

# Oracle® Tuxedo

SNMP Agent MIB Reference

12c Release 1 (12.1.1)

June 2012

ORACLE®

---

Oracle Tuxedo SNMP Agent MIB Reference, 12c Release 1 (12.1.1)

Copyright © 1996, 2012, 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

## 1. SNMP MIB

What Is an SNMP-Compliant MIB? . . . . .	1-1
MIB Information Structure . . . . .	1-3
MIB Object Identifiers . . . . .	1-3
Absolute and Relative Object Identifiers . . . . .	1-5
Specifying Object Identifiers . . . . .	1-5
SNMP MIB for Oracle Tuxedo 10.0 . . . . .	1-5
SNMP MIB Component MIBs . . . . .	1-6
SNMP MIB Object Naming Conventions . . . . .	1-8
SNMP MIB Object Definitions . . . . .	1-8
SNMP MIB Event Trap Definitions . . . . .	1-9
Using the SNMP MIB . . . . .	1-9
Querying Non-Existent MIB Objects . . . . .	1-10
Updating MIB Objects . . . . .	1-10
Understanding the Differences Between the SNMP MIB and the TMIB . . . . .	1-11

## 2. Core MIB

tuxTBridgeTbl . . . . .	2-2
tuxTclientTbl . . . . .	2-9
tuxTconnTable . . . . .	2-25
tuxTdevice . . . . .	2-31
tuxTdeviceTbl . . . . .	2-32

tuxTdomain	2-35
tuxTgroupTable	2-64
tuxTmachineTable	2-72
tuxTmachineActive	2-88
tuxTmsgTable	2-100
tuxTqueueTable	2-105
tuxTroutingTable	2-110
tuxTsrvrTbl	2-115
tuxTsrvrTblExt	2-131
tuxTsvcTbl	2-144
tuxTsvcGrp	2-151
tuxTlistenTbl	2-158
tuxTranTbl	2-159
tuxTulogTable	2-167
tuxTulogCtrl	2-174
tuxTnetGrpTbl	2-179
tuxTnetMapTbl	2-181
tuxTserverCtxtTbl	2-185
beaEventFilters	2-191
beaEvtFilterTable	2-192

### 3. Domains MIB

tuxDmAcITable	3-2
tuxDmConnectionTable	3-4
tuxDmExportTable	3-6
tuxDmImportTable	3-10
tuxDmLocalTable	3-16
tuxDmOsitpTable	3-25

tuxDmPasswordTable . . . . .	3-30
tuxDmPrincipalMapTable . . . . .	3-32
tuxDmRemoteTable . . . . .	3-35
tuxDmResourcesTable . . . . .	3-38
tuxDmRoutingTable . . . . .	3-39
tuxDmrPrincipalTable . . . . .	3-43
tuxDmSnaCRMTTable . . . . .	3-45
tuxDmSnaLinkTable . . . . .	3-47
tuxDmSnaStackTable . . . . .	3-52
tuxDmTdomainTable . . . . .	3-55
tuxDmTransactionTable . . . . .	3-59

## 4. Oracle Domain List MIB

## 5. CORBA Interface MIB

tuxFactoryTable . . . . .	5-2
tuxInterfaceTable . . . . .	5-4
tuxInterfaceTable . . . . .	5-4
tuxInterfaceTable Objects . . . . .	5-5
tuxLclInterfaceTable . . . . .	5-13
tuxIfQueueTable . . . . .	5-15
tuxLclIfQueueTable . . . . .	5-22

## 6. Access Control List MIB

tuxTAclGrpTable . . . . .	6-2
tuxTAclPermTable . . . . .	6-4
tuxTAclPrinTbl . . . . .	6-6

## 7. Workstation MIB

tuxTwshTbl . . . . .	7-2
tuxTwsITbl . . . . .	7-11

## 8. Application Queue MIB

tuxTAppQctrl . . . . .	8-2
tuxTAppQTbl . . . . .	8-7
tuxTAppQmsgTbl . . . . .	8-19
tuxTQspaceTbl . . . . .	8-27
tuxTQtransTbl . . . . .	8-50

## 9. EventBroker MIB

tuxEventClientTbl . . . . .	9-2
tuxEventCmdTbl . . . . .	9-5
tuxEventQueTbl . . . . .	9-8
tuxEventSvcTbl . . . . .	9-16
tuxEventUlogTbl . . . . .	9-19

## 10. Traps MIB

Specific Trap Number . . . . .	10-1
Variable Bindings . . . . .	10-2
Trap Definitions . . . . .	10-6
DOMAIN Traps . . . . .	10-7
MACHINE Traps . . . . .	10-7
BRIDGE Traps . . . . .	10-13
SERVER Event Traps . . . . .	10-16
CLIENT Traps . . . . .	10-21
TRANSACTION Traps . . . . .	10-23
EVENT Traps . . . . .	10-25

# SNMP MIB

The following sections define an SNMP-compliant MIB and introduce the SNMP MIB for Oracle Tuxedo 10.0:

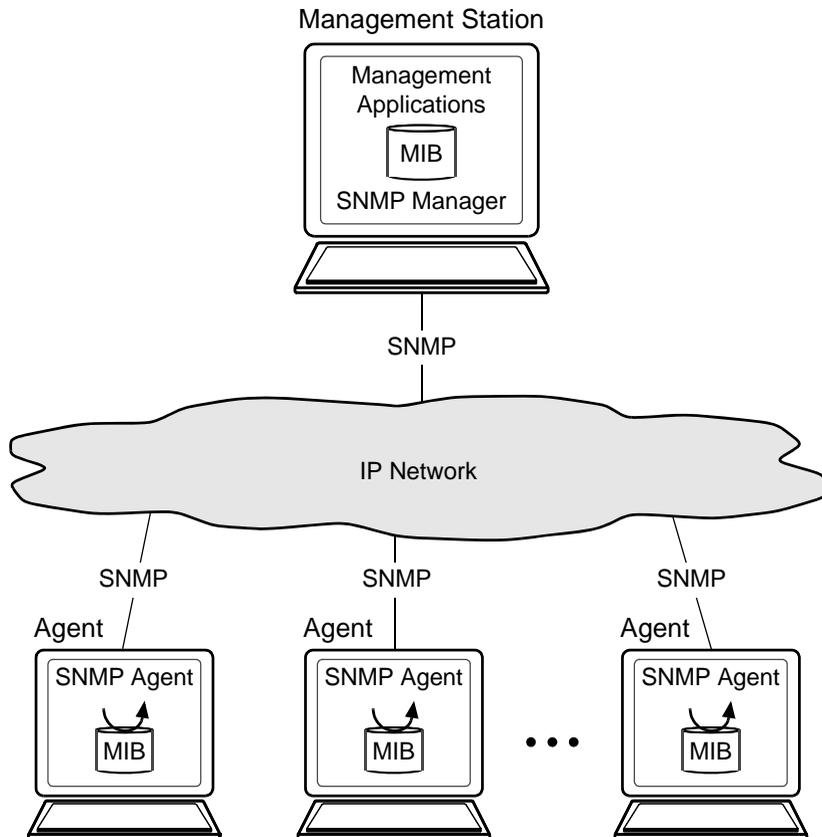
- [What Is an SNMP-Compliant MIB?](#)
- [MIB Information Structure](#)
- [MIB Object Identifiers](#)
- [SNMP MIB for Oracle Tuxedo 10.0](#)
- [SNMP MIB Component MIBs](#)
- [SNMP MIB Object Naming Conventions](#)
- [SNMP MIB Object Definitions](#)
- [SNMP MIB Event Trap Definitions](#)
- [Using the SNMP MIB](#)
- [Understanding the Differences Between the SNMP MIB and the TMIB](#)

## What Is an SNMP-Compliant MIB?

Each management station or agent in an SNMP-managed network maintains a local database of information relevant to network management, known as the management information base

(MIB). The relationship between the management station, the agent, and the MIB is shown in Figure 1-1.

Figure 1-1 SNMP-Managed Configuration



An SNMP-compliant MIB contains definitions and information about the properties of managed resources and the services that the agents support. The manageable features of resources, as defined in an SNMP-compliant MIB, are called *managed objects* or *management variables* (or just *objects* or *variables*).

A management station gets and sets objects in the MIB, and an agent notifies the management station of significant but unsolicited events called *traps*. All message exchanges between the management station and its agents take place using the Simple Network Management Protocol (SNMP).

The MIB at the management station contains network management information extracted from the MIBs of all the managed entities in the network.

## MIB Information Structure

The structure of management information (SMI), an SNMP standard described in the NWG RFC 1155, defines the structure of the MIB information and the allowable data types. The SMI identifies how resources within the MIB are represented and named. The philosophy behind SMI is to encourage simplicity and extensibility within the MIB.

The SNMP specification includes a template, known as an Abstract Syntax Notation One (ASN.1) OBJECT TYPE macro, which provides the formal model for defining objects and tables of objects in the MIB. The following keywords are used to define a MIB object:

### **Syntax**

Defines the abstract data structure corresponding to the object type. The SMI purposely restricts the ASN.1 constructs that can be used to promote simplicity.

### **Access**

Defines whether the object value may only be retrieved but not modified (read-only) or whether it may also be modified (read-write).

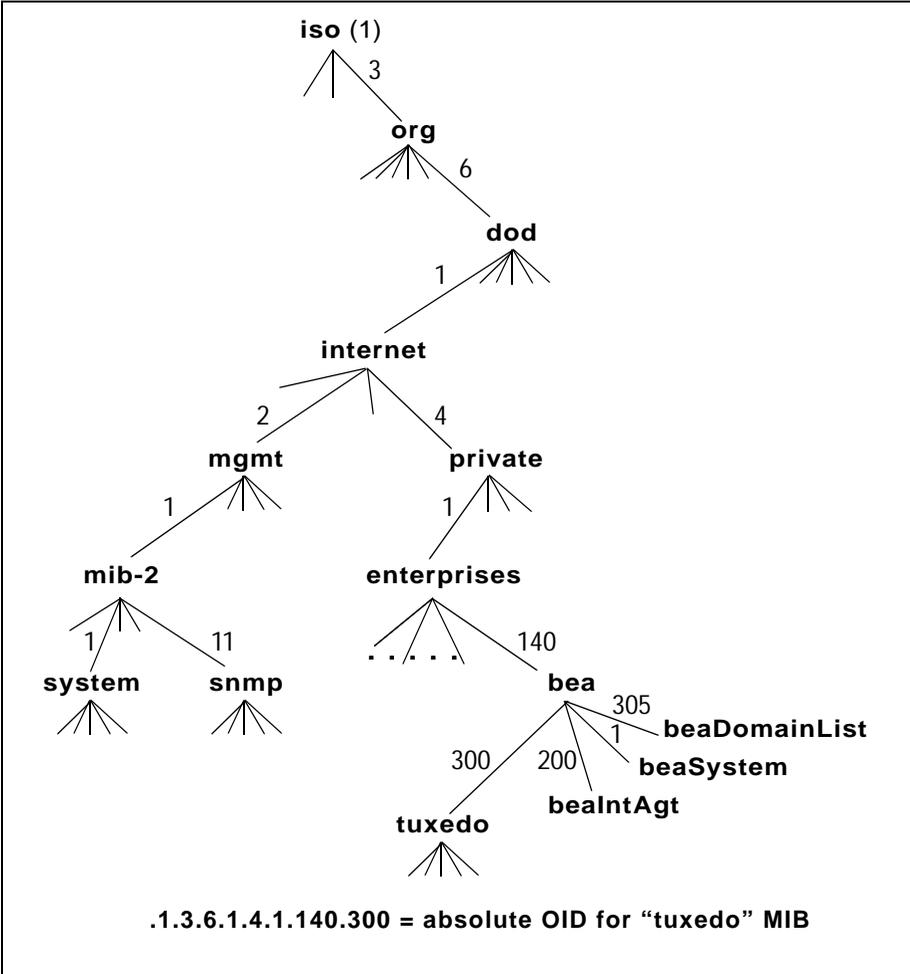
### **Description**

Contains a textual definition of the object type. The definition provides all semantic definitions necessary for interpretation; it typically contains information of the sort that would be communicated in any ASN.1 commentary annotations associated with the object.

## MIB Object Identifiers

Each object in the MIB has an *object identifier* (OID), which the management station uses to request the object's value from the agent. An OID is a sequence of integers that uniquely identifies a managed object by defining a path to that object through a tree-like structure called the *OID tree* or registration tree. When an SNMP agent needs to access a specific managed object, it traverses the OID tree to find the object. The MIB object identifier hierarchy and format is shown in [Figure 1-2](#).

Figure 1-2 SNMP MIB Object Identifier Hierarchy and Format



In this hierarchy, each Oracle private MIB object that the Oracle SNMP Agent software manages has a unique object identifier. A prefix of .1.3.6.1.4.1.140 points to the objects in the Oracle private MIB for the Oracle SNMP Agent software.

## Absolute and Relative Object Identifiers

*Absolute OIDs* specify a path to an attribute from the root of the OID tree. Absolute OID names always begin with a dot and must specify every node of the OID tree from the top-most node to the specific managed object. For example:

```
.1.3.6.1.2.1.1.1
```

*Relative OIDs* specify a path to the attribute relative to some node in the OID tree. For example, 2.1.1.1 specifies the `sysDescr` object in the `system` group, relative to the Internet node in the OID tree.

## Specifying Object Identifiers

In addition to using the “dot-dot” notation, a series of integers separated by dots to describe OIDs, you can also express OIDs by using textual symbols instead of numbers to represent nodes in the path to the object, or by using a combination of both integers and textual symbols. A *symbolic* OID uses mnemonic keywords to specify the managed object. For example:

```
mgmt.mib-2.system.sysContact
```

The following numeric OID uses integers to specify the same managed object:

```
2.1.1.7
```

Note that 2.1.1.7 in this example is a relative OID.

An OID can combine both symbolic and numeric representations of individual nodes of the OID tree; for example:

```
mgmt.mib-2.1.sysContact
```

## SNMP MIB for Oracle Tuxedo 10.0

The SNMP MIB for Oracle Tuxedo 10.0 is essentially an SNMP version of the Tuxedo management information base (TMIB) for Oracle Tuxedo 10.0. The TMIB is the standard MIB for administering the components of an Oracle Tuxedo or Oracle WebLogic Enterprise application. For more information about the TMIB, see [“Understanding the Differences Between the SNMP MIB and the TMIB” on page 1-11](#).

The SNMP MIB defines the data types and access permissions for the various managed objects that can be accessed through the Oracle SNMP Agent software. It also defines the event notifications that can be generated by the Oracle SNMP Agent software. As required by the

SNMP standard, the SNMP MIB definitions are written in concise MIB format in accordance with RFC 1212.

Oracle SNMP Agent provides a file named `bea.asn1` for defining the SNMP MIB. For an Oracle Tuxedo 10.0 installation, the `bea.asn1` file resides in the `tux_prod_dir/udataobj/snmp/etc` directory, where `tux_prod_dir` represents the directory in which the Oracle Tuxedo 10.0 distribution is installed.

The `bea.asn1` file for Oracle Tuxedo 10.0 makes the features of the following components recognizable and thus manageable within an SNMP network management framework:

- Tuxedo 10.0 components
- Tuxedo 9.1 components
- Tuxedo 9.0 components
- Tuxedo 8.1 components
- Tuxedo 8.0 components
- Tuxedo 7.1 components
- Tuxedo 6.5 components

You can use Oracle SNMP Agent and the SNMP MIB for Oracle Tuxedo 10.0 to manage Tuxedo 9.1, 9.0, 8.1, 8.0, 7.1, and 6.5 applications. You cannot use Oracle SNMP Agent and the SNMP MIB for Oracle Tuxedo 10.0 to manage WebLogic Enterprise 5.1 or earlier applications.

## SNMP MIB Component MIBs

The SNMP MIB defined by the `bea.asn1` file for Oracle Tuxedo 10.0 refers to the entire database of management information at the management station or agent. The SNMP MIB, itself, consists of distinct component MIBs, each of which refers to a specific defined collection of management information that is part of the overall SNMP MIB. The management station uses the component MIBs to administer the particular components of the Oracle Tuxedo system, to administer the agents themselves, and to collect information about the managed resources.

The SNMP MIB for Oracle Tuxedo 10.0 consists of the following component MIBs:

- *Core MIB*—OID prefix: `.1.3.6.1.4.1.140.300` (or `tuxedo`)—Contains the MIB objects for controlling the operation and configuration of a Tuxedo application. This MIB contains the main information groups for Tuxedo applications, including domains, machines,

queues, servers, routing, clients, and services. For a detailed description, see [“Core MIB” on page 2-1](#).

- *Domains MIB*—OID prefix: .1.3.6.1.4.1.140.300 (or `tuxedo`)—Contains the MIB objects for describing the interaction between Tuxedo *domains* (Tuxedo business applications). For a detailed description, see [“Domains MIB” on page 3-1](#).
- *Oracle Domain List MIB*—OID prefix: .1.3.6.1.4.1.140.305 (or `beaDomainList`)—Contains the MIB objects for identifying and describing all Tuxedo domains currently being monitored on a particular managed node (machine). For a detailed description, see [“Oracle Domain List MIB” on page 4-1](#).
- *CORBA Interface MIB*—OID prefix: .1.3.6.1.4.1.140.300 (or `tuxedo`)—Contains the MIB objects for managing Tuxedo 8.0 or later CORBA features. For a detailed description, see [“CORBA Interface MIB” on page 5-1](#).
- *Access Control List MIB*—OID prefix: .1.3.6.1.4.1.140.300 (or `tuxedo`)—Contains the MIB objects for setting and controlling the security options for the Tuxedo application. For a detailed description, see [“Access Control List MIB” on page 6-1](#).
- *Workstation MIB*—OID prefix: .1.3.6.1.4.1.140.300 (or `tuxedo`)—Contains the MIB objects for specifying information about Tuxedo client workstations including workstation listeners and handlers. For a detailed description, see [“Workstation MIB” on page 7-1](#).
- *Application Queue MIB*—OID prefix: .1.3.6.1.4.1.140.300 (or `tuxedo`)—Contains the MIB objects for managing access to Tuxedo application queues. The groups include objects for managing queue spaces, queues, messages, and transactions. For a detailed description, see [“Application Queue MIB” on page 8-1](#).
- *EventBroker MIB*—OID prefix: .1.3.6.1.4.1.140.300 (or `tuxedo`)—Contains the MIB objects for describing current event subscriptions, defining new subscriptions, or invalidating subscriptions. For a detailed description, see [“EventBroker MIB” on page 9-1](#).
- *Traps MIB*—OID prefix: .1.3.6.1.4.1.140.300 (or `tuxedo`)—Contains the MIB objects for specifying the trap notifications generated by the SNMP agent for Oracle SNMP Agent, and for specifying the objects passed in the variable bindings for the traps. For a detailed description, see [“Traps MIB” on page 10-1](#).
- *Oracle System MIB*—OID prefix: .1.3.6.1.4.1.140.1 (or `beaSystem`)—Contains the MIB objects for passing the trap notifications generated by the Oracle SNMP Agent Integrator polling rules. As an example, a rule-action might specify that when the value of the polled object at OID .1.3.6.1.4.1.140.1.0 is greater than 20, send a trap with a specific trap ID of 200; when the object’s value becomes less than 20, send a trap with a specific Trap ID of 300. For a description of the Oracle SNMP Agent Integrator polling

feature, see [“Using the Oracle SNMP Agent Integrator for Polling”](#) in the *Oracle Tuxedo SNMP Agent Administration Guide*.

- *Oracle Agent Integrator MIB*—OID prefix: .1.3.6.1.4.1.140.200 (or beaIntAgt)—Contains the MIB objects for creating user-defined traps that are generated by the Oracle SNMP Agent Integrator according to user-defined polling rules. You can configure the Oracle SNMP Agent Integrator running on the managed node to perform local polling and generate SNMP trap notifications, or execute a system command when certain conditions are met. Individual rules, stored as MIB objects, can be activated and deactivated by the management station. For a description of polling rules, see [“Configuration Files”](#) in the *Oracle Tuxedo SNMP Agent Administration Guide*.

With the exception of *Oracle Domain List*, *Traps*, *Oracle System*, and *Oracle Agent Integrator*, the SNMP MIB component MIBs correspond to the TMIB component MIBs. For more information about the TMIB, see [“Understanding the Differences Between the SNMP MIB and the TMIB”](#) on page 1-11.

## SNMP MIB Object Naming Conventions

The object names within the SNMP MIB for Oracle Tuxedo 10.0 are prefixed with the letters `tux`. For example, the Core MIB contains a group named `tuxTmachineTable`, and the following objects are included within the `tuxTmachineTable` group:

`tuxTmachinePmid`

Represents a physical machine identifier

`tuxTmachineLmid`

Represents the logical machine identifier

## SNMP MIB Object Definitions

The SNMP MIB definitions are written in concise MIB format in accordance with RFC 1212. Thus, the SNMP MIB stores only simple data types: scalars and two-dimensional arrays of scalars, called *tables*. Keywords SYNTAX, ACCESS, and DESCRIPTION as well as other keywords such as STATUS and INDEX are used to define the SNMP MIB managed objects.

To monitor or modify values of managed objects through your management station, you need to know which MIB objects represent the features of the Oracle Tuxedo resources that are relevant to your management goals. You also need to know the data types, default values, and access permissions for these MIB objects.

For table objects, keep the following tips in mind:

- In some cases a read-write table object can only be set during creation of a new row. Where true, this information is noted in the DESCRIPTION section for that object.
- Each row in a table is an instance of the Entry object under that table. The DESCRIPTION section for the Entry object under a table (such as `tuxTmachineTable`) contains information on the columnar values that are minimally necessary for creation of a row—how a new row is created, whether the values pertain only to the local machine, and other pertinent information about the table objects.

## SNMP MIB Event Trap Definitions

The `bea.asn1` file defines a full range of Oracle Tuxedo system and application events in accordance with RFC 1215, Trap definitions. These system and application events are transmitted as enterprise-specific traps to the management station. For a list of these traps, see [“Traps MIB” on page 10-1](#).

The following keywords are used to define a trap:

### ENTERPRISE

An object identifier that specifies the management enterprise under whose registration authority this trap is defined. All traps generated by the SNMP agent for Oracle SNMP Agent have an enterprise field set to the following OID: `.1.3.6.1.4.1.140.300`. This value is passed in the `enterprise` field of the trap packet (Protocol Data Unit—PDU).

### VARIABLES

Defines the ordered sequence of MIB objects that are contained in each instance of the trap type. Each variable is placed, in order, inside the variable-bindings field of the SNMP trap packet (PDU).

### DESCRIPTION

Contains a textual definition of the trap type.

### Trap ID

Specifies the enterprise-specific trap ID for the trap definition. The trap ID is passed in the specific trap ID field of the trap packet (PDU). The value of the generic trap ID field in traps is always set to 6, indicating an enterprise-specific trap.

## Using the SNMP MIB

The management station uses the `bea.asn1` file to set up the SNMP MIB for Oracle SNMP Agent on the management station. The `bea.asn1` file must be imported into the management

database of the management station, as described in [“Using the Oracle SNMP Agent with a Management Framework”](#) in *Oracle Tuxedo SNMP Agent Administration Guide*.

The SNMP agent for Oracle SNMP Agent uses a file named `mib.txt` to set up its local SNMP MIB on the managed node (machine). The `mib.txt` file, similar to the `bea.asn1` file, provides a textual description of the content of the SNMP MIB. By default, the `mib.txt` file resides in the `tux_prod_dir/udataobj/snmp/etc` directory, where `tux_prod_dir` represents the directory in which the Oracle Tuxedo 10.0 distribution is installed. For more information about using the `mib.txt` file to create the local SNMP MIB on a managed node, see [“Oracle SNMP Agent Integrator Commands”](#) in *Oracle SNMP Tuxedo Agent Administration Guide*.

The SNMP agent communicates with the TMIB of the managed Oracle Tuxedo application to get the object values that initially populate the local SNMP MIB. As the management station gets and sets object values in the local SNMP MIB through the SNMP agent, the SNMP agent issues Tuxedo commands to read and write the comparable object values in the local TMIB.

The local SNMP MIB is not persistent, meaning that the SNMP MIB is not written to disk. When the SNMP agent process terminates, its SNMP MIB also terminates.

## Querying Non-Existent MIB Objects

If you attempt to retrieve the value for an SNMP MIB object that does not exist, either no value is returned, or one of the following values is returned:

- -1 if the object is numeric
- A dash (-) if the object data type is `DisplayString`

For example, if an Oracle Tuxedo 8.0 or later application is not installed on the managed node, the CORBA-specific objects included in the SNMP MIB for Oracle Tuxedo 10.0 do not return values when queried.

## Updating MIB Objects

Some objects in the SNMP MIB can be set (updated) only under certain states of the Oracle Tuxedo system. If you get an error while trying to set read-write objects in this MIB, examine the Tuxedo `ULOG` file for more information about the error.

The Tuxedo system creates a new `ULOG` file each day on each machine in a Tuxedo domain. For a description of the `ULOG` file, see reference page [`userlog\(3c\)`](#).

## Understanding the Differences Between the SNMP MIB and the TMIB

The primary difference between the SNMP MIB for Oracle SNMP Agent and the Tuxedo MIB (TMIB) is the use of terms. In addition, the SNMP MIB contains a few additional component MIBs.

The TMIB for an Oracle Tuxedo system consists of distinct component MIBs, each used to administer a particular component of the Tuxedo system. These component MIBs are defined in individual reference pages each addressing the MIB for a particular part of the system. For example, reference page [TM\\_MIB \(5\)](#) in *Oracle Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference* defines the MIB used to administer the fundamental aspects of an Oracle Tuxedo 10.0 application. TM\_MIB is comparable to the SNMP Core MIB.

Instead of referring to *groups* and *managed objects*, as is common in SNMP terminology, the TMIB defines application resources as *classes* and *attributes*. Classes are the administrative class definitions that make up the TMIB. Each class has a set of attributes that identifies individual items in the class. Examples of TMIB classes are:

T\_MACHINE

The class definition for a machine

T\_SERVICE

The class definition for Tuxedo services

Attributes for these classes are identified by the prefix TA\_ followed by the attribute name. A few examples for the T\_MACHINE class are:

TA\_P MID

Represents a physical machine name

TA\_L MID

Represents the logical machine name

For more information about the TMIB, refer to *Oracle Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*.



# Core MIB

The Core MIB defines the groups through which the fundamental aspects of an application can be configured and managed. These MIB groups contain objects for managing machines, servers, networking, and load balancing.

The Core MIB defines the basic objects that form an Oracle Tuxedo application. It is the main information repository for controlling the operation and configuration of the application. When an application is active, the Core MIB contains groups related to the run-time activity of your application. You can use this information to monitor the behavior of your application.

The Core MIB consists of the following groups.

Group Name	Description
<a href="#">tuxTBridgeTbl</a>	Network connection
<a href="#">tuxTclientTbl</a>	Client
<a href="#">tuxTconnTable</a>	Conversation
<a href="#">tuxTdevice</a>	Device
<a href="#">tuxTdomain</a>	Domain information
<a href="#">tuxTgroupTable</a>	Server group
<a href="#">tuxTmachineTable</a>	Machine configuration objects
<a href="#">tuxTmachineActive</a>	Run-time machine characteristics

Group Name	Description
<code>tuxTmsgTable</code>	Message queue
<code>tuxTqueueTable</code>	Server queue
<code>tuxTroutingTable</code>	Routing criteria
<code>tuxTsrvrTbl</code>	Server configuration objects
<code>tuxTsrvrTblExt</code>	Server run-time characteristics
<code>tuxTsvcTbl</code>	Service
<code>tuxTsvcGrp</code>	Service-group configuration objects
<code>tuxTlistenTbl</code>	Tuxedo listeners
<code>tuxTranTbl</code>	Transaction
<code>tuxTulogTable</code>	Userlog
<code>tuxTulogCtrl</code>	Control filter MIB for <code>tuxTulogTable</code>
<code>tuxTnetMapTbl</code>	Maps logical machine IDs to network groups
<code>tuxTnetGrpTbl</code>	Application objects of network groups
<code>tuxTserverCtxtTbl</code>	Configuration and run-time objects of individual server dispatch contexts
<code>beaEventFilters</code>	Used to define a subset of Tuxedo event notifications

## tuxTBridgeTbl

The `tuxTBridgeTbl` group contains objects that represent run-time characteristics pertaining to connectivity between logical machines that make up an application. The object values represent connection status and statistics.

Objects in this table are accessible either through a Tuxedo SNMP agent installed on the local machine or using the `-c` option on the master machine. The index into the table consists of `tuxTBridgeLmid` and `tuxTBridgeNetworkGrpNo`. In Tuxedo 6.4, SET requests are allowed only for the `DEFAULTNET` network group, so all SET requests should use 0 for `tuxTBridgeNetworkGrpNo` in the SNMP index.

Object Name	Object ID
tuxTBridgeLmid	.1.3.6.1.4.1.140.300.16.1.1.1
tuxTBridgeState	.1.3.6.1.4.1.140.300.16.1.1.2
tuxTBridgeCurTime	.1.3.6.1.4.1.140.300.16.1.1.3
tuxTBridgeConTime	.1.3.6.1.4.1.140.300.16.1.1.4
tuxTBridgeSuspTime	.1.3.6.1.4.1.140.300.16.1.1.5
tuxTBridgeRcvdByte	.1.3.6.1.4.1.140.300.16.1.1.6
tuxTBridgeSentByte	.1.3.6.1.4.1.140.300.16.1.1.7
tuxTBridgeRcvdNum	.1.3.6.1.4.1.140.300.16.1.1.8
tuxTBridgeSentNum	.1.3.6.1.4.1.140.300.16.1.1.9
tuxTBridgeFlowCnt	.1.3.6.1.4.1.140.300.16.1.1.10
tuxTBridgeCurEncryptBits	.1.3.6.1.4.1.140.300.16.1.1.11
tuxTBridgeNetworkGrpNo	.1.3.6.1.4.1.140.300.16.1.1.12
tuxTBridgeNetworkGrpName	.1.3.6.1.4.1.140.300.16.1.1.13

## tuxTBridgeLmid

### Syntax

*DisplayString* (SIZE(1..61))

### Access

read-only

### Description

*DisplayString* is of the format: *LMID1*[, *LMID2*]

*LMID1*

Is the logical machine identifier for network connection and is in the range from one to 61 characters.

