 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.

1.4.3.3 Setting the FML/View Field Table Class

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

1.4.3.4 Configuration Rules

You require 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` must 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 must 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.

1.4.3.5 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.

1.4.3.6 Version 1.0 Configuration File Example

It assumes that the `MyTuxedoServerClass1` Java class defines `JAVASTRINGSVC`, `JAVAFML32SVC`, and `svc3` methods, and the `MyTuxedoServerClass2` Java class defines svc4. In this example, a

JDBC connection is made to the Oracle database, and services are exported: `svc1`, `svc2`, `svc3`, and `svc4`.

> **Note:**
>
> 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.

The following code sample illustrates an example to create Tuxedo Java server configuration file version 1.0.

```
<?xml version="1.0" encoding="UTF-8"?>
<TJSconfig>
<ClassPaths>
<ClassPath>/home/oracle/app/javaserver/MyTuxedoServerClass.jar</ClassPath>
<ClassPath>"/home/oracle/jdbc/ucp.jar</ClassPath>
<ClassPath>/home/oracle/jdbc/ojdbc8.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>
                  <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="JAVASTRINGSVC"></Service>
                <Service name="svc2" target="JAVAFML32SVC"></Service>
          </Services>
</TuxedoServerClass>
```

```
<TuxedoServerClass name="MyTuxedoServerClass2"></TuxedoServerClass>
</TuxedoServerClasses>
</TJSconfig>
```

Note:

- 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 require to provide all the libraries (except standard JRE library) used by your Java applications, such as user-defined classes and DataSource driver library.

1.5 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. The following listing 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.

This code sample illustrates how to configure the Tuxedo Transaction Manager.

```
<?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

1.5.1 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 require to be configured in the Java server configuration file. The following listing 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.

The following code sample shows how to specify libraries for the Spring Framework.

```
<?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.

1.6 Transaction Management

The Tuxedo Java-based transaction is implemented based on the existing Tuxedo transaction management framework. As before, the configuration you require 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`) must belong to this group.

The following code sample illustrates UBB configurations for Java-based transactions.

```
*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 require 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 require 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 must 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
```

The following code sample demonstrates the 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>
```


1.6.1 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.

1.6.2 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
PROPAGATION_REQUIRED	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.
PROPAGATION_SUPPORTS	Supports a current transaction, or execute non-transactionally if no transaction exists. It is analogous to EJB transaction attribute of the same name.
PROPAGATION_MANDATORY	Supports a current transaction, or throws an exception if no transaction exists. It is analogous to EJB transaction attribute of the same name.
PROPAGATION_REQUIRES_NEW	Creates a new transaction, or suspends the current transaction if exists. It is analogous to EJB transaction attribute of the same name
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 must explicitly add the transaction begin/end sentences in your scenario.

- Java Server Transaction Manager Default Timeout

1.6.2.1 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.

The following code sample illustrates how to specify 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>
```

1.7 Limitations

- 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 `<TJSconifg>/<tux-resources>` are managed and maintained by Java server itself; they must 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)
- Oracle Tuxedo Java Server Java API Reference, Release 22c
- Oracle Tuxedo Java Server Administration
- Java Server Configuration Schema File Version 1.0
- Oracle® Tuxedo Programming an Oracle Tuxedo Application Using Java

2

Oracle Tuxedo Java Server Administration

This topic contains the following sections:

2.1 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).

2.1.1 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.

The following code sample illustrates how to invoke 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.

The following code sample illustrates how to invoke the /Admin Request in the Java Server /Admin Interface.

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

2.1.2 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:

- tmadmin(1)
- TMJAVASVR(5)
- Oracle Tuxedo Java Server Java API Reference, Release 22c
- Oracle Tuxedo Java Server Administration
- Java Server Configuration Schema File Version 1.0
- Oracle® Tuxedo Programming an Oracle Tuxedo Application Using Java

A

Java Server Configuration Schema File Version 2.1

The following code sample illustrates the Java Server configuration XML schema 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>
        <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                     -->
<!-- ********************************************************* -->
<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: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>
          <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: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">
    <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: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>
            <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>
        <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>
                <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>
             <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">
```

B

Java Server Configuration Schema File Version 2.0

The following code sample illustrates the Java Server configuration XML schema version 2.0.

```
<?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: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: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>
  <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>
</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: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>
          <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>
    </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">
  <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:schema>
```

C

Java Server Configuration Schema File Version 1.0

The following code sample illustrates the Java Server configuration XML schema 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>
```

```
  </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>
```

Glossary

Index