### *LMID2*

Is the destination logical machine identifier for network connection and is in the range from one to 61 characters.

## **tuxTBridgeState**

### Syntax

```
INTEGER {active(1), inactive(2), suspended(3), pending(4)}
```

### Access

read-write

### Description

The values for GET and SET operations are as follows:

```
GET: {active(1) | inactive(2) | suspended(3) | pending(4)}
```

A GET operation retrieves run-time information for the selected `tuxTBridgeTbl` instance(s). A `tuxTBridgeLmid` object value with only one logical machine identifier matches all active connections from *LMID1* to other machines in the application. In this case, each retrieved record contains an expanded `tuxTBridgeLmid` object value with the destination LMID filled in. The following states indicate the meaning of a `tuxTBridgeState` returned in response to a GET request. States not listed are not returned.

`active(1)`

The connection is established and active.

`inactive(2)`

The connection is inactive. This state is only returned when status is requested on a particular connection, that is, both LMIDs are specified in the `tuxTBridgeLmid` object and the source logical machine is reachable.

`suspended(3)`

An established connection has been terminated due to an error condition, and reconnection has been suspended for at least the amount of time indicated in the `tuxTBridgeSuspTime` object value.

`pending(4)`

An asynchronous connection has been requested but has not yet completed. The final outcome of the connection request has not been determined.

SET: {active(1) | inactive(2) | suspended(3) | pending(4)}

A SET operation updates run-time information for the selected tuxTbridgeTbl object. The following states indicate the meaning of a tuxTbridgeState set in a SET request. States not listed cannot be set.

active(1)

Activate the tuxTbridgeTbl instance by establishing an asynchronous connection between the indicated logical machines. This operation fails if only one machine is specified, if either of the machines is not active, or if the source machine is not reachable. When in the pending(4) state, the success or failure of the connection has not yet been determined. The BRIDGE can continue to process other events and data while the connection is outstanding. This state change is allowed in the inactive(2) and suspended(3) states. Successful return leaves the instance in the active(1) or pending(4) state.

inactive(2)

Deactivate the tuxTbridgeTbl object by closing the connection between the indicated logical machines. This operation fails if only one logical machine is specified or if the two machines are not connected. State change allowed only when in the active(1) state. Successful return leaves the object in the inactive(2) state.

suspended(3)

Suspend the tuxTbridgeTbl object by closing the connection between the indicated logical machines and by setting the tuxTbridgeSuspTime parameter as indicated. State change allowed only when in the active(1) state. Successful return leaves the object in the suspended(3) state.

**Note:** Since the statistics reported are from the source logical machine, resetting those statistics causes them to be out of sync with the statistics reported by the destination logical machine for the same connection.

pending(4)

Activate the tuxTbridgeTbl instance by establishing an asynchronous connection between the indicated logical machines. This operation fails if only one logical machine is specified, if either of the two machines is inactive, or if the source logical machine is not reachable. When in the pending(4) state, the success or failure of the connection request has not yet been determined. However, the BRIDGE can continue to process other events and data while the connection request is outstanding. State change allowed in inactive(2) and suspended(3) states. Successful return leaves the instance in the pending(4) state.

## **tuxTBridgeCurTime**

### Syntax

INTEGER

### Access

read-only

### Description

Current time, in seconds, since 00:00:00 UTC, January 1, 1970, as returned by the `time(2)` system call on `tuxTBridgeLmid`. This object can be used to compute elapsed time from the following “tuxTBridge” object values.

## **tuxTBridgeConTime**

### Syntax

INTEGER

### Access

read-only

### Description

Time, in seconds, that this connection has been active.

## **tuxTBridgeSuspTime**

### Syntax

INTEGER

### Access

read-write

### Description

Time, in seconds, remaining in the suspension of this connection. After this amount of time, the connection automatically changes to a `tuxTBridgeState` of `inactive(2)` and can be activated by normal application traffic.

## **tuxTBridgeRcvdByte**

### Syntax

INTEGER

### Access

read-only

### Description

Number of bytes sent from the destination logical machine to the source logical machine.

## **tuxTBridgeSentByte**

### Syntax

INTEGER

### Access

read-only

### Description

Number of bytes sent from the source logical machine to the destination logical machine.

## **tuxTBridgeRcvdNum**

### Syntax

INTEGER

### Access

read-only

### Description

Number of messages sent from the destination logical machine to the source logical machine.

## **tuxTBridgeSentNum**

### Syntax

INTEGER

### Access

read-only

### Description

Number of messages sent from the source logical machine to the destination logical machine.

## **tuxTBridgeFlowCnt**

### Syntax

INTEGER

### Access

read-only

### Description

Number of times flow control has been encountered over this connection.

## **tuxTBridgeCurEncryptBits**

### Syntax

INTEGER {none(1), 40-bit(2), 128-bit(3), not-available(4)}

### Access

read-only

### Description

The current level of encryption for this link. The `tuxTBridgeCurEncryptBits` value is negotiated between the machines when the link is established. The number specifies the encryption key length (in bits).

## tuxTBridgeNetworkGrpNo

### Syntax

INTEGER

### Access

read-only

### Description

Logical network group number. When both the source and destination `tuxTBridgeLmid` machine identifiers are in the same network group, `tuxTBridgeTbl` presents all instances of related fields per network group.

## tuxTBridgeNetworkGrpName

### Syntax

DisplayString

### Access

read-only

### Description

Logical network group name.

## tuxTclientTbl

The `tuxTclientTbl` group contains objects that represent run-time characteristics of active clients within an application. The object values identify and track the activity of clients within a running application. Objects in this table are only accessible through a Tuxedo SNMP agent installed on the local machine.

Object Name	Object ID
<code>tuxTclientState</code>	<code>.1.3.6.1.4.1.140.300.17.1.1.1</code>
<code>tuxTclientBirthTime</code>	<code>.1.3.6.1.4.1.140.300.17.1.1.2</code>

<b>Object Name</b>	<b>Object ID</b>
tuxTclientMachineId	.1.3.6.1.4.1.140.300.17.1.1.3
tuxTclientReg	.1.3.6.1.4.1.140.300.17.1.1.4
tuxTclientClntName	.1.3.6.1.4.1.140.300.17.1.1.5
tuxTclientIdleTime	.1.3.6.1.4.1.140.300.17.1.1.6
tuxTclientPid	.1.3.6.1.4.1.140.300.17.1.1.7
tuxTclientSrvGrp	.1.3.6.1.4.1.140.300.17.1.1.8
tuxTclientUsrName	.1.3.6.1.4.1.140.300.17.1.1.9
tuxTclientWsc	.1.3.6.1.4.1.140.300.17.1.1.10
tuxTclientWsh	.1.3.6.1.4.1.140.300.17.1.1.11
tuxTclientWshClientId	.1.3.6.1.4.1.140.300.17.1.1.12
tuxTclientRelease	.1.3.6.1.4.1.140.300.17.1.1.13
tuxTclientWsProto	.1.3.6.1.4.1.140.300.17.1.1.14
tuxTclientNumConv	.1.3.6.1.4.1.140.300.17.1.1.15
tuxTclientNumDeque	.1.3.6.1.4.1.140.300.17.1.1.16
tuxTclientNumEnque	.1.3.6.1.4.1.140.300.17.1.1.17
tuxTclientNumPost	.1.3.6.1.4.1.140.300.17.1.1.18
tuxTclientNumReq	.1.3.6.1.4.1.140.300.17.1.1.19
tuxTclientNumSubscribe	.1.3.6.1.4.1.140.300.17.1.1.20
tuxTclientNumTran	.1.3.6.1.4.1.140.300.17.1.1.21
tuxTclientNumTranAbt	.1.3.6.1.4.1.140.300.17.1.1.22
tuxTclientNumTranCmt	.1.3.6.1.4.1.140.300.17.1.1.23
tuxTclientCmtRet	.1.3.6.1.4.1.140.300.17.1.1.24
tuxTclientCurConv	.1.3.6.1.4.1.140.300.17.1.1.26

Object Name	Object ID
tuxTclientCurReq	.1.3.6.1.4.1.140.300.17.1.1.27
tuxTclientCurTime	.1.3.6.1.4.1.140.300.17.1.1.28
tuxTclientLastGrp	.1.3.6.1.4.1.140.300.17.1.1.29
tuxTclientNaddr	.1.3.6.1.4.1.140.300.17.1.1.30
tuxTclientNotify	.1.3.6.1.4.1.140.300.17.1.1.31
tuxTclientNumUnSol	.1.3.6.1.4.1.140.300.17.1.1.32
tuxTclientRpid	.1.3.6.1.4.1.140.300.17.1.1.33
tuxTclientTimeLeft	.1.3.6.1.4.1.140.300.17.1.1.34
tuxTclientTimeStart	.1.3.6.1.4.1.140.300.17.1.1.36
tuxTclientTranLev	.1.3.6.1.4.1.140.300.17.1.1.37
tuxTclientId	.1.3.6.1.4.1.140.300.17.1.1.38
tuxTclientContextID	.1.3.6.1.4.1.140.300.17.1.1.50

## tuxTclientState

### Syntax

```
INTEGER { active(1), suspended(2), dead(3) }
```

### Access

read-write

### Description

The values for GET and SET operations are as follows:

```
GET: { active(1) | suspended(2) | dead(3) }
```

A GET operation retrieves run-time information for the selected tuxTclientTbl instance(s). Note that client information is kept in local bulletin board tables only. Therefore, for maximum performance, inquiries on client status should be restricted, using key fields as much as possible. The following states indicate the meaning of a

`tuxTclientState` returned in response to a GET request. States not listed are not returned.

`active(1)`

`tuxTclientTbl` instance active. The `active(1)` state is not an indication of whether the client is idle or busy. A non-0 value retrieved for either the `tuxTclientCurConv` object or the `tuxTclientCurReq` object indicates a busy client.

`suspended(2)`

`tuxTclientTbl` instance active and suspended from making further service requests (`tpcall(3)` or `tpacall(3)`) and from initiating further conversations (`tpconnect(3)`). See SET `suspended(2)` below for details.

`dead(3)`

`tuxTclientTbl` instance identified as active in the bulletin board but currently not running due to an abnormal death. This state exists only until the BBL local to the client notices the death and takes action to clean up the client's bulletin board resources.

SET: {`active(1)` | `suspended(2)` | `dead(3)`}

A SET operation updates run-time information for the selected `tuxTclientTbl` object. The following states indicate the meaning of a `tuxTclientState` set in a SET request. States not listed cannot be set.

`active(1)`

Activate a `suspended(2)` `tuxTclientTbl` instance. State change allowed only when in the `suspended(2)` state. Successful return leaves the object in the `active(1)` state.

`suspended(2)`

Suspend the `tuxTclientTbl` instance from making service requests (`tpcall(3)` or `tpacall(3)`), initiating conversations (`tpconnect(3)`), beginning transactions (`tpbegin(3)`), and enqueueing new requests (`tpenqueue(3)`). Clients within a transaction are permitted to make these calls until they abort or commit the current transaction, at which time the clients become suspended. Invocations of these routines result in a TPESYSTEM error return and a system log message being generated that indicates the situation. State change is allowed only when the object is in the `active(1)` state. Successful return leaves the object in the `suspended(2)` state.

`dead(3)`

Abortively deactivate the `tuxTclientTbl` instance. State change is allowed only when the object is in the `active(1)` or `suspended(2)` state. The recommended method for deactivating clients is to first suspend them, and then to abortively deactivate them by setting the state to `dead(3)`. Successful return leaves the object in the `dead(3)` state.

**Note:** Workstation handlers (`tuxTclientWsh == yes(1)`) cannot be set to a state of `dead(3)`. The system might not be able to kill the client, due to platform or signaling

restrictions. In this case, a native client is abortively terminated at its next access to ATMI, and a workstation client's connection to a WSH is preemptively torn down.

## **tuxTclientBirthTime**

### Syntax

INTEGER

### Access

read-only

### Description

Client identifier. The data in this field should not be interpreted directly by the end user except for equality comparison.

## **tuxTclientMachineld**

### Syntax

INTEGER

### Access

read-only

### Description

Client identifier. The data in this field should not be interpreted directly by the end user except for equality comparison.

## **tuxTclientReg**

### Syntax

INTEGER

### Access

read-only

## Description

Client identifier. The data in this field should not be interpreted directly by the end user except for equality comparison.

## **tuxTclientCntName**

### Syntax

*DisplayString*(SIZE(0..30))

### Access

read-only

## Description

Client name associated with client at `tpinit(3)` time through the `cltname` element of the TPINIT structure.

## **tuxTclientIdleTime**

### Syntax

INTEGER

### Access

read-only

## Description

Approximate amount of time, in seconds, since this client last interacted with the system through an ATMI call. This value is accurate to within `tuxTdomainScanUnit` (see the `tuxTdomain` group) seconds. When specified as a key field, a positive value indicates that all clients with idle times of at least the indicated value match, a negative value indicates that all clients with no more than the indicated value match, and a 0 value matches all clients.

## **tuxTclientPid**

### Syntax

INTEGER

## Access

read-only

## Description

Process identifier of client. Note that for workstation clients, this identifier indicates the workstation handler through which the workstation client is connected. A negative number can be specified on a `GET` operation for the purpose of retrieving client information for the calling process. If the calling process is not a client, then an error is returned.

## **tuxTclientSrvGrp**

### Syntax

*DisplayString*(SIZE(0..30))

### Access

read-only

### Description

Server group with which the client is associated. This information is set through the `grpname` element of the `TPINIT` structure at `tpinit(3)` time.

## **tuxTclientUserName**

### Syntax

*DisplayString*(SIZE(0..30))

### Access

read-only

### Description

User name associated with client at `tpinit(3)` time through the `username` element of the `TPINIT` structure.

## tuxTclientWsc

### Syntax

INTEGER { yes(1), no(2) }

### Access

read-only

### Description

If this object is set to *yes(1)*, the indicated client is logged in to the application from a remote workstation.

## tuxTclientWsh

### Syntax

INTEGER { yes(1), no(2) }

### Access

read-only

### Description

Workstation handler. If this object is set to *yes(1)*, the indicated client is a workstation handler process.

## tuxTclientWshClientId

### Syntax

*DisplayString*(SIZE(1..78))

### Access

read-only

### Description

Client identifier for the associated workstation handler (WSH) if this client is a workstation client (*tuxTclientWsc == yes(1)*); otherwise, the *tuxTclientWshClientId* value is returned as a 0-length string.

## tuxTclientRelease

### Syntax

INTEGER

### Access

read-only

### Description

The Tuxedo system major protocol release number for the machine where the client is running. This value can be different from the `tuxTmachineSWrelease` for the same machine. Note that for /WS clients (`tuxTclientWsc == yes(1)`), this value can be different from the major release associated with the application administered machine through which the /WS client accesses the application.

## tuxTclientWsProto

### Syntax

INTEGER

### Access

read-only

### Description

The Tuxedo system /WS protocol version number for a workstation client. This value is changed with each update to the /WS protocol. A value of 0 is returned for this object when it is associated with non-/WS clients (`tuxTclientWsc == no(2)`).

## tuxTclientNumConv

### Syntax

INTEGER

### Access

read-only

## Description

Number of conversations initiated by this client through `tpconnect(3)`.

## **tuxTclientNumDeque**

### Syntax

INTEGER

### Access

read-only

## Description

Number of dequeue operations initiated by this client through `tpdequeue(3)`.

## **tuxTclientNumEnque**

### Syntax

INTEGER

### Access

read-only

## Description

Number of enqueue operations initiated by this client through `tpenqueue(3)`.

## **tuxTclientNumPost**

### Syntax

INTEGER

### Access

read-only

## Description

Number of postings initiated by this client through `tppost(3)`.

## **tuxTclientNumReq**

### Syntax

INTEGER

### Access

read-only

### Description

Number of requests made by this client through `tpcall(3)` or `tpacall(3)`.

## **tuxTclientNumSubscribe**

### Syntax

INTEGER

### Access

read-only

### Description

Number of subscriptions made by this client through `tpsubscribe(3)`.

## **tuxTclientNumTran**

### Syntax

INTEGER

### Access

read-only

### Description

Number of transactions begun by this client.

## **tuxTclientNumTranAbt**

### Syntax

INTEGER

### Access

read-only

### Description

Number of transactions aborted by this client.

## **tuxTclientNumTranCmt**

### Syntax

INTEGER

### Access

read-only

### Description

Number of transactions committed by this client.

## **tuxTclientCmtRet**

### Syntax

INTEGER { complete(1) | logged(2) }

### Access

read-only

### Description

Setting of the TP\_COMMIT\_CONTROL characteristic for this client. See the description of the Tuxedo system ATMI function `tpsamt(3)` for details on this characteristic.

## tuxTclientCurConv

### Syntax

INTEGER

### Access

read-only

### Description

Number of conversations initiated by this client through `tpconnect(3)` that are still active.

## tuxTclientCurReq

### Syntax

INTEGER

### Access

read-only

### Description

Number of requests initiated by this client through `tpcall(3)` or `tpacall(3)` that are still active.

## tuxTclientCurTime

### Syntax

INTEGER

### Access

read-only

### Description

Current time, in seconds, since 00:00:00 UTC, January 1, 1970, as returned by the `time(2)` system call on the local host. This object can be used to compute elapsed time from the `tuxTclientTimeStart` object value.

## tuxTclientLastGrp

### Syntax

INTEGER

### Access

read-only

### Description

Server group number of the last service request made or conversation initiated from this client.

## tuxTclientNaddr

### Syntax

*DisplayString*(SIZE(1..256)) (up to 78 bytes for Oracle Tuxedo 8.0 or earlier)

### Access

read-only

### Description

For workstation clients, this object indicates the network address of the client. Network addresses with unprintable characters are converted to the “0x...” network address format as described in the `tuxTmachineNaddr` object. Non-workstation clients have a 0-length string associated with them for the `tuxTclientNaddr` value.

**Note:** The ability of the system to provide this information is determined by the transport provider in use. In some cases, workstation clients cannot have addresses associated with them if the provider does not make this information available.

## tuxTclientNotify

### Syntax

INTEGER { dipin(1) | signal(2) | ignore(3) }

### Access

read-only

## Description

Setting of the notification characteristic for this client. See the `tuxTdomain` group description of this object for more details.

## tuxTclientNumUnSol

### Syntax

INTEGER

### Access

read-only

## Description

Number of unsolicited messages queued for this client that are awaiting processing.

## tuxTclientRpid

### Syntax

INTEGER

### Access

read-only

## Description

UNIX system message queue identifier for the client's reply queue.

**Note:** This object is a UNIX-system-specific object value that might not be returned if the platform on which the application is being run is not UNIX-based.

## tuxTclientTimeLeft

### Syntax

INTEGER

### Access

read-only

## Description

Time left, in seconds, for this client to receive the reply for which it is currently waiting before it times out. This timeout can be a transactional timeout or a blocking timeout.

## **tuxTclientTimeStart**

### Syntax

INTEGER

### Access

read-only

## Description

Time, in seconds, since 00:00:00 UTC, January 1, 1970, as returned by the `time(2)` system call on local host, since the client joined the application.

## **tuxTclientTranLev**

### Syntax

INTEGER

### Access

read-only

## Description

Current transaction level for this client. 0 indicates that the client is not currently involved in a transaction.

## **tuxTclientId**

### Syntax

*DisplayString(SIZE(1..78))*

### Access

read-only

## Description

Client Identifier.

## tuxTclientContextID

### Syntax

INTEGER (-2..29999)

### Access

read-only

### Status

mandatory

## Description

Identifier for this particular application association.

## tuxTconnTable

The `tuxTconnTable` group contains objects that represent run-time characteristics of active conversations within an application. Objects in this table are only accessible through a Tuxedo SNMP agent installed on the local machine. All objects in this MIB group are local objects, that is, values for these objects correspond to the local host only where the Tuxedo agent is running. Thus, the user needs to run an instance of the Tuxedo agent on every node for which these values are of interest. The index into this table is `tuxTconnSerNo`.

Object Name	Object ID
<code>tuxTconnSerNo</code>	.1.3.6.1.4.1.140.300.18.1.1.1
<code>tuxTconnState</code>	.1.3.6.1.4.1.140.300.18.1.1.2
<code>tuxTconnSvcName</code>	.1.3.6.1.4.1.140.300.18.1.1.3
<code>tuxTconnClientId</code>	.1.3.6.1.4.1.140.300.18.1.1.4
<code>tuxTconnOgrpNo</code>	.1.3.6.1.4.1.140.300.18.1.1.5
<code>tuxTconnOlmid</code>	.1.3.6.1.4.1.140.300.18.1.1.6

Object Name	Object ID
tuxTconnOpid	.1.3.6.1.4.1.140.300.18.1.1.7
tuxTconnOsndcnt	.1.3.6.1.4.1.140.300.18.1.1.8
tuxTconnOsrvId	.1.3.6.1.4.1.140.300.18.1.1.9
tuxTconnSgrpNo	.1.3.6.1.4.1.140.300.18.1.1.10
tuxTconnSlmid	.1.3.6.1.4.1.140.300.18.1.1.11
tuxTconnSpid	.1.3.6.1.4.1.140.300.18.1.1.12
tuxTconnSsndcnt	.1.3.6.1.4.1.140.300.18.1.1.13
tuxTconnSsrvId	.1.3.6.1.4.1.140.300.18.1.1.14

## tuxTconnSerNo

### Syntax

INTEGER

### Access

read-only

### Description

A running number as an index for tuxTconnTable.

## tuxTconnState

### Syntax

INTEGER { active(1) }

### Access

read-only

### Description

The values for GET and SET operations are as follows:

**GET:**

A **GET** operation retrieves run-time information for the selected `tuxTconnTable` instance(s). The following state indicates the meaning of a `tuxTconnState` returned in response to a **GET** request. States not listed are not returned.

`active(1)`

The `active(1)` state returned reflects one or both sides of an active conversation within the application.

**SET:**

**SET** operations are not permitted for this group.

## **tuxTconnSvcName**

### Syntax

*DisplayString(SIZE(1..127))*

### Access

read-only

### Description

Service name of the conversational service invoked by the originator and processed by the subordinate.

## **tuxTconnClientId**

### Syntax

*DisplayString(SIZE(1..78))*

### Access

read-only

### Description

Client identifier. The data in this field should not be interpreted directly by the end user except for equality comparison.

## **tuxTconnOgrpNo**

### Syntax

INTEGER (1..30001)

### Access

read-only

### Description

Server group number for the originator of the conversation. If the originator is a client, then 30,000 is returned as the value for this object.

## **tuxTconnOImid**

### Syntax

*DisplayString*(SIZE(1..30))

### Access

read-only

### Description

Logical machine identifier that indicates where the originator is running, or (in the case of /WS clients) is accessing the application .

## **tuxTconnOpid**

### Syntax

INTEGER

### Access

read-only

### Description

Process identifier for the originator of the conversation.

## **tuxTconnOsndcnt**

### Syntax

INTEGER

### Access

read-only

### Description

Number of `tpsend(3)` calls made by the originator.

## **tuxTconnOsrvid**

### Syntax

INTEGER (1..30001)

### Access

read-only

### Description

Server identifier for the originator of the conversation.

## **tuxTconnSgrpNo**

### Syntax

INTEGER (1..30001)

### Access

read-only

### Description

Server group number for the subordinate of the conversation. If the originator is a client, then 30,000 is returned as the value for this object.

## **tuxTconnSlmid**

### Syntax

*DisplayString*(SIZE(1..30))

### Access

read-only

### Description

Logical machine identifier that indicates where the subordinate is running or, (in the case of /WS clients) is accessing the application.

## **tuxTconnSpid**

### Syntax

INTEGER

### Access

read-only

### Description

Process identifier for the subordinate in the conversation.

## **tuxTconnSsndcnt**

### Syntax

INTEGER

### Access

read-only

### Description

Number of `tpsend(3)` calls made by the subordinate.

## tuxTconnSsrvid

### Syntax

INTEGER (1..30001)

### Access

read-only

### Description

Server identifier for the subordinate in the conversation.

## tuxTdevice

The `tuxTdevice` group contains the following object and group (table).

Object Name	Object ID
tuxTwhichCfgDev	.1.3.6.1.4.1.140.300.19.2
tuxTdeviceTbl	.1.3.6.1.4.1.140.300.19.1

## tuxTwhichCfgDev

### Syntax

*DisplayString*(SIZE(2..256)) (up to 64 bytes for Oracle Tuxedo 8.0 or earlier)

### Access

read-write

### Description

The value of this object determines the device for which `tuxTdeviceTbl` returns configuration and run-time information.

The default value of this object is the `TUXCONFIG` file for the current domain.

## tuxTdeviceTbl

The `tuxTdeviceTbl` group contains objects that represent configuration and run-time objects of raw disk slices or UNIX system files being used to store Tuxedo system device lists. This group allows for the creation and deletion of device list entries within a raw disk slice or UNIX system file. Objects in this table are only accessible through a Tuxedo SNMP agent installed on the local machine. To create a new row in this table, the user needs to send a SET request, with at least a value for `tuxTdevSize`. The index into this table is `tuxTdevCfgDev` and `tuxTdevIndex`.

Object Name	Object ID
<code>tuxTdevLmid</code>	.1.3.6.1.4.1.140.300.19.1.1.1
<code>tuxTdevCfgDev</code>	.1.3.6.1.4.1.140.300.19.1.1.2
<code>tuxTdeviceName</code>	.1.3.6.1.4.1.140.300.19.1.1.3
<code>tuxTdevOffset</code>	.1.3.6.1.4.1.140.300.19.1.1.4
<code>tuxTdevSize</code>	.1.3.6.1.4.1.140.300.19.1.1.5
<code>tuxTdevIndex</code>	.1.3.6.1.4.1.140.300.19.1.1.6
<code>tuxTdevState</code>	.1.3.6.1.4.1.140.300.19.1.1.7

### tuxTdevLmid

#### Syntax

*DisplayString* (SIZE(1..30))

#### Access

read-write

#### Description

Logical machine identifier where the device is located. Note that this object can be used as a key field in both unbooted and booted applications as long as they are already configured (that is, at least one `tuxTmachineTable` instance exists). It is required as a key field on SET operations when they are accessing a booted application. If specified when accessing the `tuxTdeviceTbl` table in an unconfigured application, this object is ignored.

**Note:** This object can be set only during row creation.

## tuxTdevCfgDev

### Syntax

*DisplayString*(SIZE(2..64))

### Access

read-write

### Description

Absolute pathname of the file or device where the Tuxedo system filesystem is stored or is to be stored.

**Note:** This object can be set only during row creation.

## tuxTdeviceName

### Syntax

*DisplayString*(SIZE(2..64))

### Access

read-write

### Description

Absolute pathname of the device list entry.

**Note:** This object can be set only during row creation.

## tuxTdevOffset

### Syntax

INTEGER

### Access

read-write

## Description

The offset, in blocks, at which space on this `tuxTdevice` begins for use within the Tuxedo system VTOC specified by `tuxTdevCfgDev`.

**Note:** This object can be set only during row creation.

## **tuxTdevSize**

### Syntax

INTEGER

### Access

read-write

## Description

The size in pages of the disk area to be used for the device list entry.

**Note:** This object can be set only in conjunction with row creation.

**Note:** This object can be set only during row creation.

## **tuxTdevIndex**

### Syntax

INTEGER

### Access

read-only

## Description

Device index for `tuxTdevice` within the device list addressed by `tuxTdevCfgDev`. The `tuxTdevIndex` value is used for identification purposes only in getting and setting object values that relate to particular devices within a Tuxedo system filesystem.

## tuxTdevState

### Syntax

```
INTEGER { valid(1) | invalid(2) | re-init(3) }
```

### Access

read-write

### Description

The values for GET and SET operations are as follows:

GET: {valid(1)}

A GET operation retrieves run-time information for the selected `tuxTdeviceTbl` instance(s). The following state indicates the meaning of a `tuxTdevState` returned in response to a GET request. States not listed are not returned.

valid(1)

The Tuxedo system filesystem indicated by `tuxTdevCfgDev` exists and contains a valid device list. `tuxTdevice` is a valid device within that filesystem with the device index `tuxTdevIndex`.

SET: {invalid(2) | re-init(3)}

A SET operation updates information for the selected `tuxTdeviceTbl` instance or adds the indicated object. The following states indicate the meaning of a `tuxTdevState` set in a SET request. States not listed cannot be set.

invalid(2)

Delete `tuxTdeviceTbl` instance for application. State change is allowed only when the object is in the `valid(1)` state. Successful return leaves the object in the `invalid(2)` state. Note that `tuxTdevIndex 0` is special and must be deleted last.

re-init(3)

Re-initializes a valid device.

## tuxTdomain

The `tuxTdomain` group contains objects that represent global application characteristics for the domain to which the Tuxedo SNMP Agent is currently connected. The object values serve to identify, customize, size, secure, and tune a Tuxedo application. Many of the object values represented here serve as application defaults for other groups represented in this MIB.

There is exactly one instance of the `tuxTdomain` group for each application.

<b>Object Name</b>	<b>Object ID</b>
tuxTdomainKey	.1.3.6.1.4.1.140.300.3.1
tuxTdomainMaster	.1.3.6.1.4.1.140.300.3.2
tuxTdomainModel	.1.3.6.1.4.1.140.300.3.3
tuxTdomainState	.1.3.6.1.4.1.140.300.3.4
tuxTdomainID	.1.3.6.1.4.1.140.300.3.5
tuxTdomainUID	.1.3.6.1.4.1.140.300.3.7
tuxTdomainGID	.1.3.6.1.4.1.140.300.3.8
tuxTdomainPerm	.1.3.6.1.4.1.140.300.3.9
tuxTdomainMask	.1.3.6.1.4.1.140.300.3.10
tuxTdomainMaxAccessers	.1.3.6.1.4.1.140.300.3.11
tuxTdomainMaxConv	.1.3.6.1.4.1.140.300.3.12
tuxTdomainMaxGTT	.1.3.6.1.4.1.140.300.3.13
tuxTdomainMaxBufsType	.1.3.6.1.4.1.140.300.3.14
tuxTdomainMaxBufType	.1.3.6.1.4.1.140.300.3.15
tuxTdomainMaxDRT	.1.3.6.1.4.1.140.300.3.16
tuxTdomainMaxGroups	.1.3.6.1.4.1.140.300.3.17
tuxTdomainMaxMachines	.1.3.6.1.4.1.140.300.3.18
tuxTdomainMaxQueues	.1.3.6.1.4.1.140.300.3.19
tuxTdomainMaxRFT	.1.3.6.1.4.1.140.300.3.20
tuxTdomainMaxRTData	.1.3.6.1.4.1.140.300.3.21
tuxTdomainMaxServers	.1.3.6.1.4.1.140.300.3.22
tuxTdomainMaxServices	.1.3.6.1.4.1.140.300.3.23

<b>Object Name</b>	<b>Object ID</b>
tuxTdomainMaxACLgroups	.1.3.6.1.4.1.140.300.3.24
tuxTdomainCMTRET	.1.3.6.1.4.1.140.300.3.25
tuxTdomainLoadBalance	.1.3.6.1.4.1.140.300.3.26
tuxTdomainNotify	.1.3.6.1.4.1.140.300.3.27
tuxTdomainSystemAccess	.1.3.6.1.4.1.140.300.3.28
tuxTdomainOptions	.1.3.6.1.4.1.140.300.3.29
tuxTdomainSignal	.1.3.6.1.4.1.140.300.3.30
tuxTdomainSecurity	.1.3.6.1.4.1.140.300.3.31
tuxTdomainAuthsvc	.1.3.6.1.4.1.140.300.3.33
tuxTdomainScanUnit	.1.3.6.1.4.1.140.300.3.34
tuxTdomainBBLQuery	.1.3.6.1.4.1.140.300.3.35
tuxTdomainBlockTime	.1.3.6.1.4.1.140.300.3.36
tuxTdomainDBBLWait	.1.3.6.1.4.1.140.300.3.37
tuxTdomainSanityScan	.1.3.6.1.4.1.140.300.3.38
tuxTdomainCurDRT	.1.3.6.1.4.1.140.300.3.39
tuxTdomainCurGroups	.1.3.6.1.4.1.140.300.3.40
tuxTdomainCurMachines	.1.3.6.1.4.1.140.300.3.41
tuxTdomainCurQueues	.1.3.6.1.4.1.140.300.3.42
tuxTdomainCurRFT	.1.3.6.1.4.1.140.300.3.43
tuxTdomainCurRTdata	.1.3.6.1.4.1.140.300.3.44
tuxTdomainCurServers	.1.3.6.1.4.1.140.300.3.45
tuxTdomainCurServices	.1.3.6.1.4.1.140.300.3.46
tuxTdomainCursType	.1.3.6.1.4.1.140.300.3.47

<b>Object Name</b>	<b>Object ID</b>
tuxTdomainCurType	.1.3.6.1.4.1.140.300.3.48
tuxTdomainHwDRT	.1.3.6.1.4.1.140.300.3.49
tuxTdomainHwGroups	.1.3.6.1.4.1.140.300.3.50
tuxTdomainHwMachines	.1.3.6.1.4.1.140.300.3.51
tuxTdomainHwQueues	.1.3.6.1.4.1.140.300.3.52
tuxTdomainHwRFT	.1.3.6.1.4.1.140.300.3.53
tuxTdomainHwRTdata	.1.3.6.1.4.1.140.300.3.54
tuxTdomainHwServers	.1.3.6.1.4.1.140.300.3.55
tuxTdomainHwServices	.1.3.6.1.4.1.140.300.3.56
tuxTdomainMaxNetGroups	.1.3.6.1.4.1.140.300.3.58
tuxMaxObjects (Tuxedo 8.0 or later)	.1.3.6.1.4.1.140.300.3.63
tuxMaxInterfaces (Tuxedo 8.0 or later)	.1.3.6.1.4.1.140.300.3.68
tuxTdomainSignatureAhead	.1.3.6.1.4.1.140.300.3.70
tuxCurInterfaces (Tuxedo 8.0 or later)	.1.3.6.1.4.1.140.300.3.73
tuxHwInterfaces (Tuxedo 8.0 or later)	.1.3.6.1.4.1.140.300.3.78
tuxTdomainSignatureBehind	.1.3.6.1.4.1.140.300.3.80
tuxTdomainEncryptionRequired	.1.3.6.1.4.1.140.300.3.90
tuxTdomainSignatureRequired	.1.3.6.1.4.1.140.300.3.100

## **tuxTdomainKey**

### **Syntax**

INTEGER (32769..262143)

### **Access**

read-write

## Description

Numeric key for the well-known address in a Tuxedo system bulletin board. In a single processor environment, this key “names” the bulletin board. In a multiple processor or LAN environment, this key names the message queue of the DBBL. In addition, this key is used as a basis for deriving the names of resources other than the well-known address, such as the names for bulletin boards throughout the application.

## tuxTdomainMaster

### Syntax

*DisplayString* (SIZE (1..30))

### Access

read-write

### Description

*DisplayString* is in format: *LMID1* [ , *LMID2* ]

*LMID1*

Is the master logical machine identifier and is in the range from one to thirty characters.

*LMID2*

Is the backup logical machine identifier and is in the range from one to thirty characters.

The master identifier (*LMID1*) must correspond to the local machine for inactive applications. *single-machine*(1) mode applications (see *tuxTdomainModel* below) can set only the master logical machine identifier. Modifications to the *tuxTdomainMaster* value in an active *multi-machine*(2) application (see *tuxTdomainModel* below) have the following semantics.

Assuming current active master LMID A, current backup master LMID B, and secondary LMIDs C, D, ..., the following scenarios define the semantics of permitted changes to the *tuxTdomainMaster* object in a running *multi-machine*(2) mode application.

A,B -> B,A — Master migration from A to B. A,B -> A,C — Change backup master LMID designation to C.

Note that master migration can be either orderly or partitioned. Orderly migration takes place when the master machine is active and reachable. Otherwise, partitioned migration takes place. All newly established or re-established network connections verify that the two sites connecting share a common view of where the master machine is located. Otherwise, the connection is refused and an appropriate log message is generated.

The master and backup machines in an active application must always have a Tuxedo system release number greater than or equal to all other machines active in the application. The master and backup machines must be of the same release. Modifications to the `tuxTdomainMaster` object must preserve this relationship.

## tuxTdomainModel

### Syntax

```
INTEGER { single-machine(1) | multi-machine(2) }
```

### Access

read-write

### Description

The configuration type.

`single-machine(1)`

Specifies a single machine configuration; only one `tuxTmachineTable` object can be specified.

`multi-machine(2)`

Specifies a multi-machine or network configuration; `multi-machine(2)` must be specified if a networked application is being defined.

## tuxTdomainState

### Syntax

```
INTEGER { active(1) | inactive(2) | forcible-inactive(3) }
```

### Access

read-write

### Description

The values for GET and SET operations are as follows:

GET: { `active(1)` | `inactive(2)` }

A GET operation retrieves configuration and run-time information for the `tuxTdomain` group. The following states indicate the meaning of a `tuxTdomainState` returned in response to a GET request. States not listed are not returned.

`active(1)`

`tuxTdomain` group is defined and the master machine is active.

`inactive(2)`

`tuxTdomain` group is defined and application is inactive.

SET: `active(1) | inactive(2) | forcible-inactive(3)`

A SET operation updates configuration and run-time information for the `tuxTdomain` group. The following states indicate the meaning of a `tuxTdomainState` set in a SET request. States not listed cannot be set.

`active(1)`

Activate administrative processes (DBBL, BBL, and so on) on the master machine. A state change is allowed only when the object is in the `inactive(2)` state. Successful return leaves the object in the `active(1)` state.

`inactive(2)`

Deactivate administrative processes (DBBL, BBL, and so on) on the master machine. A state change is allowed only when the object is in the `active(1)` state. Successful return leaves the object in the `inactive(2)` state. To do a complete shutdown of the application, you must first make all groups inactive. (See [tuxTgroupState](#).) This state transition fails if any application servers or clients are still attached to the domain. To ignore any running clients or application servers, set to `forcible-inactive(3)` as explained below.

`forcible-inactive(3)`

Forcibly deactivate administrative processes (DBBL, BBL, and so on) on the master machine. Attached clients are ignored for the purpose of determining if shutdown should be allowed. State change is allowed only when the object is in the `active(1)` state. Successful return leaves the object in the `inactive(2)` state. You need to restart any clients before they can be used to process services after this state transition.

## tuxTdomainID

### Syntax

*DisplayString* (SIZE (0..30))

### Access

read-write

### Description

Domain identification string.

## tuxTdomainUID

### Syntax

INTEGER

### Access

read-write

### Description

Default value for newly configured objects in the `tuxTmachineTable` group.

**Note:** Changes to this object do not affect active or already configured `tuxTmachineTable` instances.

## tuxTdomainGID

### Syntax

INTEGER

### Access

read-write

### Description

Default value for newly configured objects in the `tuxTmachineTable` group.

**Note:** Changes to this object do not affect active or already configured `tuxTmachineTable` instances.

## tuxTdomainPerm

### Syntax

*DisplayString* (SIZE(1..9))

### Access

read-write

## Description

Default value for newly configured objects in the `tuxTmachineTable` group.

**Note:** Changes to this object do not affect active or already configured `tuxTmachineTable` instances.

## tuxTdomainMask

### Syntax

*DisplayString*(SIZE(1..9))

### Access

read-write

### Description

Attribute access mask. User type/access mode combinations specified by `tuxTdomainMask` are no longer allowed for all group/object combinations defined in `TM_MIB(5)`. For example, a setting of 0003 disallows all updates to users other than the administrator or the operator. The value of this object should be provided as an octal number — 0 through 0777.

## tuxTdomainMaxAccessers

### Syntax

INTEGER (1..32767)

### Access

read-write

### Description

Default value for newly configured objects in the `tuxTmachineTable` group.

**Note:** Changes to this object do not affect active or already configured `tuxTmachineTable` instances.

## **tuxTdomainMaxConv**

### Syntax

INTEGER (0..32767)

### Access

read-write

### Description

Default value for newly configured objects in the `tuxTmachineTable` group.

**Note:** Changes to this object do not affect active or already configured `tuxTmachineTable` instances.

## **tuxTdomainMaxGTT**

### Syntax

INTEGER (0..32767)

### Access

read-write

### Description

Default value for newly configured objects in the `tuxTmachineTable` group.

**Note:** Changes to this object do not affect active or already configured `tuxTmachineTable` instances.

## **tuxTdomainMaxBufsType**

### Syntax

INTEGER (1..32767)

### Access

read-write

## Description

Maximum number of buffer subtypes that can be accommodated in the bulletin board buffer subtype table.

## **tuxTdomainMaxBufType**

### Syntax

INTEGER (1..32767)

### Access

read-write

## Description

Maximum number of buffer types that can be accommodated in the bulletin board buffer type table.

## **tuxTdomainMaxDRT**

### Syntax

INTEGER (0..32767)

### Access

read-write

## Description

Maximum number of routing table entries that can be accommodated in the bulletin board routing table. One entry per `tuxTroutingTable` group object is required. Additional entries should be allocated to allow for run-time growth.

## **tuxTdomainMaxGroups**

### Syntax

INTEGER (100..32767)

### Access

read-write

## Description

Maximum number of server groups that can be accommodated in the bulletin board server group table.

## **tuxTdomainMaxMachines**

### Syntax

INTEGER (256..8190)

### Access

read-write

## Description

Maximum number of machines that can be accommodated in the bulletin board machine table.

## **tuxTdomainMaxQueues**

### Syntax

INTEGER (1..8191)

### Access

read-write

## Description

Maximum number of queues to be accommodated in the bulletin board queue table.

## **tuxTdomainMaxRFT**

### Syntax

INTEGER (0..32767)

### Access

read-write

## Description

Maximum number of routing criteria range table entries to be accommodated in the bulletin board range criteria table. One entry per individual range within a `tuxTroutingRanges` specification is required plus one additional entry per `tuxTroutingTable` group object. Additional entries should be allocated to allow for run-time growth.

## **tuxTdomainMaxRTData**

### Syntax

INTEGER (0..32760)

### Access

read-write

## Description

Maximum string pool space to be accommodated in the bulletin board string pool table. Strings and carrays specified within `tuxTroutingRanges` values are stored in the string pool. Additional space should be allocated to allow for run-time growth.

## **tuxTdomainMaxServers**

### Syntax

INTEGER (1..8191)

### Access

read-write

## Description

Maximum number of servers to be accommodated in the bulletin board server table. Allowances should be made in setting this object for system supplied administrative servers. Administration of each Tuxedo system site adds approximately one server. Additionally, if TMSs are specified for any server groups (see `tuxTgroupTMSname`), they are booted along with their server group and should be accounted for in setting `tuxTdomainMaxServers`.

## **tuxTdomainMaxServices**

### Syntax

INTEGER (1..32767)

### Access

read-write

### Description

Maximum number of services to be accommodated in the bulletin board service table. Allowances should be made in setting this object for system supplied servers that offer services for administrative purposes. Administration of each Tuxedo system site adds approximately five services. Other administrative components such as /WS, /Q, and /DM can also add administrative services that should be accounted for.

## **tuxTdomainMaxACLgroups**

### Syntax

INTEGER (1..16384)

### Access

read-write

### Description

Maximum number of group identifiers that can be used for checking ACL permissions. The maximum group identifier that can be defined is `tuxTdomainMaxACLgroups - 1`.

## **tuxTdomainCMTRET**

### Syntax

INTEGER { complete(1) | logged(2) }

### Access

read-write

## Description

Initial setting of the TP\_COMMIT\_CONTROL characteristic for all client and server processes in a Tuxedo application. `logged(2)` initializes the TP\_COMMIT\_CONTROL characteristic to TP\_CMT\_LOGGED; otherwise, it is initialized to TP\_CMT\_COMPLETE. See the description of the Tuxedo system ATMI function `tpscmt(3)` for details on the setting of this characteristic.

**Note:** Run-time modifications to this object do not affect active clients and servers.

## tuxTdomainLoadBalance

### Syntax

```
INTEGER { yes(1) | no(2) }
```

### Access

read-write

### Description

`yes(1)`  
Load balancing is on.

`no(2)`  
Load balancing is off.

**Note:** Run-time modifications to this object do not affect active clients and servers.

## tuxTdomainNotify

### Syntax

```
INTEGER { dipin(1) | signal(2) | ignore(3) }
```

### Access

read-write

### Description

Default notification detection method used by the system for unsolicited messages sent to client processes. This default value can be overridden on a per-client basis using the appropriate `tpinit(3)` flag value. Note that once unsolicited messages are detected, they are made available

to the application through the application defined unsolicited message handling routine identified through the `tpsetunsol(3)` function.

`dipin(1)`

The value `dipin(1)` specifies that dip-in-based notification detection should be used, which means that the system only detects notification messages on behalf of a client process while within ATMI calls. The point of detection within any particular ATMI call is not defined by the system, and dip-in detection does not interrupt blocking system calls. `dipin(1)` is the default notification detection method.

`signal(2)`

The value `signal(2)` specifies that signal-based notification detection should be used, which means that the system sends a signal to the target client process after the notification message has been made available. The system installs a signal catching routine on behalf of clients that select this method of notification.

`ignore(3)`

The value `ignore(3)` specifies that by default, notification messages are to be ignored by application clients, which would be appropriate in applications where only clients that request notification at `tpinit(3)` time should receive unsolicited messages.

**Note:** Run-time modifications to this object do not affect active clients. All signaling of client processes is done by administrative system processes and not by application processes. Therefore, only clients running with the same UNIX system user identifier can be notified by use of the `signal(2)` method.

## tuxTdomainSystemAccess

### Syntax

```
INTEGER { fastpath(1) | protected(2) | fastpath-no-override(3) |  
protected-no-override(4) }
```

### Access

read-write

### Description

Default mode used by Tuxedo system libraries within application processes to gain access to Tuxedo system's internal tables.

`fastpath(1)`

Specifies that Tuxedo system's internal tables are accessible by Tuxedo system libraries via unprotected shared memory for fast access.

protected(2)

Specifies that Tuxedo system's internal tables are accessible by Tuxedo libraries through protected shared memory for safety against corruption by application code.

fastpath-no-override(3) **or** protected-no-override(4)

These values can be specified to indicate that the mode selected cannot be overridden by an application process that uses flags available for use with `tpinit(3)`.

**Note:** Updates to the `tuxTdomainSystemAccess` value in a running application affect only newly started clients and newly configured `tuxTsrvrTbl` objects.

## tuxTdomainOptions

### Syntax

```
INTEGER { lan(1) | migrate(2) | accstats(3) | lan-migrate(4) |
lan-accstats(5) | migrate-accstats(6) | lan-migrate-accstats(7) | none(8) }
```

### Access

read-write

### Description

Comma separated list of application options in effect. Valid options are defined as follows:

lan(1)

Networked application.

migrate(2)

Allow server group migration.

accstats(3)

Exact statistics (`single-machine(1)` mode only).

**Note:** Only the `accstats(3)` can be set or reset in an active application.

## tuxTdomainSignal

### Syntax

```
INTEGER { sigusr1(1) | sigusr2(2) }
```

### Access

read-write

## Description

Signal to be used for signal-based notification (see `tuxTdomainNotify` above).

## tuxTdomainSecurity

### Syntax

INTEGER *DisplayString*

### Access

read-write

### Description

Type of application security. The format is:

*security\_mode*[/*app\_password*]

where *security\_mode* can have the following values:

NONE

APP\_PW

USER\_AUTH

ACL

MANDATORY\_ACL

NONE

A string value of NONE for this object indicates that security is/will be turned off.

APP\_PW

The value APP\_PW/*app\_password* indicates that application password security is enforced. Clients must provide the application password during initialization.

USER\_AUTH

The value USER\_AUTH is similar to APP\_PW, but indicates also that per-user authentication is done during client initialization.

ACL

The value ACL is similar to USER\_AUTH, but also indicates that access control checks are done on service names, queue names, and event names. If an associated ACL is not found for a name, it is assumed that permission is granted

MANDATORY\_ACL

The value MANDATORY\_ACL is similar to ACL, but permission is denied if an associated ACL is not found for the name.

`app_password`

This value is needed whenever `security_mode` is being set to anything but NONE. To change the value of `app_password`, SET this object to:

`current_security_mode/new_password`

On a GET operation, this object only returns the security mode; the password is not returned.

## tuxTdomainAuthsvc

### Syntax

`DisplayString (SIZE (1..127))`

### Access

read-write

### Description

Application authentication service invoked by the system for each client that joins the system. The `tuxTdomainAuthsvc` value is ignored if the `tuxTdomainSecurity` object is set to NONE or to APP-PW.

## tuxTdomainScanUnit

### Syntax

`INTEGER (0..60)`

### Access

read-write

### Description

Interval of time (in seconds) between periodic scans by the system. Periodic scans are used to detect old transactions and timed-out blocking calls within service requests. The `tuxTdomainBBLQuery`, `tuxTdomainBlockTime`, `tuxTdomainDBBLWait`, and `tuxTdomainSanityScan` objects are multipliers of this value. Passing a value of 0 for this object on a SET operation causes the object to be reset to its default value.

## **tuxTdomainBBLQuery**

### **Syntax**

INTEGER (0..32767)

### **Access**

read-write

### **Description**

Multiplier of the `tuxTdomainScanUnit` object that indicates time between DBBL status checks on registered BBLs. The DBBL checks to ensure that all BBLs have reported within the `tuxTdomainBBLQuery` cycle. If a BBL has not been heard from, the DBBL sends a message to that BBL asking for status. If no reply is received, the BBL is partitioned. Passing a value of 0 for this object on a `SET` operation causes the object to be reset to its default value. The `tuxTdomainBBLQuery` value should be set to at least twice the value set for `tuxTdomainSanityScan`.

## **tuxTdomainBlockTime**

### **Syntax**

INTEGER (0..32767)

### **Access**

read-write

### **Description**

Multiplier of the `tuxTdomainScanUnit` object that indicates the minimum amount of time a blocking ATMI call blocks before timing out. Passing a value of 0 for this object on a `SET` operation causes the object to be reset to its default value.

## **tuxTdomainDBBLWait**

### **Syntax**

INTEGER (0..32767)

## Access

read-write

## Description

Multiplier of the `tuxTdomainScanUnit` object that indicates the maximum amount of time a DBBL should wait for replies from its BBLs before timing out. Passing a value of 0 for this object on a `SET` operation causes the object to be reset to its default value.

## **tuxTdomainSanityScan**

### Syntax

INTEGER (0..32767)

### Access

read-write

### Description

Multiplier of the `tuxTdomainScanUnit` object that indicates the time interval between basic sanity checks of the system. Sanity checking includes client/server viability checks done by each BBL for clients/servers running on the local machine as well as BBL status check-ins (`multi-machine(2)` mode only). Passing a value of 0 for this object on a `SET` operation causes the object to be reset to its default value.

## **tuxTdomainCurDRT**

### Syntax

INTEGER (0..32767)

### Access

read-only

### Description

Current number of bulletin board routing table entries in use.

## **tuxTdomainCurGroups**

### Syntax

INTEGER (0..32767)

### Access

read-only

### Description

Current number of bulletin board server group table entries that are in use.

## **tuxTdomainCurMachines**

### Syntax

INTEGER (0..32767)

### Access

read-only

### Description

Current number of configured machines.

## **tuxTdomainCurQueues**

### Syntax

INTEGER (0..32767)

### Access

read-only

### Description

Current number of bulletin board queue table entries that are in use.

## **tuxTdomainCurRFT**

### Syntax

INTEGER (0..32767)

### Access

read-only

### Description

Current number of bulletin board routing criteria range table entries that are in use.

## **tuxTdomainCurRTdata**

### Syntax

INTEGER (0..32767)

### Access

read-only

### Description

Current size of routing table string pool.

## **tuxTdomainCurServers**

### Syntax

INTEGER (0..32767)

### Access

read-only

### Description

Current number of bulletin board server table entries that are in use.

## **tuxTdomainCurServices**

### Syntax

INTEGER (0..32767)

### Access

read-only

### Description

Current number of bulletin board service table entries that are in use.

## **tuxTdomainCursType**

### Syntax

INTEGER (0..32767)

### Access

read-only

### Description

Current number of bulletin board subtype table entries that are in use.

## **tuxTdomainCurType**

### Syntax

INTEGER (0..32767)

### Access

read-only

### Description

Current number of bulletin board type table entries that are in use.

## **tuxTdomainHwDRT**

### Syntax

INTEGER (0..32767)

### Access

read-only

### Description

High water number of bulletin board routing table entries that are in use.

## **tuxTdomainHwGroups**

### Syntax

INTEGER (0..32767)

### Access

read-only

### Description

High water number of bulletin board server group table entries that are in use.

## **tuxTdomainHwMachines**

### Syntax

INTEGER (0..32767)

### Access

read-only

### Description

High water number of configured machines.

## **tuxTdomainHwQueues**

### Syntax

INTEGER (0..32767)

### Access

read-only

### Description

High water number of bulletin board queue table entries that are in use.

## **tuxTdomainHwRFT**

### Syntax

INTEGER (0..32767)

### Access

read-only

### Description

High water number of bulletin board routing criteria range table entries that are in use.

## **tuxTdomainHwRTdata**

### Syntax

INTEGER (0..32767)

### Access

read-only

### Description

High water size of routing table string pool.

## **tuxTdomainHwServers**

### Syntax

INTEGER (0..32767)

### Access

read-only

### Description

High water number of bulletin board server table entries that are in use.

## **tuxTdomainHwServices**

### Syntax

INTEGER (0..32767)

### Access

read-only

### Description

High water number of bulletin board service table entries that are in use.

## **tuxTdomainMaxNetGroups**

### Syntax

INTEGER (1..8191)

### Access

read-write

### Description

The maximum number of groups that can be configured.

## **tuxMaxObjects (Tuxedo 8.0 or later)**

### Syntax

INTEGER

### Access

read-write

### Description

The default maximum number of active CORBA objects that can be accommodated in the Active Object Map tables in the Tuxedo bulletin board.

## **tuxMaxInterfaces (Tuxedo 8.0 or later)**

### Syntax

INTEGER (1..32765)

### Access

read-write

### Description

Specifies the maximum number of interfaces that can be accommodated in the interface table of the bulletin board. If not specified, the default is 100.

All instances of an interface occupy and re-use the same slot in the interface table in the bulletin board. For example, if server SVR1 advertises interfaces IF1 and IF2, SVR2 advertises IF2 and IF3, and SVR3 advertises IF3 and IF4, the interface count is 4, not 6, when calculating `tuxMaxInterfaces`.

## **tuxTdomainSignatureAhead**

### Syntax

INTEGER (1..2147483647)

### Access

read-write

## Description

Number of seconds a valid signature's timestamp can be ahead of the local machine's clock.

## **tuxCurInterfaces (Tuxedo 8.0 or later)**

### Syntax

INTEGER

### Access

read-only

### Description

The current number of interface entries used in the bulletin board interface tables.

## **tuxHwInterfaces (Tuxedo 8.0 or later)**

### Syntax

INTEGER

### Access

read-only

### Description

The high water mark for the number of interface entries used in the bulletin board interface tables.

## **tuxTdomainSignatureBehind**

### Syntax

INTEGER (1..2147483647)

### Access

read-write

### Description

Number of seconds a valid signature's timestamp can be behind the local machine's clock.

## **tuxTdomainEncryptionRequired**

### Syntax

INTEGER {yes(1) | no(2)}

### Access

read-write

### Description

If set to “yes,” every application service in this domain requires an encrypted input message buffer.

## **tuxTdomainSignatureRequired**

### Syntax

INTEGER {yes(1) | no(2)}

### Access

read-write

### Description

If set to “yes,” every application service in this domain requires a valid digital signature on its input message buffer.

## **tuxTgroupTable**

The `tuxTgroupTable` group contains objects that represent application characteristics pertaining to a particular server group. The object values represent group identification, location, and DTP information.

The index for this table is `tuxTgroupNo`. To create a new row, it is necessary to issue a `SET` request for a non-existing instance that at least specifies values for `tuxTgroupName` and `tuxTgroupLMID`.

Object Name	Object ID
tuxTgroupName	.1.3.6.1.4.1.140.300.4.1.1.1
tuxTgroupNo	.1.3.6.1.4.1.140.300.4.1.1.2
tuxTgroupLMID	.1.3.6.1.4.1.140.300.4.1.1.3
tuxTgroupState	.1.3.6.1.4.1.140.300.4.1.1.4
tuxTgroupCurLMID	.1.3.6.1.4.1.140.300.4.1.1.5
tuxTgroupCloseInfo	.1.3.6.1.4.1.140.300.4.1.1.6
tuxTgroupOpenInfo	.1.3.6.1.4.1.140.300.4.1.1.7
tuxTgroupTMScount	.1.3.6.1.4.1.140.300.4.1.1.8
tuxTgroupTMSname	.1.3.6.1.4.1.140.300.4.1.1.9
tuxTgroupEncryptionRequired	.1.3.6.1.4.1.140.300.4.1.1.20
tuxTgroupSignatureRequired	.1.3.6.1.4.1.140.300.4.1.1.30

## tuxTgroupName

### Syntax

*DisplayString*(SIZE(1..30))

### Access

read-write

### Description

Logical name of the server group. The group name must be unique within all group names in the tuxTgroupTable group and tuxTgroupLMID values in the tuxTmachineTable group. Server group names cannot contain an asterisk (\*), comma, or colon.

**Note:** This object can be set only during row creation.

## tuxTgroupNo

### Syntax

INTEGER (1..29999)

### Access

read-write

### Description

Group number associated with this server group.

**Note:** This object can be set only during row creation.

## tuxTgroupLMID

### Syntax

*DisplayString* (SIZE (1..61))

### Access

read-write

### Description

*DisplayString* is in the format: *LMID1*[, *LMID2*]

*LMID1*

Is the primary machine logical machine identifier for this server group and is in the range from one to sixty-one characters.

*LMID2*

Is the optional secondary logical machine identifier and is in the range from one to sixty-one characters.

The secondary LMID indicates the machine to which the server group can be migrated (if the MIGRATE option is specified in the `tuxTdomainOptions` object). A single LMID specified on a GET operation matches either the primary or secondary LMID. Note that the location of an active group is available in the `tuxTgroupCurLMID` object. Logical machine identifiers specified with the `tuxTgroupLMID` object must already be configured.

**Note:** Modifications to this object for an active object can only change the backup LMID designation for the group.

## tuxTgroupState

### Syntax

```
INTEGER { active(1) | inactive(2) | migrating(3) | invalid(4) |
re-active(5) | suspend-services(6) | resume-services(7) }
```

### Access

read-write

### Description

The values for GET and SET operations are as follows:

```
GET: { active(1) | inactive(2) | migrating(3) }
```

A GET operation retrieves configuration and run-time information for the selected tuxTgroupTable object(s). The following states indicate the meaning of a tuxTgroupState returned in response to a GET request. States not listed are not returned.

```
active(1)
```

tuxTgroupTable object defined and active (TMS and/or application servers). Server groups with non-0 length values for the tuxTgroupTMSname object are considered active if the TMSs associated with the group are active. Otherwise, a group is considered active if any server in the group is active.

```
inactive(2)
```

tuxTgroupTable object defined and inactive.

```
migrating(3)
```

tuxTgroupTable object defined and currently in a state of migration to the secondary logical machine. The secondary logical machine is the one listed in tuxTgroupLMID that does not match tuxTgroupCurLMID.

```
SET: { active(1) | inactive(2) | migrating(3) | invalid(4) | re-active(5)
| suspend-services(6) | resume-services(7) }
```

A SET operation updates configuration and run-time information for the selected tuxTgroupTable object. The following states indicate the meaning of a tuxTgroupState set in a SET request. States not listed cannot be set.

```
active(1)
```

Activate the tuxTgroupTable object. State change is allowed only when the group is in the inactive(2) or migrating(3) state. If the group is currently in the inactive(2) state and the primary logical machine is active, then TMS and application servers are started on the primary logical machine; otherwise, if the secondary logical machine is active, the TMS and application servers are started on the secondary logical machine. If

neither machine is active, then the request fails. If the group is currently in the `migrating(3)` state, then the active secondary logical machine (identified as the alternate to `tuxTgroupCurLMID` in the `tuxTgroupLMID` list), if it is active, is used to start TMS and application servers. Otherwise, the request fails. Successful return leaves the object in the `active(1)` state.

`inactive(2)`

Deactivate the `tuxTgroupTable` instance. TMS and application servers are deactivated. State change is allowed only when the group is in the `active(1)` or `migrating(3)` state. Successful return leaves the object in the `inactive(2)` state.

`migrating(3)`

Deactivate the `tuxTgroupTable` object on its active primary logical machine (`tuxTgroupCurLMID`) and prepare the group to be migrated to the secondary logical machine. State change is allowed only when the group is in the `active(1)` state. Successful return leaves the object in the `migrating(3)` state.

`invalid(4)`

Delete `tuxTgroupTable` object for application. State change is allowed only when the group is in the `inactive(2)` state. Successful return leaves the object in the `invalid(4)` state.

`re-active(5)`

Identical to a transition to the `active(1)` state except that this state change is also allowed in the `active(1)` state as well as the `inactive(2)` and `migrating(3)` states.

`suspend-services(6)`

Suspend the application services in the group. A `SET` operation to this state is allowed only when the group is in the `active(1)` state. The operation leaves the group in `active(1)` state but with all its application services in a suspended state.

`resume-services(7)`

Unsuspend and resume all application services that are marked suspended in the group. This operation is allowed only when the group is in the `active(1)` state. The operation leaves the group in the `active(1)` state.

## **tuxTgroupCurLMID**

### **Syntax**

*DisplayString* (SIZE(1..30))

### **Access**

read-only

## Description

Current logical machine on which the server group is running. The `tuxTgroupCurLMID` value is not returned for server groups that are not active.

## tuxTgroupCloseInfo

### Syntax

*DisplayString*(SIZE(0..256))

### Access

read-write

## Description

If a non-0 length value other than TMS is specified for the `tuxTgroupTMSname` object, then the `tuxTgroupCloseInfo` value indicates the resource manager-dependent information needed to terminate access to the resource manager. Otherwise, this object value is ignored.

The format for the `tuxTgroupCloseInfo` value is dependent on the requirements of the vendor providing the underlying resource manager. The information required by the vendor must be prefixed with `rm_name:`, which is the published name of the vendor's transaction (XA) interface followed immediately by a colon (:).

A 0-length string value for this object means that the resource manager for this group (if specified) does not require any application-specific information to close access to the resource.

**Note:** Run-time modifications to this object do not affect active servers in the group.

## tuxTgroupOpenInfo

### Syntax

*DisplayString*(SIZE(0..256))

### Access

read-write

## Description

If a non-0 length value other than TMS is specified for the `tuxTgroupTMSname` object, the `tuxTgroupOpenInfo` value indicates the resource manager-dependent information needed to initiate access to the resource manager. Otherwise, this object value is ignored.

The format for the `tuxTgroupOpenInfo` value is dependent on the requirements of the vendor that provides the underlying resource manager. The information required by the vendor must be prefixed with `rm_name:`, which is the published name of the vendor's transaction (XA) interface followed immediately by a colon (:).

A 0-length string value for `tuxTgroupOpenInfo` means that the resource manager for this group (if specified) does not require any application-specific information to open access to the resource.

**Note:** Run-time modifications to this object do not affect active servers in the group.

## **tuxTgroupTMScount**

### Syntax

INTEGER (0..11)

### Access

read-write

### Description

If a non-0 length value is specified for the `tuxTgroupTMSname` object, the `tuxTgroupTMScount` value indicates the number of transaction manager servers to start for the associated group. Otherwise, this object value is ignored.

## **tuxTgroupTMSname**

### Syntax

*DisplayString*(SIZE(0..256)) (up to 78 bytes for Oracle Tuxedo 8.0 or earlier)

### Access

read-write

## Description

Transaction manager server `a.out` associated with this group. This parameter must be specified for any group entry whose servers participate in distributed transactions (transactions across multiple resource managers and possibly machines that are started with `tpbegin(3)` and ended with `tpcommit(3)` or `tpabort(3)`).

The value `TMS` is reserved to indicate use of the null XA interface. If a non-empty value other than `TMS` is specified, a `tuxTmachineTlogDevice` must be specified for the machine(s) associated with the primary and secondary logical machines for this object

A unique server identifier is selected automatically for each TM server, and the servers are restartable an unlimited number of times.

## tuxTgroupEncryptionRequired

### Syntax

```
INTEGER { yes(1) | no(2) }
```

### Access

read-write

### Description

If set to “yes,” every application service in this group requires an encrypted message buffer.

## tuxTgroupSignatureRequired

### Syntax

```
INTEGER { yes(1) | no(2) }
```

### Access

read-write

### Description

If set to “yes,” every application service in this group requires a valid digital signature on its input message buffer.

## tuxTmachineTable

The tuxTmachineTable group contains objects that represent application characteristics pertaining to a particular machine. The object values represent machine characteristics, per-machine sizing, statistics, customization options, and UNIX system filenames. This group is available for configured-inactive as well as configured-active machines in the application.

The index into this table is tuxTmachinePmid. To create a new row, issue a SET request for a non-existing row that specifies at least the values for tuxTmachineLmid, tuxTmachineTuxDir, tuxTmachineTuxConfig, and tuxTmachineAppDir. For a multi-machine Tuxedo application, tuxTmachineNaddr, tuxTmachineNlsAddr, and tuxTmachineBridge must also be specified.

Object Name	Object ID
tuxTmachinePmid	.1.3.6.1.4.1.140.300.5.1.1.1
tuxTmachineLmid	.1.3.6.1.4.1.140.300.5.1.1.2
tuxTmachineTuxConfig	.1.3.6.1.4.1.140.300.5.1.1.3
tuxTmachineTuxDir	.1.3.6.1.4.1.140.300.5.1.1.4
tuxTmachineAppDir	.1.3.6.1.4.1.140.300.5.1.1.5
tuxTmachineState	.1.3.6.1.4.1.140.300.5.1.1.6
tuxTmachineUid	.1.3.6.1.4.1.140.300.5.1.1.7
tuxTmachineGid	.1.3.6.1.4.1.140.300.5.1.1.8
tuxTmachineEnvFile	.1.3.6.1.4.1.140.300.5.1.1.9
tuxTmachinePerm	.1.3.6.1.4.1.140.300.5.1.1.10
tuxTmachineUlogPfx	.1.3.6.1.4.1.140.300.5.1.1.11
tuxTmachineType	.1.3.6.1.4.1.140.300.5.1.1.12
tuxTmachineMaxAccessers	.1.3.6.1.4.1.140.300.5.1.1.13
tuxTmachineMaxConv	.1.3.6.1.4.1.140.300.5.1.1.14
tuxTmachineMaxGtt	.1.3.6.1.4.1.140.300.5.1.1.15
tuxTmachineMaxWsClients	.1.3.6.1.4.1.140.300.5.1.1.16

<b>Object Name</b>	<b>Object ID</b>
tuxTmachineMaxAclCache	.1.3.6.1.4.1.140.300.5.1.1.17
tuxTmachineTlogDevice	.1.3.6.1.4.1.140.300.5.1.1.18
tuxTmachineTlogName	.1.3.6.1.4.1.140.300.5.1.1.19
tuxTmachineTlogSize	.1.3.6.1.4.1.140.300.5.1.1.20
tuxTmachineBridge	.1.3.6.1.4.1.140.300.5.1.1.21
tuxTmachineNaddr	.1.3.6.1.4.1.140.300.5.1.1.22
tuxTmachineNlsaddr	.1.3.6.1.4.1.140.300.5.1.1.23
tuxTmachineCmpLimit	.1.3.6.1.4.1.140.300.5.1.1.24
tuxTmachineTmNetLoad	.1.3.6.1.4.1.140.300.5.1.1.25
tuxTmachineSpinCount	.1.3.6.1.4.1.140.300.5.1.1.26
tuxTmachineRole	.1.3.6.1.4.1.140.300.5.1.1.27
tuxTmachineMinor	.1.3.6.1.4.1.140.300.5.1.1.28
tuxTmachineRelease	.1.3.6.1.4.1.140.300.5.1.1.29
tuxTmachineMaxPendingBytes	.1.3.6.1.4.1.140.300.5.1.1.30
tuxMaxMachineObjects (Tuxedo 8.0 or later)	.1.3.6.1.4.1.140.300.5.1.1.35
tuxTmachineEncryptionRequired	.1.3.6.1.4.1.140.300.5.1.1.50
tuxTmachineSignatureRequired	.1.3.6.1.4.1.140.300.5.1.1.60

## tuxTmachinePmid

### Syntax

*DisplayString*(SIZE(1..30))

### Access

read-write

## Description

Physical machine identifier. This identifier should match the UNIX system nodename returned by the `uname -n` command when run on the identified system. For a Windows NT system, this identifier should match the computer name and the name configured with the name server.

**Note:** This object can be set only during row creation.

## tuxTmachineLmid

### Syntax

*DisplayString*(SIZE(1..30))

### Access

read-write

### Description

Logical machine identifier.

**Note:** This object can be set only during row creation.

## tuxTmachineTuxConfig

### Syntax

*DisplayString*(SIZE(2..256)) (up to 78 bytes for Oracle Tuxedo 8.0 or earlier)

### Access

read-write

### Description

Absolute pathname of the file or device where the binary Tuxedo system configuration file is found on this machine. The administrator need only maintain one such file, namely the one identified by the `tuxTmachineTuxConfig` value on the master machine. The information contained in this file is automatically propagated to all other `tuxTmachineTable` objects as they are activated. See `tuxTmachineEnvFile` for a discussion of how the `tuxTmachineTuxConfig` value is used in the environment.

## tuxTmachineTuxDir

### Syntax

*DisplayString* (SIZE(2..256)) (up to 78 bytes for Oracle Tuxedo 8.0 or earlier)

### Access

read-write

### Description

Absolute pathname of the directory where the Tuxedo system software is found on this machine. See `tuxTmachineEnvFile` that follows for a discussion about how the `tuxTmachineTuxDir` value is used in the environment.

## tuxTmachineAppDir

### Syntax

*DisplayString* (SIZE (2..256)) (up to 78 bytes for Oracle Tuxedo 8.0 or earlier)

### Access

read-write

### Description

Colon-separated list of application directory absolute pathnames. The first directory serves as the current directory for all application and administrative servers booted on this machine. All directories in the list are searched when application servers are started. See `tuxTmachineEnvFile` for a discussion of how the `tuxTmachineAppDir` value is used in the environment.

## tuxTmachineState

### Syntax

```
INTEGER { active(1) | inactive(2) | partitioned(3) | invalid(4) |  
re-activate(5) | cleaning(7) }
```

### Access

read-write

## Description

The values for GET and SET operations are as follows:

GET: {active(1) | inactive(2) | partitioned(3)}

A GET operation retrieves configuration and run-time information for the selected `tuxTmachineTable` instance(s). The following states indicate the meaning of a `tuxTmachineState` returned in response to a GET request. States not listed are not returned

active(1)

`tuxTmachineTable` instance defined and active (administrative servers, that is, DBBL, BBL, and BRIDGE).

inactive(2)

`tuxTmachineTable` instance defined and inactive.

partitioned(3)

`tuxTmachineTable` instance defined, listed in accessible bulletin boards as active, but currently unreachable.

SET: {active(1) | inactive(2) | invalid(4) | re-activate(5) | cleaning(7)}

A SET operation updates configuration and run-time information for the selected `tuxTmachineTable` instance. The following states indicate the meaning of a `tuxTmachineState` set in a SET request. States not listed can not be set.

active(1)

Activate the `tuxTmachineTable` instance. Necessary administrative servers such as the DBBL, BBL, and BRIDGE are started on the indicated site as well as application servers configured to run on that site. State change is allowed only when the machine is in the `inactive(2)` state. Successful return leaves the object in the `active(1)` state.

inactive(2)

Deactivate the `tuxTmachineTable` instance. Necessary administrative servers such as the BBL and BRIDGE are stopped on the indicated site as well as application servers running on that site. State change allowed only when the machine is in the `active(1)` state and when no other application resources are active on the indicated machine. Successful return leaves the object in the `inactive(2)` state.

invalid(4)

Delete `tuxTmachineTable` instance for application. State change is allowed only when the machine is in the `inactive(2)` state. Successful return leaves the object in the `invalid(4)` state.

re-activate(5)

Activate the `tuxTmachineTable` instance. Necessary administrative servers such as the DBBL, BBL, and BRIDGE are started on the indicated site. State change is allowed only

when the machine is in either the `active(1)` or `inactive(2)` state. Successful return leaves the object in the `active(1)` state.

`cleaning(7)`

Initiate cleanup/scanning activities on and relating to the indicated machine. If there are dead clients or servers on the machine, they are detected at this time. If the machine has been partitioned from the application master site, then global bulletin board entries for that machine are removed. This combination is allowed when the application is in the `active(1)` state and the `tuxTmachineTable` instance is in either the `active(1)` or `partitioned(3)` state. Successful return for a non-partitioned machine leaves the state unchanged. Successful return for a partitioned machine leaves the object in the `inactive(2)` state.

**Note:** State change to `inactive(2)` is allowed only for non-master machines. The master site administrative processes are deactivated through the `tuxTdomain` group.

## tuxTmachineUid

### Syntax

INTEGER

### Access

read-write

### Description

UNIX system user-identifier for the Tuxedo application administrator on this machine.

Administrative commands such as `tmboot(1)`, `tmshutdown(1)`, and `tmadmin(1)` must run as the indicated user on this machine. Application and administrative servers on this machine are started as this user.

**Note:** This object is a UNIX-system-specific object that cannot be returned if the platform on which the application is being run is not UNIX-based.

## tuxTmachineGid

### Syntax

INTEGER

## Access

read-write

## Description

UNIX system group identifier for the Tuxedo application administrator on this machine. Administrative commands such as `tmboot(1)`, `tmshutdown(1)`, and `tmadmin(1)` must run as part of the indicated group on this machine. Application and administrative servers on this machine are started as part of this group.

**Note:** This object is a UNIX-system-specific object that cannot be returned if the platform on which the application is being run is not UNIX-based.

## tuxTmachineEnvFile

### Syntax

*DisplayString*(SIZE(2..256)) (up to 78 bytes for Oracle Tuxedo 8.0 or earlier)

### Access

read-write

### Description

Environment file for clients and servers running on this machine.

## tuxTmachinePerm

### Syntax

*DisplayString*(SIZE(1..9))

### Access

read-write

### Description

UNIX system permissions associated with the shared memory bulletin board created on this machine. Default UNIX system permissions for system and application message queues.

**Note:** Modifications to this object for an active object do not affect running servers or clients.

**Note:** This object is a UNIX-system-specific object that cannot be returned if the platform on which the application is being run is not UNIX-based.

## tuxTmachineUlogPfx

### Syntax

*DisplayString*(SIZE(0..256)) (up to 78 bytes for Oracle Tuxedo 8.0 or earlier)

### Access

read-write

### Description

Absolute pathname prefix of the path for the `userlog(3)` file on this machine. The `userlog(3)` file name is formed by appending the string `.mmddyy` to the `tuxTmachineUlogPfx` object value. `.mmddyy` represents the month, day, and year that the messages were generated. All application and system `userlog(3)` messages generated by clients and servers running on this machine are directed to this file.

**Note:** Modifications to this object for an active object do not affect running servers or clients.

## tuxTmachineType

### Syntax

*DisplayString*(SIZE(1..15))

### Access

read-write

### Description

Machine type. The `tuxTmachineType` value is used to group machines into groups of like data representations. Data encoding is not performed when communicating between machines of identical types. This object can be given any string value; values are used only for comparison. Distinct `tuxTmachineType` objects should be set when the application spans a heterogeneous network of machines or when compilers generate dissimilar structure representations. The default value for `tuxTmachineType`, a 0-length string, matches any other machine having a 0-length string as its `tuxTmachineType` object value.

## **tuxTmachineMaxAccessers**

### **Syntax**

INTEGER (1..32767)

### **Access**

read-write

### **Description**

Maximum number of clients and servers that can have access to the bulletin board on this machine at one time. System administration processes such as the BBL and tmadmin need not be accounted for in this figure, but all application servers and clients and TMS servers should be counted. If the application is booting workstation listeners on this site, then both the listeners and the potential number of workstation handlers that can be booted should be counted.

## **tuxTmachineMaxConv**

### **Syntax**

INTEGER (0..32767)

### **Access**

read-write

### **Description**

Maximum number of simultaneous conversations in which clients and servers on this machine can be involved.

## **tuxTmachineMaxGtt**

### **Syntax**

INTEGER (0..32767)

### **Access**

read-write

## Description

Maximum number of simultaneous global transactions in which this machine can be involved.

## **tuxTmachineMaxWsClients**

### Syntax

INTEGER (0..32767)

### Access

read-write

## Description

Number of entries for accessers on this machine to be reserved for workstation clients. The number specified here takes a portion of the total slots for accessers specified with the `tuxTmachineMaxAccessers` object. The appropriate setting of this parameter helps to conserve IPC resources because workstation client access to the system is multiplexed through a Tuxedo system supplied surrogate, the workstation handler. It is an error to set this number greater than `tuxTmachineMaxAccessers`.

## **tuxTmachineMaxAclCache**

### Syntax

INTEGER (10..32000)

### Access

read-write

## Description

Number of entries in the cache used for ACL entries when `tuxTdomainSecurity` is set to `acl(4)` or `mandatory-acl(5)`. The appropriate setting of this parameter helps to conserve shared memory resources and yet reduce the number of disk access to do ACL checking.

## **tuxTmachineTlogDevice**

### Syntax

*DisplayString*(SIZE(0..256)) (up to 64 bytes for Oracle Tuxedo 8.0 or earlier)

### Access

read-write

### Description

The device (raw slice) or UNIX system file containing the Tuxedo system filesystem that holds the DTP transaction log for this machine. The DTP transaction log is stored as a Tuxedo system VTOC table on the device. This device or file can be the same as that specified for the `tuxTmachineTuxConfig` object for this machine.

## **tuxTmachineTlogName**

### Syntax

*DisplayString*(SIZE(0..30))

### Access

read-write

### Description

The name of the DTP transaction log for this machine. If more than one DTP transaction log exists on the same `tuxTmachineTlogDevice`, they must have unique names. `tuxTmachineTlogName` must be different from the name of any other table on the `tuxTmachineTlogDevice` where the DTP transaction log table is created.

## **tuxTmachineTlogSize**

### Syntax

INTEGER (1..2048)

### Access

read-write

## Description

The numeric size, in pages, of the DTP transaction log for this machine. The `tuxTmachineTlogSize` object value is subject to limits based on available space in the Tuxedo system filesystem identified by the `tuxTmachineTlogDevice` object.

## tuxTmachineBridge

### Syntax

*DisplayString*(SIZE(0..78))

### Access

read-write

### Description

Device name to be used by the BRIDGE process placed on this logical machine to access the network. The `tuxTmachineBridge` value is a required value for participation in a networked application through a TLI-based Tuxedo system binary. This object value is not needed for sockets-based Tuxedo system binaries.

## tuxTmachineNaddr

### Syntax

*DisplayString*(SIZE(0..256)) (up to 78 bytes for Oracle Tuxedo 8.0 or earlier)

### Access

read-write

### Description

Specifies the complete network address to be used by the BRIDGE process placed on the logical machine as its listening address. The listening address for a BRIDGE is the means by which it is contacted by other BRIDGE processes participating in the application. This object must be set if the logical machine is to participate in a networked application, that is, if the LAN option is set in the `tuxTdomainOptions` object value.

If *DisplayString* has the form `0xhex-digits` or `\\xhex-digits`, it must contain an even number of valid hexadecimal digits. These forms are translated internally into a character array that contains the hexadecimal representations of the specified string.

## **tuxTmachineNlsaddr**

### Syntax

*DisplayString* (SIZE(0..256)) (up to 78 bytes for Oracle Tuxedo 8.0 or earlier)

### Access

read-write

### Description

Network address used by the `tlisten(1)` process servicing the network on the node identified by this logical machine. This network address has the same format as that specified for the `tuxTmachineNaddr` object.

This object must be set if the logical machine is to participate in a networked application, that is, if the LAN option is set in the `tuxTdomainOptions` object value.

## **tuxTmachineCmpLimit**

### Syntax

*DisplayString*

### Access

read-write

### Description

Threshold message size at which compression occurs for remote traffic and, optionally, local traffic. Remote and local can be either non-negative numeric values or the string `MAXLONG` that is dynamically translated to the maximum long setting for the machine. Setting only the remote value defaults local to `MAXLONG`.

## **tuxTmachineTmNetLoad**

### Syntax

INTEGER (0..32767)

### Access

read-write

## Description

Service load added to any remote service evaluated during load balancing on this machine.

## tuxTmachineSpinCount

### Syntax

INTEGER

### Access

read-write

## Description

Spincount used on this machine for pre-ticket user-level semaphore access. Default values are built into the Tuxedo system binaries on each machine. For tuning purposes, these defaults can be overridden at run-time using `tuxTmachineSpinCount`. The spincount can be reset to the default built-in value for the site by resetting `tuxTmachineSpinCount` to 0.

## tuxTmachineRole

### Syntax

INTEGER { master(1) | backup(2) | other(3) }

### Access

read-only

## Description

The role of this machine in the application.

`master(1)`

Indicates that this machine is the master machine,

`backup(2)`

Indicates that it is the backup master machine, and

`other(3)`

Indicates that the machine is neither the master nor the backup master machine.

## **tuxTmachineMinor**

### Syntax

INTEGER

### Access

read-only

### Description

The Tuxedo system minor protocol release number for this machine.

## **tuxTmachineRelease**

### Syntax

INTEGER

### Access

read-only

### Description

The Tuxedo system major protocol release number for this machine. This value can be different from the `tuxTmachineSWrelease` for the same machine.

## **tuxTmachineMaxPendingBytes**

### Syntax

INTEGER

### Access

read-write

### Description

Specifies a limit for the amount of space that can be allocated for messages waiting to be transmitted by the BRIDGE process. The minimum value is 100000.

## **tuxMachineMaxObjects (Tuxedo 8.0 or later)**

### Syntax

INTEGER

### Access

read-write

### Description

The maximum number of CORBA objects that can be accommodated in the Active Object Map tables in the bulletin board.

## **tuxTmachineEncryptionRequired**

### Syntax

INTEGER { yes(1) | no(2) }

### Access

read-write

### Description

If set to “yes,” every application service on this machine requires an encrypted input message buffer.

## **tuxTmachineSignatureRequired**

### Syntax

INTEGER { yes(1) | no(2) }

### Access

read-write

### Description

If set to “yes,” every application service on this machine requires a valid digital signature on its input message buffer.

## tuxTmachineActive

The `tuxTmachineActive` group contains objects that represent run-time statistics on the local machine if the machine is active (that is, some component of the application is active on the machine). Objects in this group are only accessible through a Tuxedo SNMP agent installed on the local machine.

<b>Object Name</b>	<b>Object ID</b>
<code>tuxTmachineCurAccessers</code>	<code>.1.3.6.1.4.1.140.300.5.2.1</code>
<code>tuxTmachineCurClients</code>	<code>.1.3.6.1.4.1.140.300.5.2.2</code>
<code>tuxTmachineCurConv</code>	<code>.1.3.6.1.4.1.140.300.5.2.3</code>
<code>tuxTmachineCurGTT</code>	<code>.1.3.6.1.4.1.140.300.5.2.4</code>
<code>tuxTmachineCurLoad</code>	<code>.1.3.6.1.4.1.140.300.5.2.5</code>
<code>tuxTmachineCurWsClients</code>	<code>.1.3.6.1.4.1.140.300.5.2.6</code>
<code>tuxTmachineHwAccessers</code>	<code>.1.3.6.1.4.1.140.300.5.2.7</code>
<code>tuxTmachineHwClients</code>	<code>.1.3.6.1.4.1.140.300.5.2.8</code>
<code>tuxTmachineHwConv</code>	<code>.1.3.6.1.4.1.140.300.5.2.9</code>
<code>tuxTmachineHwGTT</code>	<code>.1.3.6.1.4.1.140.300.5.2.10</code>
<code>tuxTmachineHwWsClients</code>	<code>.1.3.6.1.4.1.140.300.5.2.11</code>
<code>tuxTmachineNumConv</code>	<code>.1.3.6.1.4.1.140.300.5.2.12</code>
<code>tuxTmachineNumDequeue</code>	<code>.1.3.6.1.4.1.140.300.5.2.13</code>
<code>tuxTmachineNumEnqueue</code>	<code>.1.3.6.1.4.1.140.300.5.2.14</code>
<code>tuxTmachineNumPost</code>	<code>.1.3.6.1.4.1.140.300.5.2.15</code>
<code>tuxTmachineNumReq</code>	<code>.1.3.6.1.4.1.140.300.5.2.16</code>
<code>tuxTmachineNumSubscribe</code>	<code>.1.3.6.1.4.1.140.300.5.2.17</code>
<code>tuxTmachineNumTran</code>	<code>.1.3.6.1.4.1.140.300.5.2.18</code>
<code>tuxTmachineNumTranAbt</code>	<code>.1.3.6.1.4.1.140.300.5.2.19</code>

Object Name	Object ID
tuxTmachineNumTranCmt	.1.3.6.1.4.1.140.300.5.2.20
tuxTmachineLicExpires	.1.3.6.1.4.1.140.300.5.2.21
tuxTmachineLicMaxUsers	.1.3.6.1.4.1.140.300.5.2.22
tuxTmachineLicSerial	.1.3.6.1.4.1.140.300.5.2.23
tuxTmachinePageSize	.1.3.6.1.4.1.140.300.5.2.24
tuxTmachineSWrelease	.1.3.6.1.4.1.140.300.5.2.25
tuxTmachineHwAclCache	.1.3.6.1.4.1.140.300.5.2.26
tuxTmachineAclCacheHits	.1.3.6.1.4.1.140.300.5.2.27
tuxTmachineAclCacheAccess	.1.3.6.1.4.1.140.300.5.2.28
tuxTmachineAclFail	.1.3.6.1.4.1.140.300.5.2.29
tuxTmachineWkCompleted	.1.3.6.1.4.1.140.300.5.2.30
tuxTmachineWkInitiated	.1.3.6.1.4.1.140.300.5.2.31
tuxMachineCurObjects (Tuxedo 8.0 or later)	.1.3.6.1.4.1.140.300.5.2.36
tuxMachineHwObjects (Tuxedo 8.0 or later)	.1.3.6.1.4.1.140.300.5.2.41

## tuxTmachineCurAccessers

### Syntax

INTEGER (0..32767)

### Access

read-only

### Description

Number of clients and servers that currently access the application either directly on this machine or through a workstation handler on this machine.

## **tuxTmachineCurClients**

### Syntax

INTEGER (0..32767)

### Access

read-only

### Description

Number of clients, both native and workstation, currently logged in to this machine.

## **tuxTmachineCurConv**

### Syntax

INTEGER (0..32767)

### Access

read-only

### Description

Number of active conversations with participants on this machine.

## **tuxTmachineCurGTT**

### Syntax

INTEGER (0..32767)

### Access

read-only

### Description

Number of in use transaction table entries on this machine.

## tuxTmachineCurLoad

### Syntax

INTEGER

### Access

read-only

### Description

Current service load enqueued on this machine.

**Note:** If the `tuxTdomainLoadBalance` object is `no(2)` or the `tuxTdomainModel` object is `multi-machine(2)`, then an FML32 NULL value (0) is returned.

## tuxTmachineCurWsClients

### Syntax

INTEGER (0..32767)

### Access

read-only

### Description

Number of workstation clients currently logged in to this machine.

## tuxTmachineHwAccessers

### Syntax

INTEGER (0..32767)

### Access

read-only

### Description

High water number of clients and servers accessing the application either directly on this machine or through a workstation handler on this machine.

## **tuxTmachineHwClients**

### Syntax

INTEGER (0..32767)

### Access

read-only

### Description

High water number of clients, both native and workstation, logged in to this machine.

## **tuxTmachineHwConv**

### Syntax

INTEGER (0..32767)

### Access

read-only

### Description

High water number of active conversations with participants on this machine.

## **tuxTmachineHwGTT**

### Syntax

INTEGER (0..32767)

### Access

read-only

### Description

High water number of in use transaction table entries on this machine.

## **tuxTmachineHwWsClients**

### Syntax

INTEGER (0..32767)

### Access

read-only

### Description

High water number of workstation clients currently logged in to this machine.

## **tuxTmachineNumConv**

### Syntax

INTEGER

### Access

read-only

### Description

Number of `tpconnect(3)` operations performed from this machine.

## **tuxTmachineNumDequeue**

### Syntax

INTEGER

### Access

read-only

### Description

Number of `tpdequeue(3)` operations performed from this machine.

## **tuxTmachineNumEnqueue**

### Syntax

INTEGER

### Access

read-only

### Description

Number of `tpenqueue(3)` operations performed from this machine.

## **tuxTmachineNumPost**

### Syntax

INTEGER

### Access

read-only

### Description

Number of `tppost(3)` operations performed from this machine.

## **tuxTmachineNumReq**

### Syntax

INTEGER

### Access

read-only

### Description

Number of `tpacall(3)` or `tpcall(3)` operations performed from this machine.

## **tuxTmachineNumSubscribe**

### Syntax

INTEGER

### Access

read-only

### Description

Number of `tpsubscribe(3)` operations performed from this machine.

## **tuxTmachineNumTran**

### Syntax

INTEGER

### Access

read-only

### Description

Number of transactions initiated (`tpbegin(3)`) from this machine.

## **tuxTmachineNumTranAbt**

### Syntax

INTEGER

### Access

read-only

### Description

Number of transactions aborted (`tpabort(3)`) from this machine.

## **tuxTmachineNumTranCmt**

### Syntax

INTEGER

### Access

read-only

### Description

Number of transactions committed (`tpcommit(3)`) from this machine.

## **tuxTmachineLicExpires**

### Syntax

*DisplayString*(SIZE(0..78))

### Access

read-only

### Description

Expiration date for the binary on the machine or a 0-length string if binary is not a Tuxedo system master binary.

## **tuxTmachineLicMaxUsers**

### Syntax

INTEGER (0..32767)

### Access

read-only

### Description

Maximum number of licensed users on that machine, or -1 if binary is not a Tuxedo system master binary.

## tuxTmachineLicSerial

### Syntax

*DisplayString*(SIZE(0..78))

### Access

read-only

### Description

Serial number for binary on the machine or a 0-length string if binary is not a Tuxedo system master binary.

## tuxTmachinePageSize

### Syntax

INTEGER

### Access

read-only

### Description

Disk pagesize used on this machine.

## tuxTmachineSWrelease

### Syntax

*DisplayString*(SIZE(0..78))

### Access

read-only

### Description

Software release for binary on that machine or a 0-length string if binary is not a Tuxedo system master binary.

## **tuxTmachineHwAclCache**

### Syntax

INTEGER

### Access

read-only

### Description

High water number of entries used in the ACL cache.

## **tuxTmachineAclCacheHits**

### Syntax

INTEGER

### Access

read-only

### Description

Number of accesses to the ACL cache that resulted in a “hit” (that is, the entry was already in the cache).

## **tuxTmachineAclCacheAccess**

### Syntax

INTEGER

### Access

read-only

### Description

Number of accesses to the ACL cache.

## **tuxTmachineAcIFail**

### Syntax

INTEGER

### Access

read-only

### Description

Number of accesses to the ACL cache that resulted in a access control violation.

## **tuxTmachineWkCompleted**

### Syntax

INTEGER

### Access

read-only

### Description

Total service load dequeued and processed successfully by servers running on this machine. Note that for long running applications this object can wraparound, that is, exceed the maximum value for a long, and start back at 0 again.

## **tuxTmachineWkInitiated**

### Syntax

INTEGER

### Access

read-only

### Description

Total service load enqueued by clients/servers running on this machine. Note that for long running applications this object can wraparound, that is, exceed the maximum value for a long, and start back at 0 again.

## tuxMachineCurObjects (Tuxedo 8.0 or later)

### Syntax

INTEGER

### Access

read-only

### Description

The number of entries in use in the bulletin board object table for this machine.

## tuxMachineHwObjects (Tuxedo 8.0 or later)

### Syntax

INTEGER

### Access

read-only

### Description

The high water mark of entries used in the bulletin board object table for this machine.

## tuxTmsgTable

The `tuxTmsgTable` group contains objects that represent run-time characteristics of the Tuxedo system managed UNIX system message queues. Objects in this table are only accessible through a Tuxedo SNMP agent installed on the local machine. `tuxTmsgId` is the index into this table.

Object Name	Object ID
<a href="#">tuxTmsgId</a>	.1.3.6.1.4.1.140.300.6.1.1.1
<a href="#">tuxTmsgState</a>	.1.3.6.1.4.1.140.300.6.1.1.2
<a href="#">tuxTmsgCurTime</a>	.1.3.6.1.4.1.140.300.6.1.1.3
<a href="#">tuxTmsgCbytes</a>	.1.3.6.1.4.1.140.300.6.1.1.4

Object Name	Object ID
<a href="#">tuxTmsgCtime</a>	.1.3.6.1.4.1.140.300.6.1.1.5
<a href="#">tuxTmsgLrPid</a>	.1.3.6.1.4.1.140.300.6.1.1.6
<a href="#">tuxTmsgLsPid</a>	.1.3.6.1.4.1.140.300.6.1.1.7
<a href="#">tuxTmsgQbytes</a>	.1.3.6.1.4.1.140.300.6.1.1.8
<a href="#">tuxTmsgQnum</a>	.1.3.6.1.4.1.140.300.6.1.1.9
<a href="#">tuxTmsgRtime</a>	.1.3.6.1.4.1.140.300.6.1.1.10
<a href="#">tuxTmsgStime</a>	.1.3.6.1.4.1.140.300.6.1.1.11

## tuxTmsgId

### Syntax

INTEGER

### Access

read-only

### Description

UNIX system message queue identifier.

**Note:** This object is a UNIX-system-specific object that cannot be returned if the platform on which the application is being run is not UNIX-based.

## tuxTmsgState

### Syntax

INTEGER { active(1) }

### Access

read-only

### Description

The values for GET and SET operations are as follows:

GET: active(1)

A GET operation retrieves run-time information for the selected `tuxTmsgTable` object(s). The following state indicates the meaning of a `tuxTmsgState` returned in response to a GET request. States not listed are not returned.

active(1)

`tuxTmsgTable` object active. This state corresponds exactly to the related `tuxTmachineTable` object being active.

SET:

SET operations are not permitted on this group.

## **tuxTmsgCurTime**

### Syntax

INTEGER

### Access

read-only

### Description

Current time, in seconds, since 00:00:00 UTC, January 1, 1970, as returned by the `time(2)` system call on the local host.

## **tuxTmsgCbytes**

### Syntax

INTEGER

### Access

read-only

### Description

Current number of bytes on the queue.

## tuxTmsgCtime

### Syntax

INTEGER

### Access

read-only

### Description

Time of the last `msgctl(2)` operation that changed a member of the `msgqid_ds` structure associated with the queue.

## tuxTmsgLrPid

### Syntax

INTEGER

### Access

read-only

### Description

Process identifier of the last process that read from the queue.

## tuxTmsgLsPid

### Syntax

INTEGER

### Access

read-only

### Description

Process identifier of the last process that wrote to the queue.

## **tuxTmsgQbytes**

### Syntax

INTEGER

### Access

read-only

### Description

Maximum number of bytes allowed on the queue.

## **tuxTmsgQnum**

### Syntax

INTEGER

### Access

read-only

### Description

Number of messages currently on the queue.

## **tuxTmsgRtime**

### Syntax

INTEGER

### Access

read-only

### Description

Time since the last read from the queue.

## tuxTmsgStime

### Syntax

INTEGER

### Access

read-only

### Description

Time since the last write to the queue.

## tuxTqueueTable

The `tuxTqueueTable` group contains objects that represent run-time characteristics of queues in an application. The object values identify and characterize allocated Tuxedo system request queues associated with servers in a running application. They also track statistics related to application workloads associated with each queue object. The index into this table is `tuxTqueueRqAddr`. Objects in this table are only accessible through a Tuxedo SNMP agent installed on the local machine.

Object Name	Object ID
<code>tuxTqueueRqAddr</code>	.1.3.6.1.4.1.140.300.7.1.1.1
<code>tuxTqueueState</code>	.1.3.6.1.4.1.140.300.7.1.1.2
<code>tuxTqueueRqId</code>	.1.3.6.1.4.1.140.300.7.1.1.3
<code>tuxTqueueSrvrCnt</code>	.1.3.6.1.4.1.140.300.7.1.1.4
<code>tuxTqueueTotNqueued</code>	.1.3.6.1.4.1.140.300.7.1.1.5
<code>tuxTqueueTotWkQueued</code>	.1.3.6.1.4.1.140.300.7.1.1.6
<code>tuxTqueueSource</code>	.1.3.6.1.4.1.140.300.7.1.1.7
<code>tuxTqueueNqueued</code>	.1.3.6.1.4.1.140.300.7.1.1.8
<code>tuxTqueueWkQueued</code>	.1.3.6.1.4.1.140.300.7.1.1.9

## tuxTqueueRqAddr

### Syntax

*DisplayString*(SIZE(1..30))

### Access

read-only

### Description

Symbolic address of the request queue. Servers with the same `tuxTsrvrRqAddr` object value are grouped into a Multiple Server Single Queue (MSSQ) set. object values returned with a `tuxTqueueTable` object apply to all active servers associated with this symbolic queue address.

## tuxTqueueState

### Syntax

INTEGER { active(1) | migrating(2) | suspended(3) | partitioned(4) }

### Access

read-only

### Description

The values for GET and SET operations are as follows:

GET: {active(1) |migrating(2) |suspended(3) |partitioned(4) }

A GET operation retrieves run-time information for the selected `tuxTqueueTable` instance(s). The `tuxTqueueTable` group does not address configuration information directly. Configuration-related objects discussed here must be set as part of the related `tuxTsrvrTbl` instances. The following states indicate the meaning of a `tuxTqueueState` returned in response to a GET request. States not listed are not returned.

active(1)

At least one server associated with this `tuxTqueueTable` instance is active(1).

migrating(2)

The server(s) associated with this `tuxTqueueTable` instance is currently in the migrating(2) state. See the `tuxTsrvrTbl` group for more details on this state.

suspended(3)

The server(s) associated with this `tuxTqueueTable` instance is currently in the `suspended(3)` state. See the `tuxTsrvrTbl` group for more details on this state.

partitioned(4)

The server(s) associated with this `tuxTqueueTable` instance is currently in the `partitioned(4)` state. See the `tuxTsrvrTbl` group for more details on this state.

SET:

A SET operation updates run-time information for the selected `tuxTqueueTable` object. State changes are not allowed when updating `tuxTqueueTable` object information. Modification of an existing `tuxTqueueTable` object is allowed only when the object is in the `active(1)` state.

## tuxTqueueRqld

### Syntax

INTEGER

### Access

read-only

### Description

UNIX system message queue identifier.

**Note:** This object is a UNIX-system-specific object that cannot be returned if the platform on which the application is being run is not UNIX-based.

## tuxTqueueSrvrCnt

### Syntax

INTEGER

### Access

read-only

### Description

Number of active servers associated with this queue.

## tuxTqueueTotNqueued

### Syntax

INTEGER

### Access

read-only

### Description

The sum of the queue lengths of this queue while it has been active. This sum includes requests enqueued to and processed by servers that are no longer active on the queue. Each time a new request is assigned to the queue, the sum is incremented by the length of the queue immediately before the new request is enqueued.

**Note:** If the `tuxTdomainLoadBalance` object is `no(2)` or the `tuxTdomainModel` object is `multi-machine(2)`, then `tuxTqueueTotNqueued` is not returned. In the same configuration, updates to this object are ignored. Consequently, when this object is returned `tuxTqueueSource` has the same value as the local host.

## tuxTqueueTotWkQueued

### Syntax

INTEGER

### Access

read-only

### Description

The sum of the workloads enqueued to this queue while it has been active. This sum includes requests enqueued to and processed by servers that are no longer active on the queue. Each time a new request is assigned to the queue, the sum is incremented by the workload on the queue immediately before the new request is enqueued.

**Note:** If the `tuxTdomainLoadBalance` object is `no(2)` or the `tuxTdomainModel` object is `multi-machine(2)`, then `tuxTqueueTotWkQueued` is not returned. In the same configuration, updates to this object are ignored. Consequently, when this object is returned `tuxTqueueSource` has the same value as the local host.

## tuxTqueueSource

### Syntax

*DisplayString*(SIZE(1..30))

### Access

read-only

### Description

Logical machine from which local object values are retrieved.

## tuxTqueueNqueued

### Syntax

INTEGER

### Access

read-only

### Description

Number of requests currently enqueued to this queue from the `tuxTqueueSource` logical machine. This value is incremented at enqueue time and decremented when the server dequeues the request.

**Note:** If the `tuxTdomainLoadBalance` object is `no(2)` or the `tuxTdomainModel` object is `multi-machine(2)`, then `tuxTqueueNqueued` is not returned. Consequently, when this object is returned `tuxTqueueSource` has the same value as the local host.

## tuxTqueueWkQueued

### Syntax

INTEGER

### Access

read-only

## Description

Workload currently enqueued to this queue from the `tuxTqueueSource` logical machine. If the `tuxTdomainModel` object is set to `single-machine` (1) and the `tuxTdomainLoadBalance` object is set to `yes` (1), then `tuxTqueueWkQueued` reflects the application-wide workload enqueued to this queue. However, if `tuxTdomainModel` is set to `multi-machine` (2) and `tuxTdomainLoadBalance` is set to `yes` (1), then `tuxTqueueWkQueued` reflects the workload enqueued to this queue from the `tuxTqueueSource` logical machine during a recent timespan. The `tuxTqueueWkQueued` value is used for load balancing purposes. In order not discriminate against newly started servers, `tuxTqueueWkQueued` is zeroed out on each machine periodically by the BBL.

## tuxTroutingTable

The `tuxTroutingTable` group contains objects that represent configuration characteristics of routing specifications for an application. The object values identify and characterize application data-dependent routing criteria with respect to field names, buffer types, and routing definitions. This table also represents configuration objects for factory-based routing for Tuxedo 8.0 or later applications. Object `tuxRoutingFieldType` (Tuxedo 8.0 or later) is valid only for factory-based routing. Object `tuxTroutingBufType` is valid only for service-based routing.

The index into this table consists of the following objects: `tuxTroutingName`, `tuxRoutingType`, and `tuxInternalIdx`.

Object `tuxRoutingFieldType` is valid only for factory-based routing and is supported only for Tuxedo 8.0 or later applications.

Object `tuxTroutingBufType` is valid only for service-based routing.

When specifying the index in SET requests, `tuxInternalIdx` is used as an index.

For factory-based routing, `tuxInternalIdx` must always have a value of `-`.

For service-based routing, `tuxInternalIdx` should equal the first 30 characters in `tuxTroutingBufType`.

To create a new row in the table, it is necessary to issue a SET request for a non-existing row specifying the values of all objects applicable to the `tuxRoutingType`.

Object Name	Object ID
tuxTroutingName	.1.3.6.1.4.1.140.300.8.1.1.1
tuxTroutingBufType	.1.3.6.1.4.1.140.300.8.1.1.2
tuxTroutingField	.1.3.6.1.4.1.140.300.8.1.1.3
tuxTroutingRanges	.1.3.6.1.4.1.140.300.8.1.1.4
tuxTroutingState	.1.3.6.1.4.1.140.300.8.1.1.5
tuxRoutingType	.1.3.6.1.4.1.140.300.8.1.1.6
tuxRoutingFieldType (Tuxedo 8.0 or later)	.1.3.6.1.4.1.140.300.8.1.1.7
tuxInternalIdx	.1.3.6.1.4.1.140.300.8.1.1.8

## tuxTroutingName

### Syntax

*DisplayString*(SIZE(1..15))

### Access

read-write

### Description

Routing criterion name.

**Note:** This object can be set only during row creation.

## tuxTroutingBufType

### Syntax

*DisplayString*(SIZE(1..256))

### Access

read-write

## Description

List of types and subtypes of data buffers for which this routing entry is valid. A maximum of 32 type/subtype combinations are allowed. The types are restricted to one of FML, VIEW, X\_C\_TYPE, or X\_COMMON. No subtype can be specified for type FML, and subtypes are required for types VIEW, X\_C\_TYPE, and X\_COMMON (\* is not allowed). Note that subtype names should not contain semicolon, colon, comma, or asterisk characters. Duplicate type/subtype pairs cannot be specified for the same routing criterion name. More than one routing entry can have the same criterion name as long as the type/subtype pairs are unique. If multiple buffer types are specified for a single routing entry, the data types of the routing field for each buffer type must be the same.

**Note:** This object is applicable only for service-based routing.

**Note:** This object can be set only during row creation.

## tuxRoutingField

### Syntax

*DisplayString*(SIZE(1..30))

### Access

read-write

### Description

Routing field name.

For Service-based Routing: This field is assumed to be an FML buffer or view field name that is identified in an FML field table (using the FLDTBLDIR and FIELDTBLS environment) or an FML view table (using the VIEWDIR and VIEWFILES environment), respectively. This information is used to get the associated field value for data dependent routing during the sending of a message.

For factory-based routing: This is assumed to be a field that is specified in an NVList parameter to:

```
PortableServer::POA::create_reference_with_criteria
```

for an interface that has this factory routing criteria associated with it. See the Tuxedo 8.0 or later documentation for more details.

## tuxTroutingRanges

### Syntax

```
DisplayString(SIZE(1..2048))
```

### Access

read-write

### Description

The ranges and associated server groups for a routing criterion are as follows:

```

criterion: range: group
range: value | lower - upper | *
lower: value
upper: value
value: MIN | MAX | numeric | string
group: string | *
numeric: [+ | -]digits[.digits][e | E[ + | - ] digit
digit: 0-9
digits: digit[digit]

```

\ can be used to escape the single-quote character in strings.

*lower* must be less than *upper*. A group specified as a string must specify a valid tuxTgroupName.

## tuxTroutingState

### Syntax

```
INTEGER { valid(1) | unknown(2) | invalid(3) }
```

### Access

read-write

### Description

The values for GET and SET operations are as follows:

GET: valid(1)

A GET operation retrieves configuration information for the selected `tuxTroutingTable` instance(s). The following state indicates the meaning of a `tuxTroutingState` returned in response to a GET request. States not listed are not returned.

valid(1)

`tuxTroutingTable` instance is defined. Note that `valid(1)` is the only valid state for this group. Routing criteria are never active; rather, they are associated through the configuration with service names and are acted upon at run-time to provide data dependent routing.

SET: invalid(3)

A SET operation updates configuration information for the selected `tuxTroutingTable` instance. The following state indicates the meaning of a `tuxTroutingState` set in a SET request. States not listed cannot be set.

invalid(3)

Delete `tuxTroutingTable` instance for application. State change allowed only when in the `valid(1)` state. Successful return leaves the object in the `invalid(2)` state.

## tuxRoutingType

### Syntax

```
INTEGER { service(1) |factory(2) }
```

### Access

read-write

### Description

`service(1)`

Specifies that routing criteria apply to data-dependent routing for an Oracle Tuxedo service.

`factory(2)`

Specifies that the routing criterion applies to factory-based routing for a CORBA interface.

**Note:** The routing type affects the validity and possible values for other objects defined for this table.

**Note:** This object can be set during row creation only.

## tuxRoutingFieldType (Tuxedo 8.0 or later)

### Syntax

```
INTEGER { short(1) | long(2) | float(3) | double(4) | char(5) | string(6) }
```

### Access

read-write

### Description

This object specifies the type of `tuxTroutingField` on which this routing criterion is defined. Its value is valid only for factory-based routing.

**Note:** This object can be set only during row creation.

## tuxInternalIdx

### Syntax

```
DisplayString(SIZE(1..30))
```

### Access

read-write

### Description

This object is used as an index of this table instead of `tuxTroutingBufType` (for service-based routing) or `tuxTroutingField` (for factory-based routing) to reduce the size of the index. Its value for service-based routing (`tuxRoutingType = service(1)`) is equal to the first 30 characters in `tuxTroutingBufType`.

In case of entries for factory-based routing (`tuxRoutingType = factory(2)`), the value is always `tuxTroutingField`.

**Note:** This object can be set only during row creation.

## tuxTsrvrTbl

The `tuxTsrvrTbl` group contains objects that represent configuration and run-time characteristics of servers within an application. The object values identify and characterize configured servers as well as provide run-time tracking of statistics and resources associated with each server object.

The index into this table is provided by the objects `tuxTsrvrGrpNo` and `tuxTsrvrId`. To create a new row in the table, it is necessary to issue a SET request specifying the values of at least `tuxTsrvrGrp` and `tuxTsrvrName`.

<b>Object Name</b>	<b>Object ID</b>
<code>tuxTsrvrGrp</code>	.1.3.6.1.4.1.140.300.20.1.1.1
<code>tuxTsrvrId</code>	.1.3.6.1.4.1.140.300.20.1.1.2
<code>tuxTsrvrName</code>	.1.3.6.1.4.1.140.300.20.1.1.3
<code>tuxTsrvrGrpNo</code>	.1.3.6.1.4.1.140.300.20.1.1.4
<code>tuxTsrvrState</code>	.1.3.6.1.4.1.140.300.20.1.1.5
<code>tuxTsrvrBaseSrvId</code>	.1.3.6.1.4.1.140.300.20.1.1.6
<code>tuxTsrvrClOpt</code>	.1.3.6.1.4.1.140.300.20.1.1.7
<code>tuxTsrvrEnvFile</code>	.1.3.6.1.4.1.140.300.20.1.1.8
<code>tuxTsrvrGrace</code>	.1.3.6.1.4.1.140.300.20.1.1.9
<code>tuxTsrvrMaxgen</code>	.1.3.6.1.4.1.140.300.20.1.1.10
<code>tuxTsrvrMax</code>	.1.3.6.1.4.1.140.300.20.1.1.11
<code>tuxTsrvrMin</code>	.1.3.6.1.4.1.140.300.20.1.1.12
<code>tuxTsrvrRcmd</code>	.1.3.6.1.4.1.140.300.20.1.1.13
<code>tuxTsrvrRestart</code>	.1.3.6.1.4.1.140.300.20.1.1.14
<code>tuxTsrvrSequence</code>	.1.3.6.1.4.1.140.300.20.1.1.15
<code>tuxTsrvrSystemAccess</code>	.1.3.6.1.4.1.140.300.20.1.1.16
<code>tuxTsrvrConv</code>	.1.3.6.1.4.1.140.300.20.1.1.17
<code>tuxTsrvrReplyQ</code>	.1.3.6.1.4.1.140.300.20.1.1.18
<code>tuxTsrvrRpPerm</code>	.1.3.6.1.4.1.140.300.20.1.1.19
<code>tuxTsrvrRqAddr</code>	.1.3.6.1.4.1.140.300.20.1.1.20
<code>tuxTsrvrRqPerm</code>	.1.3.6.1.4.1.140.300.20.1.1.21

Object Name	Object ID
tuxTsrvrGeneration	.1.3.6.1.4.1.140.300.20.1.1.22
tuxTsrvrPid	.1.3.6.1.4.1.140.300.20.1.1.23
tuxTsrvrRpid	.1.3.6.1.4.1.140.300.20.1.1.24
tuxTsrvrRqId	.1.3.6.1.4.1.140.300.20.1.1.25
tuxTsrvrTimeRestart	.1.3.6.1.4.1.140.300.20.1.1.26
tuxTsrvrTimeStart	.1.3.6.1.4.1.140.300.20.1.1.27
tuxTsrvrMinDispatchThreads	.1.3.6.1.4.1.140.300.20.1.1.40
tuxTsrvrMaxDispatchThreads	.1.3.6.1.4.1.140.300.20.1.1.50
tuxTsrvrThreadStackSize	.1.3.6.1.4.1.140.300.20.1.1.60

## tuxTsrvrGrp

### Syntax

*DisplayString* (SIZE(1..30))

### Access

read-write

### Description

Logical name of the server group. Server group names cannot contain an asterisk (\*), comma, or colon.

**Note:** This object can be set only during row creation.

## tuxTsrvrId

### Syntax

INTEGER (1..30001)

## Access

read-write

## Description

Unique (within the server group) server identification number.

**Note:** This object can be set only during row creation.

## **tuxTsrvrName**

### Syntax

*DisplayString* (SIZE(1..78))

### Access

read-write

### Description

Name of the server executable file. The server identified by `tuxTsrvrName` runs on the machine(s) identified by the `tuxTgroupLMID` object for this server's server group. If a relative pathname is given, the search for the executable file is done first in `tuxTmachineAppDir`, then in `tuxTmachineTuxDir/bin`, then in `/bin` and `/usr/bin`, and then in `<path>`, where `<path>` is the value of the first `PATH=` line that appears in the machine environment file, if one exists. Note that the object value returned for an active server is always a full pathname.

## **tuxTsrvrGrpNo**

### Syntax

INTEGER (1..30000)

### Access

read-only

### Description

Group number associated with this server's group.

## tuxTsrvrState

### Syntax

```
INTEGER { active(1) | inactive(2) | migrating(3) | cleaning(4) |
restarting(5) | suspended(6) | partitioned(7) | dead(8) | invalid(10) }
```

### Access

read-write

### Description

The values for GET and SET operations are as follows:

```
GET: active(1) | inactive(2) | migrating(3) | cleaning(4) | restarting(5) |
suspended(6) | partitioned(7) | dead(8)
```

A GET operation retrieves configuration and run-time information for the selected tuxTsrvrTbl instance(s). The following states indicate the meaning of a tuxTsrvrState returned in response to a GET request. States not listed are not returned.

active(1)

tuxTsrvrTbl instance is defined and active. The active(1) state is not an indication of whether the server is idle or busy. An active server with a non-0 length tuxTsrvrCurService object should be interpreted as a busy server, that is, one that is processing a service request.

inactive(2)

tuxTsrvrTbl instance is defined and inactive.

migrating(3)

tuxTsrvrTbl instance is defined and currently in a state of migration to the server group's secondary logical machine. The secondary logical machine is the one listed in tuxTgroupLMID object that does not match the tuxTgroupCurLMID object.

cleaning(4)

tuxTsrvrTbl instance is defined and currently being cleaned up after by the system due to an abnormal death. Note that restartable servers can enter this state if they exceed tuxTsrvrMaxgen starts/restarts within their tuxTsrvrGrace period.

restarting(5)

tuxTsrvrTbl instance is defined and currently being restarted by the system due to an abnormal death.

suspended(6)

tuxTsrvrTbl instance is defined and currently suspended, pending shutdown.

partitioned(7)

`tuxTsrvrTbl` instance is defined and active; however, the machine where the server is running is currently partitioned from the `tuxTdomainMaster` site.

dead(8)

`tuxTsrvrTbl` instance is defined, identified as active in the bulletin board, but currently not running due to an abnormal death. This state exists only until the BBL local to the server notices the death and takes action (`restarting(5)` | `cleaning(4)`).

SET: {`active(1)` | `inactive(2)` | `dead(8)` | `invalid(10)`}

A SET operation updates configuration and run-time information for the selected `tuxTsrvrTbl` instance. The following states indicate the meaning of a `tuxTsrvrState` set in a SET request. States not listed cannot be set.

`active(1)`

Activate the `tuxTsrvrTbl` instance. State change is allowed only when the server is in the `inactive(2)` state. (Servers in the `migrating(3)` state must be restarted by setting the `tuxTgroupState` to `active(1)`.) Successful return leaves the object in the `active(1)` state.

`inactive(2)`

Deactivate the `tuxTsrvrTbl` instance. State change is allowed only when the server is in the `active(1)` state. Successful return leaves the object in the `inactive(2)` state.

`dead(8)`

Deactivate the `tuxTsrvrTbl` instance by sending the server a SIGTERM signal followed by a SIGKILL signal if the server is still running after 20 seconds. Note that by default, a SIGTERM signal causes the server to initiate orderly shutdown and the server becomes inactive even if it is restartable. If a server is processing a long running service or has chosen to disable the SIGTERM signal, then SIGKILL can be used and is treated by the system as an abnormal termination. State change is allowed only when the server is in the `active(1)` or `suspended(6)` state. Successful return leaves the object in the `inactive(2)`, `cleaning(4)`, or `restarting(5)` state.

`invalid(10)`

Delete `tuxTsrvrTbl` instance for application. State change is allowed only when the server is in the `inactive(2)` state. Successful return leaves the object in the `invalid(10)` state.

## **tuxTsrvrBaseSrvId**

### **Syntax**

INTEGER (1..30001)

## Access

read-only

## Description

Base server identifier. For servers with a `tuxTsrvrMax` object value of 1, this object is always the same as `tuxTsrvrId`. However, for servers with a `tuxTsrvrMax` value of greater than 1, this object indicates the base server identifier for the set of servers configured identically.

## tuxTsrvrCIOpt

### Syntax

*DisplayString*(SIZE(0..256))

### Access

read-write

### Description

Command line options to be passed to server when it is activated. See the `servopts(5)` manual page for details.

**Note:** Run-time modifications to this object do not affect a running server.

## tuxTsrvrEnvFile

### Syntax

*DisplayString*(SIZE(0..256)) (up to 78 bytes for Oracle Tuxedo 8.0 or earlier)

### Access

read-write

### Description

Server specific environment file. See `tuxTmachineEnvFile` for a complete discussion of how this file is used to modify the environment.

**Note:** Run-time modifications to this object do not affect a running server.

## tuxTsrvrGrace

### Syntax

INTEGER

### Access

read-write

### Description

The period of time, in seconds, over which the `tuxTsrvrMaxgen` object limit applies. The `tuxTsrvrGrace` value is meaningful only for restartable servers, that is, if the `tuxTsrvrRestart` object is set to `yes(1)`. When a restarting server would exceed the `tuxTsrvrMaxgen` limit but the `tuxTsrvrGrace` period has expired, the system resets the current generation (`tuxTsrvrGeneration`) to 1 and resets the initial boot time (`tuxTsrvrTimeStart`) to the current time. A value of 0 for this object indicates that a server should always be restarted.

Note that servers sharing a request queue (that is, equal values for `tuxTsrvrRqAddr`) should have equal values for this object. If they do not, then the first server activated establishes the run-time value associated with all servers on the queue.

**Note:** Run-time modifications to this object affect a running server and all other active servers with which it is sharing a request queue. However, only the selected server's configuration parameter is modified. Thus, the behavior of the application depends on the order of boot in subsequent activations unless the administrator ensures that all servers sharing a queue have the same value for this object.

## tuxTsrvrMaxgen

### Syntax

INTEGER (0..256)

### Access

read-write

### Description

Number of generations allowed for a restartable server (`tuxTsrvrRestart == yes(1)`) over the specified grace period (`tuxTsrvrGrace`). The initial activation of the server counts as one

generation and each restart also counts as one. Processing after the maximum generations is exceeded is discussed above with respect to `tuxTsrvrGrace`.

Note that servers sharing a request queue (that is, equal values for `tuxTsrvrRqAddr`) should have equal values for this object. If they do not, then the first server activated establishes the run-time value associated with all servers on the queue.

**Note:** Run-time modifications to this object affect a running server and all other active servers with which it is sharing a request queue. However, only the selected server's configuration parameter is modified. Thus, the behavior of the application depends on the order of boot in subsequent activations unless the administrator ensures that all servers sharing a queue have the same value for this object.

## tuxTsrvrMax

### Syntax

INTEGER (1..1001)

### Access

read-write

### Description

Maximum number of occurrences of the server to be booted. Initially, `tmboot(1)` boots `tuxTsrvrMin` objects of the server, and additional objects can be started individually (by starting a particular server id) or through automatic spawning (conversational servers only). Run-time modifications to this object affect all running servers in the set of identically configured servers (see `tuxTsrvrBaseSrvId` above) as well as the configuration definition of the server.

## tuxTsrvrMin

### Syntax

INTEGER (1..1001)

### Access

read-write

## Description

Minimum number of occurrences of the server to be booted by `tmboot(1)`. If a `tuxTsrvrRqAddr` is specified and `tuxTsrvrMin` is greater than 1, then the servers form an MSSQ set. The server identifiers for the servers are `tuxTsrvrId` up to `tuxTsrvrId + tuxTsrvrMax - 1`. All occurrences of the server have the same sequence number, as well as any other server parameters.

**Note:** Run-time modifications to this object do not affect a running server.

## tuxTsrvrRcmd

### Syntax

*DisplayString* (SIZE(0..256)) (up to 78 bytes for Oracle Tuxedo 8.0 or earlier)

### Access

read-write

## Description

Application-specified command to be executed in parallel with the system restart of an application server. This command must be an executable file.

Note that servers sharing a request queue (that is, equal values for `tuxTsrvrRqAddr`) should have equal values for this object. If they do not, then the first server activated establishes the run-time value associated with all servers on the queue.

**Note:** Run-time modifications to this object affect a running server and all other active servers with which it shares a request queue. However, only the selected server's configuration parameter is modified. Thus, the behavior of the application depends on the order of boot in subsequent activations unless the administrator ensures that all servers sharing a queue have the same value for this object.

## tuxTsrvrRestart

### Syntax

INTEGER { yes(1) | no(2) }

### Access

read-write

## Description

Restartable *yes* (1) or non-restartable *no* (2) server. If server migration is specified for this server group (*tuxTdomainOptions* = *migrate* (2) and *tuxTgroupLMID* with alternate site), this object must be set to *yes* (1).

Note that servers sharing a request queue (that is, equal values for *tuxTsrvrRqAddr*) should have equal values for this object. If they do not, the first server activated establishes the run-time value associated with all servers on the queue.

**Note:** Run-time modifications to this object affect a running server and all other active servers with which it shares a request queue. However, only the selected server's configuration parameter is modified. Thus, the behavior of the application depends on the order of boot in subsequent activations unless the administrator ensures that all servers sharing a queue have the same value for this object.

## tuxTsrvrSequence

### Syntax

INTEGER (1..10000)

### Access

read-write

### Description

Specifies when this server should be booted (*tmboot*(1)) or shutdown (*tmshutdown*(1)) relative to other servers. If two servers are given the same sequence number, it is possible for *tmboot*(1) to boot them in parallel and for *tmshutdown*(1) to shut them down in parallel. *tuxTsrvrTbl* instances added without a *tuxTsrvrSequence* object specified or with an invalid value have one generated for them that is 10,000 or more and is higher than any other automatically selected default value. Servers are booted by *tmboot*(1) in increasing order of sequence number and shutdown by *tmshutdown*(1) in decreasing order. Run-time modifications to this object affect only *tmboot*(1) and *tmshutdown*(1) and affect the order in which running servers can be shutdown by a subsequent invocation of *tmshutdown*(1).

## tuxTsrvrSystemAccess

### Syntax

INTEGER { *fastpath*(1) | *protected*(2) }

## Access

read-write

## Description

Mode used by Tuxedo system libraries within this server process to gain access to Tuxedo system's internal tables. See `tuxTdomainSystemAccess` for a complete discussion of this object.

**Note:** Run-time modifications to this object do not affect a running server.

## **tuxTsrvrConv**

### Syntax

```
INTEGER { yes(1) | no(2) }
```

### Access

read-write

### Description

Conversational server `yes(1)` or request/response server `no(2)`.

## **tuxTsrvrReplyQ**

### Syntax

```
INTEGER { yes(1) | no(2) }
```

### Access

read-write

### Description

Specifies whether to allocate a separate reply queue for the server (`tuxTsrvrReplyQ == yes(1)`). MSSQ servers that expect to receive replies should set this object to `yes(1)`.

## tuxTsrvrRpPerm

### Syntax

*DisplayString*(SIZE(4))

### Access

read-write

### Description

UNIX system permissions for the server's reply queue. If a separate reply queue is not allocated (`tuxTsrvrReplyQ == no(2)`), this object is ignored. The `tuxTsrvrRpPerm` value is a string representation of octal numbers starting with a leading 0 0001 through 0777.

**Note:** This object is a UNIX-system-specific object that cannot be returned if the platform on which the application is being run is not UNIX-based.

## tuxTsrvrRqAddr

### Syntax

*DisplayString*(SIZE(0..30))

### Access

read-write

### Description

Symbolic address of the request queue for the server. Specifying the same `tuxTsrvrRqAddr` object value for more than one server is the way multiple server, single queue (MSSQ) sets are defined. Servers with the same `tuxTsrvrRqAddr` object value must be in the same server group.

## tuxTsrvrRqPerm

### Syntax

*DisplayString*(SIZE(4))

### Access

read-write

## Description

UNIX system permissions for the server's request queue. The `tuxTsrvrRgPerm` value is a string representation of octal numbers starting with a leading 0 0001 through 0777.

**Note:** This object is a UNIX-system-specific object that cannot be returned if the platform on which the application is being run is not UNIX-based.

## tuxTsrvrGeneration

### Syntax

INTEGER (1..32768)

### Access

read-only

## Description

Generation of the server. When a server is initially booted via `tmboot(1)` or activated through the SNMP agent, its generation is set to 1. Each time the server dies abnormally and is restarted, its generation is incremented. Note that when `tuxTsrvrMaxgen` is exceeded and `tuxTsrvrGrace` has expired, the server is restarted with the generation reset to 1.

## tuxTsrvrPid

### Syntax

INTEGER

### Access

read-only

## Description

UNIX system process identifier for the server. Note that this value cannot be a unique value since servers can be located on different machines, allowing for duplication of process identifiers.

**Note:** This object is a UNIX-system-specific object that cannot be returned if the platform on which the application is being run is not UNIX-based.

## tuxTsrvrRpid

### Syntax

INTEGER

### Access

read-only

### Description

UNIX system message queue identifier for the server's reply queue. If a separate reply queue is not allocated (`tuxTsrvrReplyQ == no(2)`), the `tuxTsrvrRqid` value is the same as the `tuxTsrvrRqId` value.

**Note:** This object is a UNIX-system-specific object that cannot be returned if the platform on which the application is being run is not UNIX-based.

## tuxTsrvrRqid

### Syntax

INTEGER

### Access

read-only

### Description

UNIX system message queue identifier for the server's request queue. If a separate reply queue is not allocated (`tuxTsrvrReplyQ == no(2)`), the `tuxTsrvrRpId` value is the same as the `tuxTsrvrRpid` value.

**Note:** This object is a UNIX-system-specific object that cannot be returned if the platform on which the application is being run is not UNIX-based.

## tuxTsrvrTimeRestart

### Syntax

INTEGER

## Access

read-only

## Description

Time, in seconds, since 00:00:00 UTC, January 1, 1970, as returned by the `time(2)` system call on local host, when the server was last started or restarted.

## **tuxTsrvrTimeStart**

### Syntax

INTEGER

## Access

read-only

## Description

Time, in seconds, since 00:00:00 UTC, January 1, 1970, as returned by the `time(2)` system call on local host, when the server was first started. Restarts of the server do not reset this value; however, if `tuxTsrvrMaxgen` is exceeded and `tuxTsrvrGrace` is expired, this object is reset to the time of the restart.

## **tuxTsrvrMinDispatchThreads**

### Syntax

INTEGER (1..999)

## Access

read-only

## Description

Specifies the number of server dispatch threads started on the initial server boot. This object is effective only if the server has been built with the `buildserver -t` command.

The separate dispatcher thread that is used when the value of `tuxTsrvrMaxDispatchThreads` is greater than one is not counted as part of the `tuxTsrvrMinDispatchThreads` value. The value of `tuxTsrvrMinDispatchThreads` must be less than the value of `tuxTsrvrMaxDispatchThreads`. If this object is not specified, the default is 0.

## tuxTsrvrMaxDispatchThreads

### Syntax

INTEGER (0..999)

### Access

read-only

### Description

Specifies the maximum number of concurrently dispatched threads that each server process can spawn. This object is effective only if the server has been built with the `buildserver -t` command.

If `tuxTsrvrMaxDispatchThreads` is greater than one, a separate dispatcher thread is used and does not count against this limit. The value of `tuxTsrvrMinDispatchThreads` must be less than the value of `tuxTsrvrMaxDispatchThreads`. If this object is not specified, the default is 1.

## tuxTsrvrThreadStackSize

### Syntax

INTEGER (0..2147483647)

### Access

read-write

### Description

If this object is not specified or if the value specified is 0, the operating system default is used. This option affects the server only when a value greater than 1 is specified for `tuxTsrvrMaxDispatchThreads`.

## tuxTsrvrTblExt

The `tuxTsrvrTblExt` group is an extension of `tuxTsrvrTbl`. Objects in this table are only accessible through a Tuxedo SNMP agent installed on the local machine.

<b>Object Name</b>	<b>Object ID</b>
tuxTsrvrIdExt	.1.3.6.1.4.1.140.300.20.2.1.1
tuxTsrvrGrpNoExt	.1.3.6.1.4.1.140.300.20.2.1.2
tuxTsrvrNumConv	.1.3.6.1.4.1.140.300.20.2.1.3
tuxTsrvrNumDeque	.1.3.6.1.4.1.140.300.20.2.1.4
tuxTsrvrNumEnque	.1.3.6.1.4.1.140.300.20.2.1.5
tuxTsrvrNumPost	.1.3.6.1.4.1.140.300.20.2.1.6
tuxTsrvrNumReq	.1.3.6.1.4.1.140.300.20.2.1.7
tuxTsrvrNumSubscribe	.1.3.6.1.4.1.140.300.20.2.1.8
tuxTsrvrNumTran	.1.3.6.1.4.1.140.300.20.2.1.9
tuxTsrvrTranAbt	.1.3.6.1.4.1.140.300.20.2.1.10
tuxTsrvrTranCmt	.1.3.6.1.4.1.140.300.20.2.1.11
tuxTsrvrTotReqC	.1.3.6.1.4.1.140.300.20.2.1.12
tuxTsrvrTotWorkL	.1.3.6.1.4.1.140.300.20.2.1.13
tuxTsrvrClgLmid	.1.3.6.1.4.1.140.300.20.2.1.14
tuxTsrvrClgLpid	.1.3.6.1.4.1.140.300.20.2.1.15
tuxTsrvrClgLreply	.1.3.6.1.4.1.140.300.20.2.1.16
tuxTsrvrCmtRet	.1.3.6.1.4.1.140.300.20.2.1.17
tuxTsrvrCurConv	.1.3.6.1.4.1.140.300.20.2.1.18
tuxTsrvrCurReq	.1.3.6.1.4.1.140.300.20.2.1.19
tuxTsrvrCurService	.1.3.6.1.4.1.140.300.20.2.1.20
tuxTsrvrCurTime	.1.3.6.1.4.1.140.300.20.2.1.21
tuxTsrvrLastGrp	.1.3.6.1.4.1.140.300.20.2.1.22

<b>Object Name</b>	<b>Object ID</b>
tuxTsrvrSvcTimeOut	.1.3.6.1.4.1.140.300.20.2.1.23
tuxTsrvrTimeLeft	.1.3.6.1.4.1.140.300.20.2.1.24
tuxTsrvrTranLev	.1.3.6.1.4.1.140.300.20.2.1.25
tuxTsrvrStateExt	.1.3.6.1.4.1.140.300.20.2.1.26
tuxTsrvrGrpExt	.1.3.6.1.4.1.140.300.20.2.1.27
tuxSrvrCurObjsExt (Tuxedo 8.0 or later)	.1.3.6.1.4.1.140.300.20.2.1.32
tuxSrvrCurInterfaceExt (Tuxedo 8.0 or later)	.1.3.6.1.4.1.140.300.20.2.1.37
tuxTsrvrCurDispatchThreads	.1.3.6.1.4.1.140.300.20.2.1.100
tuxTsrvrHwDispatchThreads	.1.3.6.1.4.1.140.300.20.2.1.110
tuxTsrvrNumDispatchThreads	.1.3.6.1.4.1.140.300.20.2.1.120

## tuxTsrvrIdExt

### Syntax

INTEGER (1..30001)

### Access

read-only

### Description

Unique (within the server group) server identification number.

## tuxTsrvrGrpNoExt

### Syntax

INTEGER (1..30000)

### Access

read-only

## Description

Group number associated with this server's group.

## **tuxTsrvrNumConv**

### Syntax

INTEGER

### Access

read-only

## Description

Number of conversations initiated by this server through `tpconnect(3)`.

## **tuxTsrvrNumDeque**

### Syntax

INTEGER

### Access

read-only

## Description

Number of dequeue operations initiated by this server through `tpdequeue(3)`.

## **tuxTsrvrNumEnque**

### Syntax

INTEGER

### Access

read-only

## Description

Number of enqueue operations initiated by this server through `tpenqueue(3)`.

## **tuxTsrvrNumPost**

### Syntax

INTEGER

### Access

read-only

### Description

Number of postings initiated by this server through `tppost(3)`.

## **tuxTsrvrNumReq**

### Syntax

INTEGER

### Access

read-only

### Description

Number of requests made by this server through `tpcall(3)` or `tpacall(3)`.

## **tuxTsrvrNumSubscribe**

### Syntax

INTEGER

### Access

read-only

### Description

Number of subscriptions made by this server through `tpsubscribe(3)`.

## **tuxTsrvrNumTran**

### Syntax

INTEGER

### Access

read-only

### Description

Number of transactions begun by this server since its last (re)start.

## **tuxTsrvrTranAbt**

### Syntax

INTEGER

### Access

read-only

### Description

Number of transactions aborted by this server since its last (re)start.

## **tuxTsrvrTranCmt**

### Syntax

INTEGER

### Access

read-only

### Description

Number of transactions committed by this server since its last (re)start.

## tuxTsrvrTotReqC

### Syntax

INTEGER

### Access

read-only

### Description

Total number of requests completed by this server. For conversational servers (`tuxTsrvrConv == yes(1)`), the `tuxTsrvrTotReqC` value indicates the number of completed incoming conversations. The `tuxTsrvrTotReqC` value is a run-time value that is kept across server restart but is lost at server shutdown.

## tuxTsrvrTotWorkL

### Syntax

INTEGER

### Access

read-only

### Description

Total workload completed by this server. For conversational servers (`tuxTsrvrConv == yes(1)`), the `tuxTsrvrTotWorkL` value indicates the workload of completed incoming conversations. The `tuxTsrvrTotWorkL` value is a run-time value that is kept across server restart but is lost at server shutdown.

## tuxTsrvrClitLmid

### Syntax

*DisplayString*(SIZE(1..30))

### Access

read-only

## Description

Logical machine for the initiating client or server. The initiating client or server is the process that made the service request on which the server is currently working.

## tuxTsrvrCltpid

### Syntax

```
INTEGER
```

### Access

read-only

### Description

UNIX system process identifier for the initiating client or server.

**Note:** This object is a UNIX-system-specific object that cannot be returned if the platform on which the application is being run is not UNIX-based.

## tuxTsrvrCltrply

### Syntax

```
INTEGER { yes(1) | no(2) | null(3) }
```

### Access

read-only

### Description

yes(1)

The initiating client or server expects a reply.

no(2)

The initiating client or server does not expect a reply.

## tuxTsrvrCmtret

### Syntax

```
INTEGER { complete(1) | logged(2) }
```

## Access

read-only

## Description

The setting of the TP\_COMMIT\_CONTROL characteristic for this server. For details on this characteristic, see the description of the Tuxedo system ATMI function `tpscmt(3)`.

## **tuxTsrvrCurConv**

### Syntax

INTEGER

### Access

read-only

### Description

Number of conversations initiated by this server through `tpconnect(3)` that are still active.

## **tuxTsrvrCurReq**

### Syntax

INTEGER

### Access

read-only

### Description

Number of requests initiated by this server through `tpcall(3)` or `tpacall(3)` that are still active.

## **tuxTsrvrCurService**

### Syntax

*DisplayString(SIZE(1..127))*

## Access

read-only

## Description

Service name, if any, on which the server is currently working.

## **tuxTsrvrCurTime**

### Syntax

INTEGER

## Access

read-only

## Description

Current time, in seconds, since 00:00:00 UTC, January 1, 1970, as on the local host. This object can be used to compute elapsed time from the `tuxTsrvrTimeStart` and `tuxTsrvrTimeRestart` object values.

## **tuxTsrvrLastGrp**

### Syntax

INTEGER (1..30000)

## Access

read-only

## Description

Server group number (`tuxTgroupNo`) of the last service request made or conversation initiated from this server outward.

## **tuxTsrvrSvcTimeOut**

### Syntax

INTEGER

## Access

read-only

## Description

Time left in seconds, if any, for this server to process the current service request. A value of 0 for an active service indicates that no timeout processing is being done. See `tuxTsvcTimeOut` for more information.

## **tuxTsrvrTimeLeft**

### Syntax

INTEGER

### Access

read-only

### Description

Time left, in seconds, for this server to receive the reply for which it is currently waiting before it times out. This timeout can be a transactional timeout or a blocking timeout.

## **tuxTsrvrTranLev**

### Syntax

INTEGER

### Access

read-only

### Description

Current transaction level for this server. 0 indicates that the server is not currently involved in a transaction.

## **tuxTsrvrStateExt**

### Syntax

```
INTEGER { active(1) | inactive(2) | migrating(3) | cleaning(4) |  
restarting(5) | suspended(6) | partitioned(7) | dead(8) }
```

### Access

read-only

### Description

Refer to description of `tuxTsrvrState` for details.

## **tuxTsrvrGrpExt**

### Syntax

```
DisplayString
```

### Access

read-only

### Description

Name of group to which this server belongs. This object is included for readability purposes only.

## **tuxSrvrCurObjsExt (Tuxedo 8.0 or later)**

### Syntax

```
INTEGER
```

### Access

read-only

### Description

The number of entries in use in the bulletin board object table for this server.

## **tuxSrvrCurInterfaceExt (Tuxedo 8.0 or later)**

### Syntax

*DisplayString*(SIZE(1..128))

### Access

read-only

### Description

The interface name of the interface currently active in this server.

## **tuxTsrvtCurDispatchThreads**

### Syntax

INTEGER

### Access

read-only

### Description

Current number of active service dispatch threads for this server.

## **tuxTsrvtHwDispatchThreads**

### Syntax

INTEGER

### Access

read-only

### Description

Highest number of active service dispatch threads created for this server since its last restart. This number can differ from the number of service calls, because an administrator can specify parameters that control the caching of idle service threads.

## tuxTsrvrNumDispatchThreads

### Syntax

INTEGER

### Access

read-only

### Description

Total number of active service dispatch threads created for this server since its last restart.

## tuxTsvcTbl

The `tuxTsvcTbl` group contains objects that represent configuration characteristics of services within an application. The object values identify and characterize configured services. A `tuxTsvcTbl` object provides activation time configuration objects for services not specifically configured as part of the `tuxTsvcGrp` group.

The index into this table is `tuxTsvcName`. Objects in this group are only accessible through a Tuxedo SNMP agent installed on the local machine. To create a new row in the table, it is necessary to issue a `SET` request for a non-existing row in the table.

Object Name	Object ID
<code>tuxTsvcName</code>	.1.3.6.1.4.1.140.300.10.1.1.1
<code>tuxTsvcType</code>	.1.3.6.1.4.1.140.300.10.1.1.2
<code>tuxTsvcState</code>	.1.3.6.1.4.1.140.300.10.1.1.3
<code>tuxTsvcAutoTran</code>	.1.3.6.1.4.1.140.300.10.1.1.4
<code>tuxTsvcLoad</code>	.1.3.6.1.4.1.140.300.10.1.1.5
<code>tuxTsvcPrio</code>	.1.3.6.1.4.1.140.300.10.1.1.6
<code>tuxTsvcTimeOut</code>	.1.3.6.1.4.1.140.300.10.1.1.7
<code>tuxTsvcTranTime</code>	.1.3.6.1.4.1.140.300.10.1.1.8
<code>tuxTsvcBufType</code>	.1.3.6.1.4.1.140.300.10.1.1.9

Object Name	Object ID
tuxTsvcRoutingName	.1.3.6.1.4.1.140.300.10.1.1.10
tuxTsvcEncryptionRequired	.1.3.6.1.4.1.140.300.10.1.1.20
tuxTsvcSignatureRequired	.1.3.6.1.4.1.140.300.10.1.1.30

## tuxTsvcName

### Syntax

*DisplayString*(SIZE(1..127))

### Access

read-write

### Description

Service name.

**Note:** This object can be set only during row creation.

## tuxTsvcType

### Syntax

INTEGER { app(1) | callable(2) | system(3) | unknown(4) }

### Access

read-only

### Description

Type of service.

app(1)

Indicates an application-defined service name.

callable(2)

Indicates a system-provided callable service.

`system(3)`

Indicates a system-provided and system-callable service. `system(3)` services are not available to application clients and servers for direct access.

## tuxTsvcState

### Syntax

```
INTEGER { active(1) | inactive(2) | invalid(3) }
```

### Access

read-write

### Description

The values for GET and SET operations are as follows:

GET: { active(1) | inactive(2) }

A GET operation retrieves configuration information for the selected `tuxTsvcTbl` instance(s). The following states indicate the meaning of a `tuxTsvcState` returned in response to a GET request. States not listed are not returned.

`active(1)`

`tuxTsvcTbl` instance is defined and at least one `tuxTsvcGrp` object with a matching `tuxTsvcName` value is active.

`inactive(2)`

`tuxTsvcTbl` instance is defined and no `tuxTsvcGrp` object with a matching `tuxTsvcName` value is active.

SET: `invalid(3)`

A SET operation updates configuration information for the selected `tuxTsvcTbl` instance. The following state indicates the meaning of a `tuxTsvcState` set in a SET request. States not listed cannot be set.

`invalid(3)`

Delete `tuxTsvcTbl` instance for application. State change is allowed only when the service is in the `inactive(2)` state. Successful return leaves the object in the `invalid(3)` state.

## tuxTsvcAutoTran

### Syntax

INTEGER { yes(1) | no(2) }

### Access

read-write

### Description

Automatically begin a transaction.

yes(1)

If the request is not already in transaction mode when a service request message is received for this service, automatically begin a transaction.

no(2)

Do not automatically begin a transaction

**Note:** Run-time updates to this object are not reflected in active `tuxTsvcGrp` objects.

## tuxTsvcLoad

### Syntax

INTEGER (1..32768)

### Access

read-write

### Description

This `tuxTsvcTbl` object imposes the indicated load on the system. Service loads are used for load balancing purposes, that is, queues with higher enqueued workloads are less likely to be chosen for a new request. Service loads have meaning only if the `tuxTdomainLoadBalance` is set to `yes(1)`.

**Note:** Run-time updates to this object are not reflected in active `tuxTsvcGrp` objects.

## tuxTsvcPrio

### Syntax

INTEGER (1..100)

### Access

read-write

### Description

This `tuxTsvcTbl` object has the indicated dequeuing priority. If multiple service requests are waiting on a queue for servicing, the higher priority requests are serviced first.

**Note:** Run-time updates to this object are not reflected in active `tuxTsvcGrp` objects.

## tuxTsvcTimeOut

### Syntax

INTEGER

### Access

read-write

### Description

Time limit (in seconds) for processing requests for this service name. Servers processing service requests for this service are abortively terminated (`kill -9`) if they exceed the specified time limit in processing the request. A value of 0 for this object indicates that the service should not be abortively terminated.

**Note:** Run-time updates to this object are not reflected in active `tuxTsvcGrp` objects.

## tuxTsvcTranTime

### Syntax

INTEGER

### Access

read-write

## Description

Transaction timeout value (in seconds) for transactions automatically started for this `tuxTsvcTbl` object. Transactions are started automatically when a request not in transaction mode is received and the `tuxTsvcAutoTran` object value for the service is `yes(1)`.

**Note:** Run-time updates to this object are not reflected in active `tuxTsvcGrp` objects.

## tuxTsvcBufType

### Syntax

*DisplayString*(SIZE(1..256))

### Access

read-write

### Description

`type1[:subtype1[,subtype2 . . . ]];type2[:subtype3[. . . ]]` . . .

List of types and subtypes of data buffers accepted by this service. A maximum of 32 type/subtype combinations are allowed. Types of data buffers provided with Tuxedo system are FML (for FML buffers), VIEW, X\_C\_TYPE, or X\_COMMON (for FMLviews), STRING (for NULL terminated character arrays), and CARRAY or X\_OCTET (for a character array that is neither encoded nor decoded during transmission). Of these types, only VIEW, X\_C\_TYPE, and X\_COMMON have subtypes. A VIEW subtype gives the name of the particular VIEW expected by the service. Application types and subtypes can also be added (see `tuxtypes(5)`). For a buffer type that has subtypes, "\*" can be specified for the subtype to indicate that the service accepts all subtypes for the associated buffer type.

A single service can only interpret a fixed number of buffer types, namely those found in its buffer type switch (see `tuxtypes(5)`). If the `tuxTsvcBufType` value is set to ALL, that service accepts all buffer types found in its buffer type switch.

A type name can be 8 characters or less in length and a subtype name can be 16 characters or less in length. Note that type and subtype names should not contain semicolon, colon, comma, or asterisk characters.

**Note:** The `tuxTsvcBufType` value represents the buffer types that must be supported by each and every instance of an application service with this service name. Since this object value is processed at service activation time, updates to this object are allowed only when there are no active `tuxTsvcGrp` objects with matching service names.

## **tuxTsvcRoutingName**

### Syntax

*DisplayString*(SIZE(0..15))

### Access

read-write

### Description

This `tuxTsvcTbl` object has the indicated routing criteria name. Active updates to this object are reflected in all associated `tuxTsvcGrp` objects.

## **tuxTsvcEncryptionRequired**

### Syntax

INTEGER { yes(1) | no(2) }

### Access

read-write

### Description

If set to `yes`, every application service in this group requires an encrypted input message buffer.

## **tuxTsvcSignatureRequired**

### Syntax

INTEGER { yes(1) | no(2) }

### Access

read-write

### Description

If set to `yes`, every application service in this group requires a valid digital signature on its input message buffer.

## tuxTsvcGrp

The `tuxTsvcGrp` group contains objects that represent configuration and run-time characteristics of services/groups within an application. The object values identify and characterize configured services/groups as well as provide run-time tracking of statistics and resources associated with each object.

Both `tuxTsvcTbl` and `tuxTsvcGrp` define activation time object values for service names within the application. When a new service is activated (advertised), either due to initial activation of a server or due to a call to `tpadvertise(3)`, the following hierarchy exists for determining the object values to be used at service startup time.

1. If a matching configured `tuxTsvcGrp` entry exists (matching service name and server group), the objects defined in that object are used to initially configure the advertised service.
2. Otherwise, if a matching configured `tuxTsvcTbl` entry exists (matching service name), the objects defined in that object are used to initially configure the advertised service.
3. Otherwise, if any configured `tuxTsvcGrp` entries are found with matching service name value, the first one found is used to initially configure the advertised service.
4. If none of the preceding cases is used, the system defaults for service objects are used to initially configure the advertised service.

Objects in this group are only accessible through a Tuxedo SNMP agent installed on the local machine.

To create a new row in the table, it is necessary to issue a `SET` request that specifies at least `tuxTsvcGrpName`. The combination of values specified for `tuxTsvcGrpName` and `tuxTsvcGrpSvcName` in the `SET` request should not correspond to an existing row. If the value of `tuxTsvcSrvrId` is zero in the `SET` request, the service entry is configured but not activated (advertised). If `tuxTsvcSrvrId` is not set to zero, the service is activated using the value of `tuxTsvcSrvrId` to identify the server instance.

Object Name	Object ID
<code>tuxTsvcGrpSvcName</code>	.1.3.6.1.4.1.140.300.10.2.1.1
<code>tuxTsvcGrpName</code>	.1.3.6.1.4.1.140.300.10.2.1.2
<code>tuxTsvcGrpNo</code>	.1.3.6.1.4.1.140.300.10.2.1.3
<code>tuxTsvcGrpState</code>	.1.3.6.1.4.1.140.300.10.2.1.4

<b>Object Name</b>	<b>Object ID</b>
tuxTsvcGrpAutoTran	.1.3.6.1.4.1.140.300.10.2.1.5
tuxTsvcGrpLoad	.1.3.6.1.4.1.140.300.10.2.1.6
tuxTsvcGrpPrio	.1.3.6.1.4.1.140.300.10.2.1.7
tuxTsvcGrpSvcTimeOut	.1.3.6.1.4.1.140.300.10.2.1.8
tuxTsvcGrpTranTime	.1.3.6.1.4.1.140.300.10.2.1.9
tuxTsvcSrvrLmid	.1.3.6.1.4.1.140.300.10.2.1.10
tuxTsvcSrvrRqAddr	.1.3.6.1.4.1.140.300.10.2.1.11
tuxTsvcSrvrId	.1.3.6.1.4.1.140.300.10.2.1.12
tuxTsvcrName	.1.3.6.1.4.1.140.300.10.2.1.13
tuxTsvcSrvrNcompleted	.1.3.6.1.4.1.140.300.10.2.1.14
tuxTsvcSrvrNqueued	.1.3.6.1.4.1.140.300.10.2.1.15

## **tuxTsvcGrpSvcName**

### **Syntax**

*DisplayString* (SIZE(1..15))

### **Access**

read-only

### **Description**

Service name.

## **tuxTsvcGrpName**

### **Syntax**

*DisplayString* (SIZE(1..30))

## Access

read-only

## Description

Server group name. Server group names cannot contain an asterisk.

## **tuxTsvcGrpNo**

### Syntax

```
INTEGER (1..29999)
```

### Access

read-write

### Description

Server group number.

## **tuxTsvcGrpState**

### Syntax

```
INTEGER { active(1) | inactive(2) | invalid(3) }
```

### Access

read-write

### Description

The values for GET and SET operations are as follows:

GET: active(1) | inactive(2)

A GET operation retrieves configuration information for the selected tuxTsvcGrpState instance(s). The following states indicate the meaning of a tuxTsvcGrpState returned in response to a GET request. States not listed are not returned.

active(1)

At least one instance is active, suspended, or partitioned.

inactive(2)

tuxTsvcGrp instance defined and inactive.

SET: `invalid(3)`

A SET operation removes the corresponding `tuxTsvcGrp` instance. When a `tuxTsvcGrp` instance is deleted it also removes the associated `tuxTsvcSrvr` instances that correspond to server instances that are a part of the group advertising this service. This transition is permissible only in `inactive(2)` state.

## **tuxTsvcGrpAutoTran**

### Syntax

INTEGER { `yes(1)` | `no(2)` }

### Access

read-write

### Description

Automatically begin a transaction (`yes(1)`) when a service request message is received for this service if the request is not already in transaction mode.

## **tuxTsvcGrpLoad**

### Syntax

INTEGER (1..32767)

### Access

read-write

### Description

This `tuxTsvcGrp` instance imposes the indicated load on the system. Service loads are used for load balancing purposes, that is, queues with higher enqueued workloads are less likely to be chosen for a new request.

## **tuxTsvcGrpPrio**

### Syntax

INTEGER (1..100)

## Access

read-write

## Description

This `tuxTsvcGrp` object has the indicated dequeuing priority. If multiple service requests are waiting on a queue for servicing, the higher priority requests are serviced first.

## **tuxTsvcGrpSvcTimeOut**

### Syntax

INTEGER

### Access

read-write

### Description

Time limit (in seconds) for processing requests for this service name. Servers processing service requests for this service are abortively terminated (`kill -9`) if they exceed the specified time limit in processing the request. A value of 0 for this object indicates that the service should not be abortively terminated.

## **tuxTsvcGrpTranTime**

### Syntax

INTEGER

### Access

read-write

### Description

Transaction timeout value (in seconds) for transactions automatically started for this `tuxTsvcGrp` instance. Transactions are started automatically when a request not in transaction mode is received and the `tuxTsvcGrpAutoTran` object value for the service is `yes (1)`.

## **tuxTsvcSrvrLmid**

### Syntax

*DisplayString* (SIZE(1..30))

### Access

read-only

### Description

Current logical machine on which an active server that offers this service is running.

## **tuxTsvcSrvrRqAddr**

### Syntax

*DisplayString* (SIZE(1..30))

### Access

read-only

### Description

Symbolic address of the request queue for an active server that offers this service. See `tuxTsrvrRqAddr` for more information on this object.

## **tuxTsvcSrvrId**

### Syntax

INTEGER (1..30000)

### Access

read-write

### Description

Server ID of which the service is a part. The user can also set the value of this object to activate (advertise) one or more `tuxTsvcGrp` instances. The value provided to set this object is used to activate another instance of `tuxTsvcGrp`.

## tuxTsvcrName

### Syntax

*DisplayString*(SIZE(1..127))

### Access

read-write

### Description

Function name within the associated server assigned to process requests for this service. When this object is specified, the `tuxTsvcGrp` instance is activated (advertised). The user needs to specify the server ID of the corresponding server instance (`tuxTsvcSrvrId`) in the SNMP index. This object can be updated only during row creation.

## tuxTsvcSrvrNcompleted

### Syntax

INTEGER

### Access

read-only

### Description

Number of service requests completed with respect to the retrieved active or suspended instance since it was activated (advertised).

**Note:** The `tuxTsvcSrvrNcompleted` value is returned only when `tuxTdomainLoadBalance` is equal to `yes` (1).

## tuxTsvcSrvrNqueued

### Syntax

INTEGER (0..32767)

### Access

read-only

## Description

Number of requests currently enqueued to this service. The `tuxTsvcSrvrNqueued` value is incremented at enqueue time and decremented when the server dequeues the request.

**Note:** The `tuxTsvcSrvrNqueued` value is returned only when `tuxTdomainModel` is set to `single-machine(1)` and `tuxTdomainLoadBalance` is set to `yes(1)`.

## tuxTlistenTbl

The `tuxTlistenTbl` group contains objects that represent run-time characteristics of Tuxedo listener processes for a distributed application.

Object Name	Object ID
<code>tuxTlistenLmid</code>	<code>.1.3.6.1.4.1.140.300.21.1.1.1</code>
<code>tuxTlistenState</code>	<code>.1.3.6.1.4.1.140.300.21.1.1.2</code>

## tuxTlistenLmid

### Syntax

*DisplayString*(SIZE(1..30))

### Access

read-only

### Description

Logical machine identifier.

## tuxTlistenState

### Syntax

INTEGER { inactive(2) | active(1) }

### Access

read-only

## Description

The values for GET and SET operations are as follows:

GET: {active(1) | inactive(2) }

A GET operation retrieves run-time information for the selected tuxTlistenTbl instance(s). The following states indicate the meaning of a tuxTlistenState returned in response to a GET request. States not listed are not returned.

active(1)

tuxTlistenTbl instance active.

inactive(2)

tuxTlistenTbl instance not active.

## tuxTranTbl

The tuxTranTbl group contains objects that represent run-time characteristics of active transactions within the application. The following objects comprise the index for rows in this table: tuxTranIndx1, tuxTranIndx2, tuxTranIndx3, tuxTranIndx4, tuxTranIndx5. Objects in this table are accessible only through a Tuxedo SNMP agent running on the local machine.

Object Name	Object ID
tuxTranCoordLmid	.1.3.6.1.4.1.140.300.23.1.1.1
tuxTpTranId	.1.3.6.1.4.1.140.300.23.1.1.2
tuxTranXid	.1.3.6.1.4.1.140.300.23.1.1.3
tuxTranIndx1	.1.3.6.1.4.1.140.300.23.1.1.4
tuxTranIndx2	.1.3.6.1.4.1.140.300.23.1.1.5
tuxTranIndx3	.1.3.6.1.4.1.140.300.23.1.1.6
tuxTranIndx4	.1.3.6.1.4.1.140.300.23.1.1.7
tuxTranIndx5	.1.3.6.1.4.1.140.300.23.1.1.8
tuxTranState	.1.3.6.1.4.1.140.300.23.1.1.9
tuxTranTimeOut	.1.3.6.1.4.1.140.300.23.1.1.10

Object Name	Object ID
tuxTranGrpCnt	.1.3.6.1.4.1.140.300.23.1.1.11
tuxTranGrpIndex	.1.3.6.1.4.1.140.300.23.1.1.12
tuxTranGrpNo	.1.3.6.1.4.1.140.300.23.1.1.13
tuxTranGstate	.1.3.6.1.4.1.140.300.23.1.1.14

## tuxTranCoordLmid

### Syntax

*DisplayString*(SIZE(1..30))

### Access

read-only

### Description

Logical machine identifier of the server group responsible for coordinating the transaction.

## tuxTpTranId

### Syntax

*DisplayString*(SIZE(2..78))

### Access

read-only

### Description

Transaction identifier as returned from `tpsuspend(3)` mapped to a string representation. The data in this field should not be interpreted directly by the user except for equality comparison.

## tuxTranXid

### Syntax

*DisplayString*(SIZE(2..78))

## Access

read-only

## Description

Transaction identifier as returned from `tx_info(3)` mapped to a string representation. The data in this field should not be interpreted directly by the user except for equality comparison.

## **tuxTranIdx1**

### Syntax

INTEGER

### Access

read-only

### Description

This number is purely for unique indexing of this table.

## **tuxTranIdx2**

### Syntax

INTEGER

### Access

read-only

### Description

This number is purely for unique indexing of this table.

## **tuxTranIdx3**

### Syntax

INTEGER

## Access

read-only

## Description

This number is purely for unique indexing of this table.

## **tuxTranIdx4**

### Syntax

INTEGER

## Access

read-only

## Description

This number is purely for unique indexing of this table.

## **tuxTranIdx5**

### Syntax

INTEGER

## Access

read-only

## Description

This number is purely for unique indexing of this table.

## **tuxTranState**

### Syntax

INTEGER { active(1) | abort-only(2) | aborted(3) | com-called(4) | ready(5)  
| decided(6) | suspended(7) }

## Access

read-write

## Description

The values for GET and SET operations are as follows:

GET: active(1) | abort-only(2) | aborted(3) | com-called(4) | ready(5) |  
decided(6) | suspended(7)

A GET operation retrieves run-time information for the selected tuxTranTbl instance(s). The following states indicate the meaning of a tuxTranState object. States not listed are not returned. Note that distinct objects pertaining to the same global transaction (equivalent transaction identifiers) can indicate differing states. In general, the state indicated on the coordinator's site (tuxTranCoordLmid) indicates the true state of the transaction. The exception is when a noncoordinator site notices a condition that transitions the transaction state to abort-only(2). This transition is eventually propagated to the coordinator site and results in the rollback of the transaction, but this change cannot be immediately reflected on the coordinator site.

active(1)

The transaction is active.

abort-only(2)

The transaction has been identified for rollback on the retrieval site.

aborted(3)

The transaction has been identified for rollback and rollback has been initiated on the retrieval site.

com-called(4)

The initiator of the transaction has called tpccommit(3) and the first phase of two-phase commit has begun on the retrieval site.

ready(5)

All of the participating groups on the retrieval site have successfully completed the first phase of two-phase commit and are ready to be committed.

decided(6)

The second phase of the two-phase commit has begun on the retrieval site.

suspended(7)

The initiator of the transaction has suspended processing on the transaction. Note that this state is returned from the initiator's site only.

SET: aborted(3)

A SET operation updates run-time information for the selected tuxTranTbl instance. The following state indicates the meaning of a tuxTranState set in a SET request. States not listed cannot be set.

aborted(3)

Abort the `tuxTranTbl` instance for the application. State change is allowed only when the transaction is in the `active(1)`, `abort-only(2)`, or `com-called(4)` state. Cannot be accompanied by a change to `tuxTranGstate`. Successful return leaves the object in the `aborted(3)` state.

## **tuxTranTimeOut**

### Syntax

INTEGER

### Access

read-only

### Description

Time left (in seconds) before the transaction times out on the retrieval site. Note that the `tuxTranTimeOut` value is returned only when the transaction state is `active(1)`.

## **tuxTranGrpCnt**

### Syntax

INTEGER

### Access

read-only

### Description

Number of groups identified as participants in the transaction by the information returned from the retrieval site.

## **tuxTranGrpIndex**

### Syntax

INTEGER

## Access

read-only

## Description

Index of the first group-specific object values (`tuxTranGrpNo` and `tuxTranGstate`) corresponding to this object.

## tuxTranGrpNo

### Syntax

INTEGER

## Access

read-only

## Description

Group number of the participating group.

## tuxTranGstate

### Syntax

INTEGER { active(1) | aborted(2) | rd-only(3) | ready(4) | hcommit(5) | habort(6) | done(7) | pre-prepare(8) | post-abort(9) | post-commit(10) | unknown(11) }

## Access

read-write

## Description

The values for GET and SET operations are as follows:

GET: active(1) | aborted(2) | rd-only(3) | ready(4) | hcommit(5) | habort(6) | done(7)

A GET operation retrieves run-time information for the selected `tuxTranTbl` instance(s) pertaining to the indicated group. The following states indicate the meaning of a `tuxTranGstate` returned in response to a GET request. States not listed are not returned. Note that distinct objects pertaining to the same global transaction (equivalent transaction

identifiers) can indicate differing states for individual groups. In general, the state indicated on the group's site indicates the true state of the group's participation in the transaction. The exception is when the coordinator site determines that the transaction should abort and sets each participant group state to `aborted(2)`. This transition is propagated to the group's site and results in the rollback of the group's work in the transaction, but cannot be reflected immediately

`active(1)`

The transaction is active in the indicated group.

`aborted(2)`

The transaction has been identified for rollback and rollback has been initiated for the indicated group.

`rd-only(3)`

The group has successfully completed the first phase of two-phase commit and has performed only read operations on the resource manager, thus making it unnecessary to perform the second phase of commit for this group.

`ready(4)`

The group has successfully completed the first phase of two-phase commit and is ready to be committed.

`hcommit(5)`

The group has been heuristically committed. This state might or might not agree with the final resolution of the transaction.

`habort(6)`

The group has been heuristically rolled back. This state might or might not agree with the final resolution of the transaction.

`done(7)`

This group has completed the second phase of the two-phase commit.

`pre-prepare(8)`

Indicates that the transaction group contains Tuxedo servers that have called `xa_end(TMSUSPEND)` during the course of transactional work and that commit processing is beginning. This state exists until either (1) all servers that called `xa_end(TMSUSPEND)` have caused a call to `xa_end(TMSUCCESS)`, at which point the group state becomes `ready`, or (2) one of the target servers does a rollback of the transaction, at which point the group state becomes either `post-abort(9)` or `aborted(2)`.

**Note:** This state is supported only for Tuxedo 8.0 or later applications.

`post-abort(9)`

Indicates that a Tuxedo server called `xa_end(TPFAIL)` and that the TMS has not yet called `xa_rollback()`. In this case, other Tuxedo servers that called `xa_end`

(TMSUSPEND) are being notified by the TMS in order to clean up their associated CORBA objects.

**Note:** This state is supported only for Tuxedo 8.0 or later applications.

post-commit (10)

This state is not implemented yet.

**Note:** This state is supported only for Tuxedo 8.0 or later applications.

SET: hcommit (5) | habort (6)

A SET operation updates run-time information for the first group in the originating request within the selected `tuxTranTbl` instance. The following states indicate the meaning of a `tuxTranGstate` set in a SET request. States not listed cannot be set. State transitions are allowed only when performed within the object representing the group's site.

hcommit (5)

Heuristically commit the group's work as part of the indicated transaction. State change is allowed only when `tuxTranGstate` is ready, `tuxTranState` is ready, and the indicated group is not on the coordinator's site. Successful return leaves the object in the `hcommit (5)` state.

habort (6)

Heuristically rollback the group's work as part of the indicated transaction. State change is allowed only when `tuxTranGstate` is active(1) or ready(4), `tuxTranState` is ready(4), and the indicated group is not on the coordinator's site. Successful return leaves the object in the `habort (6)` state.

## tuxTulogTable

The `tuxTulogTable` group contains objects that represent run-time characteristics of userlog (ULOG) files within an application. The index into this table is `tuxTulogSerNo`. The values returned for objects in this table are controlled by the MIB control group `tuxTulogCtrl`.

Object Name	Object ID
<code>tuxTulogSerNo</code>	.1.3.6.1.4.1.140.300.9.1.1.1
<code>tuxTulogLmid</code>	.1.3.6.1.4.1.140.300.9.1.1.2
<code>tuxTulogPmid</code>	.1.3.6.1.4.1.140.300.9.1.1.3
<code>tuxTulogMmDdYy</code>	.1.3.6.1.4.1.140.300.9.1.1.4

<b>Object Name</b>	<b>Object ID</b>
tuxTulogTime	.1.3.6.1.4.1.140.300.9.1.1.5
tuxTulogLine	.1.3.6.1.4.1.140.300.9.1.1.6
tuxTulogMsg	.1.3.6.1.4.1.140.300.9.1.1.7
tuxTulogTpTranId	.1.3.6.1.4.1.140.300.9.1.1.8
tuxTulogXid	.1.3.6.1.4.1.140.300.9.1.1.9
tuxTulogPid	.1.3.6.1.4.1.140.300.9.1.1.10
tuxTulogSeverity	.1.3.6.1.4.1.140.300.9.1.1.11
tuxTulogCat	.1.3.6.1.4.1.140.300.9.1.1.12
tuxTulogMsgNum	.1.3.6.1.4.1.140.300.9.1.1.13
tuxTulogProcName	.1.3.6.1.4.1.140.300.9.1.1.14
tuxTulogThreadID	.1.3.6.1.4.1.140.300.9.1.1.20
tuxTulogContextID	.1.3.6.1.4.1.140.300.9.1.1.30

## **tuxTulogSerNo**

### Syntax

INTEGER

### Access

read-only

### Description

A running serial number for the rows in tuxTulogTable.

## **tuxTulogLmid**

### Syntax

*DisplayString*(SIZE(1..30))

**Access**

read-only

**Description**

Retrieval machine logical machine identifier.

**tuxTulogPmid****Syntax**

*DisplayString* (SIZE(1..30))

**Access**

read-only

**Description**

Physical machine identifier.

**tuxTulogMmDdYy****Syntax**

INTEGER

**Access**

read-only

**Description**

Month, day, and year of the log file.

**tuxTulogTime****Syntax**

INTEGER

**Access**

read-only

## Description

Time at which the message was generated.

## **tuxTulogLine**

### Syntax

INTEGER

### Access

read-only

## Description

Line number of the message in the log file.

## **tuxTulogMsg**

### Syntax

*DisplayString*(SIZE(1..256))

### Access

read-only

## Description

The entire text of the userlog message as it appears in the userlog file.

## **tuxTulogTpTranId**

### Syntax

*DisplayString*(SIZE(1..78))

### Access

read-only

## Description

Transaction identifier as returned from `tpsuspend(3)`. The data in this field should not be interpreted directly by the user except for equality comparison. Messages not associated with transactions retrieve a 0-length string as the value for this object.

## tuxTulogXid

### Syntax

*DisplayString*(SIZE(1..78))

### Access

read-only

## Description

Transaction identifier as returned from `tx_info(3)`. The data in this field should not be interpreted directly by the user except for equality comparison. Messages not associated with transactions retrieve a 0-length string as the value for this object.

## tuxTulogPid

### Syntax

INTEGER

### Access

read-only

## Description

Process identifier of the client or server that generated the userlog message.

## tuxTulogSeverity

### Syntax

*DisplayString*(SIZE(1..30))

## Access

read-only

## Description

Severity of message, if any.

## **tuxTulogCat**

### Syntax

*DisplayString* (SIZE(1..30))

### Access

read-only

### Description

Catalog name from which the message was derived, if any.

## **tuxTulogMsgNum**

### Syntax

INTEGER

### Access

read-only

### Description

Catalog message number, if the message was derived from a catalog.

## **tuxTulogProcName**

### Syntax

*DisplayString* (SIZE(1..30))

### Access

read-only

## Description

Process name of the client or server that generated the userlog message.

## tuxTulogThreadID

### Syntax

INTEGER

### Access

read-only

## Description

Identifier for the thread that wrote this userlog message.

## tuxTulogContextID

### Syntax

INTEGER

### Access

read-only

## Description

Identifier for this particular application association.

## tuxTulogCtrl

The `tuxTulogCtrl` group contains objects that control the userlog (ULOG) messages returned by the `tuxTulogTable`.

Object Name	Object ID
<code>tuxTulogLmidCtrl</code>	.1.3.6.1.4.1.140.300.9.2.1
<code>tuxTulogPmidCtrl</code>	.1.3.6.1.4.1.140.300.9.2.2

Object Name	Object ID
tuxTulogMmddyCtrl	.1.3.6.1.4.1.140.300.9.2.3
tuxTulogTimeCtrl	.1.3.6.1.4.1.140.300.9.2.4
tuxTulogEndTimeCtrl	.1.3.6.1.4.1.140.300.9.2.5
tuxTulogLineCtrl	.1.3.6.1.4.1.140.300.9.2.6
tuxTulogMsgCtrl	.1.3.6.1.4.1.140.300.9.2.7
tuxTulogTptranIdCtrl	.1.3.6.1.4.1.140.300.9.2.8
tuxTulogXidCtrl	.1.3.6.1.4.1.140.300.9.2.9
tuxTulogPidCtrl	.1.3.6.1.4.1.140.300.9.2.10
tuxTulogSeverityCtrl	.1.3.6.1.4.1.140.300.9.2.11
tuxTulogCatCtrl	.1.3.6.1.4.1.140.300.9.2.12
tuxTulogMsgNumCtrl	.1.3.6.1.4.1.140.300.9.2.13
tuxTulogProcNameCtrl	.1.3.6.1.4.1.140.300.9.2.14

## tuxTulogLmidCtrl

### Syntax

*DisplayString* (SIZE(1..30))

### Access

read-write

### Description

Logical machine ID to qualify machine from where the userlog file is read for tuxTulogTable. By default, the ULOG files from the local host are returned, in accordance to the ULOGPFX. To revert to the default setting, set this object to null.

## tuxTulogPmidCtrl

### Syntax

*DisplayString*(SIZE(1..30))

### Access

read-write

### Description

Physical machine name to qualify the source machine for userlog messages to be listed in `tuxTulogTable`. By default, messages from all hosts within ULOG files qualified by `tuxTulogLmidCtrl` are returned. To revert to the default setting, set this object to `null`.

## tuxTulogMmddyCtrl

### Syntax

INTEGER

### Access

read-write

### Description

Date value to qualify userlog messages listed in `tuxTulogTable`. Default value is current date. To reset the value of the qualifier to its default, set this object to 0.

## tuxTulogTimeCtrl

### Syntax

INTEGER

### Access

read-write

### Description

Starting time of the time range for which the userlog messages are listed in `tuxTulogTable`. This number is calculated as under - "hrs\*10000 + mins\*100 + secs". The default value is 0.

## **tuxTulogEndTimeCtrl**

### Syntax

INTEGER

### Access

read-write

### Description

Ending time of the time range for which the userlog messages are listed in `tuxTulogTable`. This number is calculated as under - “hrs\*10000 + mins\*100 + secs”. By default, the maximum value is considered. To revert to the default setting, set this object to 0.

## **tuxTulogLineCtrl**

### Syntax

INTEGER

### Access

read-write

### Description

Beginning line number from which the userlog messages are listed in `tuxTulogTable`. By default, all messages are returned. To revert to the default setting, set this object to 0.

## **tuxTulogMsgCtrl**

### Syntax

*DisplayString*(SIZE(1..30))

### Access

read-write

## Description

Regular expression to qualify userlog messages listed in `tuxTulogTable` on the basis of the message body. By default, all messages are listed. To revert to the default setting, set this object to `null`.

## tuxTulogTptranIdCtrl

### Syntax

*DisplayString* (SIZE(1..78))

### Access

read-write

## Description

Value of `tuxTpTranId` to qualify messages to be displayed in the in `tuxTulogTable`. By default, all messages are returned. To revert to the default setting, set it to `null`.

## tuxTulogXidCtrl

### Syntax

*DisplayString* (SIZE(1..30))

### Access

read-write

## Description

Value of `tuxTranXid` to qualify messages to be displayed in the in `tuxTulogTable`. By default, all messages are returned. To revert to the default setting, set it to `null`.

## tuxTulogPidCtrl

### Syntax

INTEGER

## Access

read-write

## Description

Value of process Id of the source to qualify messages to be displayed in the `tuxTuLogTable`. By default, messages with any pid are listed. To revert to the default setting, set this object to 0.

## **tuxTuLogSeverityCtrl**

### Syntax

*DisplayString* (SIZE(1..30))

## Access

read-write

## Description

Regular expression to qualify userlog messages to be listed in `tuxTuLogTable` on the basis of message severity, if any. By default, messages with any severity are listed. To revert to the default setting, set this object to `null`.

## **tuxTuLogCatCtrl**

### Syntax

*DisplayString* (SIZE(1..30))

## Access

read-write

## Description

Regular expression to qualify userlog messages to be listed in `tuxTuLogTable` on the basis of the catalog name, if any. By default, messages from all catalogs are listed. To revert to the default setting, set this object to `null`.

## tuxTulogMsgNumCtrl

### Syntax

INTEGER

### Access

read-write

### Description

Message number in catalog to qualify userlog messages to be listed in `tuxTulogTable`. By default, all message numbers are returned. To revert to the default setting, set this object to 0.

## tuxTulogProcNameCtrl

### Syntax

*DisplayString* (SIZE(1..30))

### Access

read-write

### Description

Regular expression to qualify userlog messages to be listed in `tuxTulogTable` on the basis of the process name that generated the message, if known. By default, all messages are returned. To revert to the default setting, set this object to `null`.

## tuxTnetGrpTbl

The `tuxTnetGrpTbl` group contains objects that represent application characteristics of network groups. Network groups are groups of logical machine IDs that can communicate over the network address defined in the `tuxTnetMapNaddr` object in the `tuxTnetMapTbl` table entry. For row creation, a SET request with `tuxTnetGrpName`, `tuxTnetGrpNo`, and `tuxTnetGrpPrio` is required. `tuxTnetGrpNo` provides the index into this table.

<b>Object Name</b>	<b>Object ID</b>
tuxTnetGrpName	.1.3.6.1.4.1.140.300.28.1.1
tuxTnetGrpNo	.1.3.6.1.4.1.140.300.28.1.2
tuxTnetGrpState	.1.3.6.1.4.1.140.300.28.1.3
tuxTnetGrpPrio	.1.3.6.1.4.1.140.300.28.1.4

## **tuxTnetGrpName**

### **Syntax**

*DisplayString*(SIZE(1..30))

### **Access**

read-write

### **Description**

Logical name of the network group. A group name is a string of printable characters and cannot contain a pound sign (#), comma (,), colon (:), or newline character. This object can be updated only during row creation.

## **tuxTnetGrpNo**

### **Syntax**

INTEGER (1..8191)

### **Access**

read-write

### **Description**

Group identifier of the network group. This object can be updated only during row creation.

## tuxTnetGrpState

### Syntax

```
INTEGER { valid(1) | invalid(2) }
```

### Access

read-write

### Description

A GET request retrieves configuration information for the selected tuxTnetGrpTbl instance (or instances). The following states indicate the meaning of the value that is returned:

GET: valid(1)

The instance is defined. This state is the only valid state for this object.

SET: invalid(2)

Delete the selected tuxTnetGrpTbl instance from the application.

States not listed are not returned.

## tuxTnetGrpPrio

### Syntax

```
INTEGER (1..8191)
```

### Access

read-write

### Description

The priority band for this network group. All network groups that have an equivalent band priority are used in parallel.

## tuxTnetMapTbl

The instances in the tuxTnetMapTbl associate tuxTmachineLmids to an instance in the tuxTnetGrpTbl. The rows in this table identify which logical machines belong to which network groups. For row creation, a SET request with at least tuxTnetMapNaddr is needed. The index into this table is provided by tuxTnetMapGrpNo and tuxTnetMapLmid.

<b>Object Name</b>	<b>Object ID</b>
tuxTnetMapGrpName	.1.3.6.1.4.1.140.300.33.1.1
tuxTnetMapGrpNo	.1.3.6.1.4.1.140.300.33.1.2
tuxTnetMapLmid	.1.3.6.1.4.1.140.300.33.1.3
tuxTnetMapState	.1.3.6.1.4.1.140.300.33.1.4
tuxTnetMapNaddr	.1.3.6.1.4.1.140.300.33.1.5
tuxTnetMapMinEncryptBit	.1.3.6.1.4.1.140.300.33.1.6
tuxTnetMapMaxEncryptBit	.1.3.6.1.4.1.140.300.33.1.7

## **tuxTnetMapGrpName**

### **Syntax**

*DisplayString* (SIZE(1..30))

### **Access**

read-write

### **Description**

The logical name of the network group. A group name is a string of printable characters and cannot contain a pound sign (#), comma (,), colon (:), or a newline character.

## **tuxTnetMapGrpNo**

### **Syntax**

INTEGER (1..8191)

### **Access**

read-write

### **Description**

Identifier for this logical network group. This object can be updated only during row creation.

## tuxTnetMapLmid

### Syntax

```
DisplayString(SIZE(1..30))
```

### Access

read-write

### Description

Logical machine name for this network mapping. This object can be updated only during row creation.

## tuxTnetMapState

### Syntax

```
INTEGER { valid(1) | invalid(2) }
```

### Access

read-write

### Description

A GET request retrieves configuration information for the selected tuxTnetMapTbl instance (or instances). The following states indicate the meaning of the value of tuxTnetMapState that is returned:

GET: valid(1)

The instance is defined. This state is the only valid state for this object.

SET: invalid(2)

Delete the selected tuxTnetMapTbl instance from the application. If any network links are active as a result of the mapping, they are disconnected. This disconnection can cause a state change in tuxTBridgeTbl instances associated with the network links.

States not listed are not returned.

## tuxTnetMapNaddr

### Syntax

*DisplayString*(SIZE(1..256)) (up to 78 bytes for Oracle Tuxedo 8.0 or earlier)

### Access

read-write

### Description

Specifies the complete network address to be used by the BRIDGE process placed on the logical machine as its listening address. The listening address for a BRIDGE is the means by which it is contacted by other BRIDGE processes participating in a networked application, that is, if the value of `tuxTdomainOptions` is `lan(1)`. If the string is of the form `0xhex-digits` or `\\xhex-digits`, it must contain an even number of valid hexadecimal digits. These forms are translated internally into a character array containing the hexadecimal representation of the string specified. For TCP/IP addresses, either the `//hostname:port` or `#. #. #. #:port` format is used.

## tuxTnetMapMinEncryptBit

### Syntax

INTEGER { none(1) | 40-bit(2) | 128-bit(3) | unknown(4) }

### Access

read-write

### Description

Specifies the required level of encryption when establishing a network link to this machine.

`none(1)`

No encryption

`40-bit(2)` **and** `128-bit(3)`

These values specify the encryption key length (in bits). If this minimum level of encryption cannot be met, the attempt to establish the link fails.

The default value is `none(1)`. Modifications to this object do not affect network links that have already been established.

## tuxTnetMapMaxEncryptBit

### Syntax

```
INTEGER {none(1) | 40-bit(2) | 128-bit(3) | unknown(4) }
```

### Access

read-write

### Description

Encryption can be negotiated up to the specified level when establishing a network link.

none(1)

No encryption

40-bit(2) **and** 128-bit(3)

These values specify the encryption key length (in bits).

The default value is 128-bit(3). Modifications to this object do not affect network links that are already established.

## tuxTserverCtxtTbl

The `tuxTserverCtxtTbl` group contains objects that represent configuration and run-time characteristics of individual server dispatch contexts within an application. This group is defined for both single-context and multi-context servers. For single-context servers, the object values in this group are repeated as part of the `tuxTsrvrTbl` group. The objects in this group are read-only.

These object values provide run-time tracking of statistics and resources associated with each server dispatch context.

Object Name	Object ID
tuxTserverCtxtGrp	.1.3.6.1.4.1.140.300.34.1.1.10
tuxTserverCtxtServerID	.1.3.6.1.4.1.140.300.34.1.1.20
tuxTserverCtxtContextID	.1.3.6.1.4.1.140.300.34.1.1.30
tuxTserverCtxtCltLmId	.1.3.6.1.4.1.140.300.34.1.1.40
tuxTserverCtxtCltPid	.1.3.6.1.4.1.140.300.34.1.1.50

<b>Object Name</b>	<b>Object ID</b>
tuxTserverCtxtClrtReply	.1.3.6.1.4.1.140.300.34.1.1.60
tuxTserverCtxtCmtRet	.1.3.6.1.4.1.140.300.34.1.1.70
tuxTserverCtxtCurConv	.1.3.6.1.4.1.140.300.34.1.1.80
tuxTserverCtxtCurReq	.1.3.6.1.4.1.140.300.34.1.1.90
tuxTserverCtxtCurService	.1.3.6.1.4.1.140.300.34.1.1.100
tuxTserverCtxtLastGrp	.1.3.6.1.4.1.140.300.34.1.1.110
tuxTserverCtxtSvcTimeOut	.1.3.6.1.4.1.140.300.34.1.1.120
tuxTserverCtxtTimeLeft	.1.3.6.1.4.1.140.300.34.1.1.130
tuxTserverCtxtTranLev	.1.3.6.1.4.1.140.300.34.1.1.140

## **tuxTserverCtxtGrp**

### **Syntax**

*DisplayString* (SIZE(1..30))

### **Access**

read-only

### **Description**

Logical name of the server group. Server group names cannot contain an asterisk (\*), comma (,), or colon (:).

## **tuxTserverCtxtServerID**

### **Syntax**

INTEGER (SIZE(1..30000))

### **Access**

read-only

## Description

Unique (within the server group) server identification number.

## **tuxTserverCtxtContextID**

### Syntax

INTEGER (SIZE(-2..29999))

### Access

read-only

### Description

Identifier of this particular server context.

## **tuxTserverCtxtCltLmId**

### Syntax

INTEGER (SIZE(1..30))

### Access

read-only

### Description

Logical machine for the initiating client or server. The initiating client or server is the process that made the service request on which the server is currently working.

## **tuxTserverCtxtCltPid**

### Syntax

INTEGER

### Access

read-only

## Description

UNIX system process identifier for the initiating client or server.

Limitation: This object is a UNIX-system-specific object that cannot be returned if the platform on which the application is being run is not UNIX-based.

## **tuxTserverCtxtCltReply**

### Syntax

```
INTEGER { yes(1) | no(2) }
```

### Access

read-only

### Description

The initiating client or server is expecting a reply (*yes(1)*) or is not expecting a reply (*no(2)*).

## **tuxTserverCtxtCmtRet**

### Syntax

```
INTEGER { complete(1) | logged(2) }
```

### Access

read-only

### Description

This object value is the setting of the TP\_COMMIT\_CONTROL characteristic for this server.

See the description of the Oracle Tuxedo ATMI function `tpscmt(3c)` for details on this characteristic.

## **tuxTserverCtxtCurConv**

### Syntax

```
INTEGER
```

## Access

read-only

## Description

Number of conversations initiated by this server through `tpconnect()` that are still active.

## **tuxTserverCtxtCurReq**

### Syntax

INTEGER

### Access

read-only

### Description

Number of requests initiated by this server through `tpcall()` or `tpacall()` that are still active.

## **tuxTserverCtxtCurService**

### Syntax

*DisplayString* (SIZE (1..127))

### Access

read-only

### Description

Service name, if any, on which the server is currently working.

## **tuxTserverCtxtLastGrp**

### Syntax

INTEGER (1..29999)

### Access

read-only

## Description

Time left (in seconds), if any, for this server to process the current service request. A value of 0 for an active service indicates that no time out processing is being done.

See `tuxTsvcTbl: tuxTsvcTimeOut` for more information.

## **tuxTserverCtxtSvcTimeOut**

### Syntax

INTEGER

### Access

read-only

### Description

Server group number (`tuxTgroupTable: tuxTgroupNo`) of the last service request made or conversation initiated from this server outward.

## **tuxTserverCtxtTimeLeft**

### Syntax

INTEGER

### Access

read-only

### Description

Time left (in seconds) for this server to receive the reply for which it is currently waiting before it will time out. This timeout can be a transactional timeout or a blocking timeout.

## **tuxTserverCtxtTranLev**

### Syntax

INTEGER

## Access

read-only

## Description

Current transaction level for this server. 0 indicates that the server is not currently involved in a transaction.

# beaEventFilters

You can use the Tuxedo event filters to define a subset of Tuxedo event notifications to be generated for each Tuxedo domain being monitored. The `beaEventFilters` group consists of the following object and group (table).

Object Name	Object ID
<code>beaEvtFilterTblStatus</code>	.1.3.6.1.4.1.140.300.14.1
<code>beaEvtFilterTable</code>	.1.3.6.1.4.1.140.300.14.2

## beaEvtFilterTblStatus

### Syntax

```
INTEGER { sync(1) | dirty(2) }
```

### Access

read-write

### Description

When the agent starts, this object value is always `sync(1)`. If any change is done to `beaEvtFilterTable` through `SET` requests, the value of this object becomes `dirty(2)` and the changes made to `beaEvtFilterTable` do not take effect. The changes made to the `beaEvtFilterTable` take effect only when you set the value of this object to `sync(1)`. When you set the value to `sync(1)`, all changes since the last synchronization are applied to the event-processing module.

## beaEvtFilterTable

The `beaEvtFilterTable` group contains objects that represent all the event filters defined for the SNMP Agent. The object values are used to determine the collection of events to be forwarded as SNMP trap notifications.

The columnar objects in the `beaEvtFilterTable` correspond to fields in the `TMEVENT_FILTER` entries in the Oracle SNMP Agent configuration file (`beamgr.conf`). For more detail, see “[Configuration Files](#)” in the *Oracle SNMP Agent Administration Guide*.

**Note:** Changes to this table are applied only when `beaEvtFilterStatus` is set to `sync(1)`.

Object Name	Object ID
<code>beaEvtFilterId</code>	.1.3.6.1.4.1.140.300.14.1.1.1
<code>beaEvtAgentName</code>	.1.3.6.1.4.1.140.300.14.1.1.2
<code>beaEvtExpr</code>	.1.3.6.1.4.1.140.300.14.1.1.3
<code>beaEvtFilter</code>	.1.3.6.1.4.1.140.300.14.1.1.4
<code>beaEvtFilterState</code>	.1.3.6.1.4.1.140.300.14.1.1.5

### beaEvtFilterId

#### Syntax

*DisplayString(SIZE(1..16))*

#### Access

read-write

#### Description

A unique identifier for the event filter within the filter table.

**Note:** This object can be `SET` only during row creation.

## beaEvtAgentName

### Syntax

*DisplayString*(SIZE(1..32))

### Access

read-only

### Description

This logical agent name of the agent supporting this filter. This object is provided only for user convenience since the MIB only returns the event filters for the agent that was queried.

## beaEvtExpr

### Syntax

*DisplayString*(SIZE(1..255))

### Access

read-write

### Description

An event name expression, which is a regular expression. For the format of regular expressions, see reference page [tpsubscribe\(3c\)](#) in *Oracle Tuxedo ATMI C Function Reference*. For a Tuxedo system event to be forwarded as an SNMP trap, its name should match this expression. For a list of Tuxedo event names, see reference page [EVENTS\(5\)](#) in *Oracle Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*. The default for this object is all system events.

### Examples

`\.Sys.*`

matches all system events. (This event name expression is the default.)

`\.SysServer.*`

matches all system events related to servers.

A value of `NONE` blocks all events from being forwarded by the selected agent and overrides any other filter table entries for the same logical agent name.

## beaEvtFilter

### Syntax

```
DisplayString(SIZE(1..255))
```

### Access

read-write

### Description

An event filter expression. Each Tuxedo event is accompanied by an FML buffer that contains pertinent information about the event. The buffer's contents are evaluated with respect to this filter, if it is present. The filter must evaluate to TRUE or the event is not forwarded.

The SNMP Agent uses this object as an argument to `tpsubscribe()`. For more information, see reference page [tpsubscribe\(3c\)](#) in *Oracle Tuxedo ATMI C Function Reference*.

### Example

```
TA_EVENT_SEVERITY=='ERROR' || TA_EVENT_SEVERITY=='WARN'  
TA_EVENT_SEVERITY!='INFO'  
TA_EVENT_LMID=='SITE1'
```

This filter selects events with a severity of either ERROR or WARNING.

## beaEvtFilterState

### Syntax

```
INTEGER { active(1) | inactive(2) | invalid(3) }
```

### Access

read-write

### Description

This object denotes the current state of the event filter instance.

```
GET { active(1) | inactive(2) }
```

A GET operation retrieves configuration and run-time information for the selected `beaEvtFilterTbl` instance(s). The following states indicate the meaning of a

`beaEvtFilterState` returned in response to a `GET` request. States not listed are not returned.

`active(1)`

This filter is being used.

`inactive(2)`

This filter is not being used.

`SET {active(1) | inactive(2) | invalid(3)}`

A `SET` operation updates configuration and run-time information for the selected `beaEvtFilterTbl` instance. The following states indicate the meaning of a `beaEvtFilterState` set in a `SET` request. States not listed cannot be set.

`active(1)`

Activate the event filter. Activating the event filter can be accomplished only when the filter is in the `inactive(2)` state.

`inactive(2)`

Inactivate the event filter. Inactivating the event filter can be accomplished only when the filter is in the `active(1)` state.

`invalid(3)`

Inactivate (if active) and remove this event filter.



# Domains MIB

The Domains MIB describes the interaction between Oracle Tuxedo domains. The term *access point* defines an object through which you gain access to another object. Therefore, you access a remote domain through a remote domain access point, and remote domains gain access to a local domain through a local domain access point.

The Domains MIB consists of the following groups.

Group Name	Description
<a href="#">tuxDmAclTable</a>	Domains access control list
<a href="#">tuxDmConnectionTable</a>	Domain access points connection status
<a href="#">tuxDmExportTable</a>	Resources for exporting to remote domains
<a href="#">tuxDmImportTable</a>	Resources imported through access points
<a href="#">tuxDmLocalTable</a>	Defines a local domain access point
<a href="#">tuxDmOsitpTable</a>	Defines the local or remote OSI TP protocol
<a href="#">tuxDmPasswordTable</a>	Inter-domain authentication
<a href="#">tuxDmPrincipalMapTable</a>	For mapping principal names
<a href="#">tuxDmRemoteTable</a>	Remote domain configuration information
<a href="#">tuxDmResourcesTable</a>	Domains-specific configuration information

Group Name	Description
<a href="#">tuxDmRoutingTable</a>	Routing criteria information
<a href="#">tuxDmrPrincipalTable</a>	Configuration information for remote principal names
<a href="#">tuxDmSnaCRMTable</a>	SNA-CRM-specific configuration information
<a href="#">tuxDmSnaLinkTable</a>	Snax-specific configuration information
<a href="#">tuxDmSnaStackTable</a>	Defines SNA stack used by a specific SNA CRM
<a href="#">tuxDmTdomainTable</a>	Defines the TDomain specific configuration
<a href="#">tuxDmTransactionTable</a>	Information about transactions that span domains

## tuxDmAcITable

The `tuxDmAcITable` group contains objects that represent access control information for domains.

Object Name	Object ID
<code>tuxDmAcIName</code>	<code>.1.3.6.1.4.1.140.300.80.1.1.10</code>
<code>tuxDmrAccessPointList</code>	<code>.1.3.6.1.4.1.140.300.80.1.1.20</code>
<code>tuxDmAcIState</code>	<code>.1.3.6.1.4.1.140.300.80.1.1.30</code>

## tuxDmAcIName

### Syntax

*DisplayString*(SIZE(1..15))

### Access

read-only

### Description

The access control list name.

## tuxDmrAccessPointList

### Syntax

```
DisplayString(SIZE(1..1000))
```

### Access

read-write

### Description

The list of remote domain access points associated with this access control list.

`tuxDmrAccessPointList` is a comma-separated list of remote access point names (that is, the value of the `tuxDMRemoteDmAccessPoint` object of a valid `tuxDmRemote` object). The list can contain up to 50 remote access point identifier elements. Setting the value of this object to "\*" means that all the remote domains in the configuration are associated with this entry. blank string means no remote access points are associated with this entry. The default is - (equivalent of NULL string).

## tuxDmAcIState

### Syntax

```
INTEGER { valid(1) | invalid(2) }
```

### Access

read-write

### Description

This object denotes the current state of the `tuxDmAcI` instance.

GET requests:

`valid(1)`: `tuxDmAcI` object is defined and inactive. This state is the only valid state for this group. ACL groups are never active.

SET requests:

`invalid(2)`: Delete.

# tuxDmConnectionTable

The `tuxDmConnectionTable` group contains objects that represent the status of connections between domain access points.

Object Name	Object ID
<code>tuxDmConDmlAccessPoint</code>	<code>.1.3.6.1.4.1.140.300.90.1.1.10</code>
<code>tuxDmConDmrAccessPoint</code>	<code>.1.3.6.1.4.1.140.300.90.1.1.20</code>
<code>tuxDmConDmType</code>	<code>.1.3.6.1.4.1.140.300.90.1.1.30</code>
<code>tuxDmConState</code>	<code>.1.3.6.1.4.1.140.300.90.1.1.40</code>
<code>tuxDmConDmCurEncryptBits</code>	<code>.1.3.6.1.4.1.140.300.90.1.1.50</code>

## tuxDmConDmlAccessPoint

### Syntax

*DisplayString*(SIZE(1..24))

### Access

read-only

### Description

The name of the local domain access point that identifies the connection between the domains.

## tuxDmConDmrAccessPoint

### Syntax

*DisplayString*(SIZE(1..24))

### Access

read-only

### Description

The name of the remote domain access point that identifies the connection between the domains

## tuxDmConDmType

### Syntax

```
INTEGER { tdomain(1) }
```

### Access

read-only

### Description

The type of domain, such as tdomain.

## tuxDmConState

### Syntax

```
INTEGER { active(1) | suspended(2) | initializing(3) | inactive(4) |
unknown(5) }
```

### Access

read-only

### Description

This object denotes the current state of the tuxDmConnection instance.

GET requests:

active(1): The connection is active.

suspended(2): The connection is awaiting retry.

initializing(3): The connection is initializing.

inactive(4): The specified domain access points are disconnected. (Returned in case of Tuxedo 7.1 and later only.)

unknown(5): The state cannot be determined.

SET requests:

active(1): Connect the specified domain access points. If the current state is suspended or inactive, SET:active places the connection into the state initializing, otherwise, there is no change.

`inactive(4)`: Disconnect the specified domain access points and destroy the `tuxDmConnection` object

## tuxDmConDmCurEncryptBits

### Syntax

```
INTEGER { enc-0-bit(1) | enc-40-bits(2) | enc-56-bits(3) | enc-128-bits(4) }
```

### Access

read-only

### Description

This object is available when `tuxDmConDmType=tdomain`.

The level of encryption in use on this connection:

`enc-0-bit(1)`  
No encryption

`enc-40-bits(2)`, `enc-56-bits(3)`, **and** `enc-128-bits(4)`  
These values specify the encryption length (in bits).

## tuxDmExportTable

The `tuxDmExportTable` group contains objects that represent local resources that are exported to one or more remote domains through a local access point.

Object Name	Object ID
<code>tuxDmExpDmResourceName</code>	<code>.1.3.6.1.4.1.140.300.100.1.1.10</code>
<code>tuxDmExpDmIAccessPoint</code>	<code>.1.3.6.1.4.1.140.300.100.1.1.20</code>
<code>tuxDmExpDmState</code>	<code>.1.3.6.1.4.1.140.300.100.1.1.30</code>
<code>tuxDmExpDmAclName</code>	<code>.1.3.6.1.4.1.140.300.100.1.1.40</code>
<code>tuxDmExpDmConv</code>	<code>.1.3.6.1.4.1.140.300.100.1.1.50</code>
<code>tuxDmExpDmResourceType</code>	<code>.1.3.6.1.4.1.140.300.100.1.1.60</code>

Object Name	Object ID
tuxDmExpDmRemoteName	.1.3.6.1.4.1.140.300.100.1.1.70
tuxDmExpDmInBufType	.1.3.6.1.4.1.140.300.100.1.1.80
tuxDmExpDmOutBufType	.1.3.6.1.4.1.140.300.100.1.1.90

## tuxDmExpDmResourceName

### Syntax

*DisplayString*(SIZE(1..15))

### Access

read-only

### Description

The local resource name for entries of resource type `service` (the service name), `qspace` (the queue space name), and `qname` (the queue name). For a service entry, the value of this object corresponds to the value of an active `tuxTSrvGrp:tuxTsvcName` object. This resource is exported to other domains with the same name or with the alias defined in the `tuxDmExpDmRemoteName` objects.

## tuxDmExpDmlAccessPoint

### Syntax

*DisplayString*(SIZE(1..30))

### Access

read-only

### Description

The local access point name. Setting this object to “\*” means the resource is available at all local access points.

## **tuxDmExpState**

### Syntax

```
INTEGER { valid(1) | invalid(2) }
```

### Access

read-write

### Description

This object denotes the current state of the `tuxDmExport` instance.

GET requests:

`valid(1)`: The object exists.

SET requests:

`invalid(2)`: Delete object.

## **tuxDmExpDmAcIName**

### Syntax

```
DisplayString(SIZE(1..15))
```

### Access

read-write

### Description

The name of a `tuxDmAcI` object to use for security on this local service. This object is not permitted if `tuxDmExpDmResourceType=qname`.

## **tuxDmExpDmConv**

### Syntax

```
INTEGER { yes(1) | no(2) }
```

### Access

read-only

## Description

Specifies whether this local service is conversational or not.

## tuxDmExpDmResourceType

### Syntax

```
INTEGER { service(1) | qspace(2) | qname(3) }
```

### Access

read-only

## Description

Specifies whether this entry is for a service, qspace, or qname. The default is *service*.

## tuxDmExpDmRemoteName

### Syntax

```
DisplayString(SIZE(1..127))
```

### Access

read-write

## Description

For entries of type *service* or *qspace*, this object specifies the name exported through non-topend remote access points.

## tuxDmExpDmInBufType

### Syntax

```
DisplayString(SIZE(1..513))
```

### Access

read-write

## Description

Attributes available from remote access points of `tuxDmExpDmResourceType=snax|ositp:`  
`type[:subtype]` -Input buffer type, optionally followed by subtype.

If this object is present, it defines the buffer type (and subtype) accepted. This object should be defined for entries of `tuxDmExpDmResourceType=service` when access is permitted from remote access points using `ositp` with the UDT application context, or when using `snax`. This object is not permitted if `tuxDmExpDmResourceType=qspace`.

## tuxDmExpDmOutBufType

### Syntax

*DisplayString* (SIZE(1..513))

### Access

read-write

### Description

Attributes available from remote access points of `tuxDmExpDmResourceType=snax|ositp:`  
`type[:subtype]` -Output buffer type, optionally followed by subtype.

If this object is present, it defines the buffer type (and subtype) output by the service. This object should be defined for entries of `tuxDmExpDmResourceType=service` when access is permitted from remote access points using `ositp` with the UDT application context, or when using `snax`. This object is not permitted if `tuxDmExpDmResourceType=qspace` and `qname`.

## tuxDmImportTable

The `tuxDmImportTable` group contains objects that represent remote resources that are imported through one or more remote domain access points and made available to the local domain through one or more local domain access points.

Object Name	Object ID
<code>tuxDmImpDmResourceName</code>	.1.3.6.1.4.1.140.300.110.1.1.10
<code>tuxDmImpDmrAccessPointList</code>	.1.3.6.1.4.1.140.300.110.1.1.20

<b>Object Name</b>	<b>Object ID</b>
tuxDmImpDmIAccessPoint	.1.3.6.1.4.1.140.300.110.1.1.30
tuxDmImpDmState	.1.3.6.1.4.1.140.300.110.1.1.40
tuxDmImpDmAutoTran	.1.3.6.1.4.1.140.300.110.1.1.50
tuxDmImpDmConv	.1.3.6.1.4.1.140.300.110.1.1.60
tuxDmImpDmLoad	.1.3.6.1.4.1.140.300.110.1.1.70
tuxDmImpDmPrio	.1.3.6.1.4.1.140.300.110.1.1.80
tuxDmImpDmResourceType	.1.3.6.1.4.1.140.300.110.1.1.90
tuxDmImpDmRemoteName	.1.3.6.1.4.1.140.300.110.1.1.100
tuxDmImpDmRoutingName	.1.3.6.1.4.1.140.300.110.1.1.110
tuxDmImpDmTranTime	.1.3.6.1.4.1.140.300.110.1.1.120
tuxDmImpDmInBufType	.1.3.6.1.4.1.140.300.110.1.1.130
tuxDmImpDmOutBufType	.1.3.6.1.4.1.140.300.110.1.1.140
tuxDmImpDmteProduct	.1.3.6.1.4.1.140.300.110.1.1.150
tuxDmImpDmteFunction	.1.3.6.1.4.1.140.300.110.1.1.160
tuxDmImpDmteTarget	.1.3.6.1.4.1.140.300.110.1.1.170
tuxDmImpDmteQualifier	.1.3.6.1.4.1.140.300.110.1.1.180
tuxDmImpDmteRtqGroup	.1.3.6.1.4.1.140.300.110.1.1.190
tuxDmImpDmteRtqName	.1.3.6.1.4.1.140.300.110.1.1.200

## **tuxDmImpDmResourceName**

### **Syntax**

*DisplayString* (SIZE(1..127))

### **Access**

read-only

## Description

The remote resource name used for entries of resource type `service` (the service name, `qspace` (the queue space name), and `qname` (the queue name). This resource is imported from remote domains with the same name or with the alias defined in the `tuxDmImpDmRemoteName` or `tuxDmImpDmte*` objects.

## **tuxDmImpDmAccessPointList**

### Syntax

*DisplayString* (SIZE(1..15))

### Access

read-only

### Description

Identifies the remote domain access point through which this resource should be imported. This object value is a comma-separated failover domain list; it can contain up to three remote domain access points. If this object is set to “\*”, the resource can be imported from all remote access points.

## **tuxDmImpDmAccessPoint**

### Syntax

*DisplayString* (SIZE(1..15))

### Access

read-only

### Description

The name of the local domain access point through which this imported resource should be made available. If set to the null string, the resource is made available through all local domain access points.

## tuxDmImpState

### Syntax

```
INTEGER { valid(1) | invalid(2) }
```

### Access

read-write

### Description

This object denotes the current state of the tuxDmImport instance.

GET requests:

valid(1): The object exists.

SET requests:

invalid(2): The object is deleted. A state change is allowed in the active or suspended state and results in the invalid state.

## tuxDmImpDmAutoTran

### Syntax

```
INTEGER { yes(1) | no(2) }
```

### Access

read-write

### Description

When a request is received for a resource that is not already within a transaction, this object automatically starts a transaction for the resource. The default is no(2).

## tuxDmImpDmConv

### Syntax

```
INTEGER { yes(1) | no(2) }
```

### Access

read-write

## Description

A boolean value (yes or no) specifies whether the service is conversational.

## **tuxDmImpDmLoad**

### Syntax

INTEGER (1..32767)

### Access

read-write

## Description

The service load.

## **tuxDmImpDmPrio**

### Syntax

INTEGER (1..100)

### Access

read-write

## Description

The dequeuing priority. Service requests with a higher priority are serviced first.

## **tuxDmImpDmResourceType**

### Syntax

INTEGER { service(1) | qspace(2) | qname(3) }

### Access

read-write

## Description

Specifies whether this entry is for a service, qspace, or qname. The default is *service*.

## tuxDmImpDmRemoteName

### Syntax

*DisplayString*(SIZE(1..127))

### Access

read-write

### Description

For entries of type *service* or *qspace*, this object specifies the name imported through non-topend remote access points.

## tuxDmImpDmRoutingName

### Syntax

*DisplayString*(SIZE(1..15))

### Access

read-write

### Description

The name of a *tuxDmRoutingTable* object to use for routing criteria for this *service* or *qspace*.

## tuxDmImpDmTranTime

### Syntax

INTEGER (1..32767)

### Access

read-write

### Description

Transaction time value (in seconds) of transactions automatically started for this *service* or *qspace*. Transactions are started automatically when a request not in transaction mode is received and the *tuxDmImpDmAutoTran* object is set to *yes*.

Limitation: Run-time updates to this object are not reflected in active requests.

## **tuxDmImpDmInBufType**

### Syntax

*DisplayString*(SIZE(1..256))

### Access

read-write

### Description

Attributes available from remote access points of `tuxDmRemoteDmType=snax|ositp`:

`type[:subtype]` - Input buffer type, optionally followed by subtype. If this object is present, it defines the buffer type (and subtype) accepted. This object should be defined for entries of `DMRESOURCEType=service` when access is permitted to remote access points that use `ositp` with the UDT application context, or that use `snax`. This object is not permitted if `tuxDmImpDmResourceType=qspace`.

## **tuxDmImpDmOutBufType**

### Syntax

*DisplayString*(SIZE(1..256))

### Access

read-write

### Description

Attributes available from remote access points of `tuxDmRemoteDmType=snax|ositp`:

## **tuxDmLocalTable**

The `tuxDmLocalTable` group defines a local domain access point. A local domain access point is used to control access to local services exported to remote domains and to control access to remote services imported from remote domains.

<b>Object Name</b>	<b>Object ID</b>
tuxDmLclDmAccessPoint	.1.3.6.1.4.1.140.300.120.1.1.10
tuxDmLclDmAccessPointID	.1.3.6.1.4.1.140.300.120.1.1.20
tuxDmLclDmSrvGroup	.1.3.6.1.4.1.140.300.120.1.1.30
tuxDmLclDmType	.1.3.6.1.4.1.140.300.120.1.1.40
tuxDmLclDmState	.1.3.6.1.4.1.140.300.120.1.1.50
tuxDmLclDmAuditLog	.1.3.6.1.4.1.140.300.120.1.1.60
tuxDmLclDmBlockTime	.1.3.6.1.4.1.140.300.120.1.1.70
tuxDmLclDmMaxRapTran	.1.3.6.1.4.1.140.300.120.1.1.80
tuxDmLclDmMaxTran	.1.3.6.1.4.1.140.300.120.1.1.90
tuxDmLclDmSecurity	.1.3.6.1.4.1.140.300.120.1.1.100
tuxDmLclDmTlogDev	.1.3.6.1.4.1.140.300.120.1.1.110
tuxDmLclDmTlogName	.1.3.6.1.4.1.140.300.120.1.1.120
tuxDmLclDmTlogSize	.1.3.6.1.4.1.140.300.120.1.1.130
tuxDmLclDmConnectionPolicy	.1.3.6.1.4.1.140.300.120.1.1.140
tuxDmLclDmRetryInterval	.1.3.6.1.4.1.140.300.120.1.1.150
tuxDmLclDmMaxRetry	.1.3.6.1.4.1.140.300.120.1.1.160
tuxDmLclDmConnPrincipalName	.1.3.6.1.4.1.140.300.120.1.1.170
tuxDmLclDmMachineType	.1.3.6.1.4.1.140.300.120.1.1.180
tuxDmLclDmBlobShmSize	.1.3.6.1.4.1.140.300.120.1.1.190

## **tuxDmLclDmAccessPoint**

### **Syntax**

*DisplayString* (SIZE(1..30))

## Access

read-only

## Description

This object value is an identifier unique within the scope of `tuxDmLocal` and `tuxDmRemote` entry names in the domain configuration.

## **tuxDmLclDmAccessPoint**

### Syntax

*DisplayString*(SIZE(1..30))

## Access

read-write

## Description

The domain access point identifier. This identifier is unique across all local and remote domain access points.

## **tuxDmLclDmSrvGroup**

### Syntax

*DisplayString*(SIZE(1..30))

## Access

read-write

## Description

The group in which the administrative servers and gateway processes of the local domain reside.

## **tuxDmLclDmType**

### Syntax

INTEGER { tdomain(1) | ositp(2) | snax(3) }

## Access

read-write

## Description

The type of domain: `tdomain` for an Oracle Tuxedo domain, `ositp` for an OSI domain, and `snax` for an SNA domain. The presence or absence of other objects depends on the value of this object.

## tuxDmLclState

### Syntax

```
INTEGER { valid(1) | invalid(2) }
```

### Access

read-write

### Description

This object denotes the current state of the tuxDmLocal instance.

GET requests:

`valid(1)`: The object exists.

SET requests:

`invalid(2)`: The object is deleted.

## tuxDmLclDmAuditLog

### Syntax

*DisplayString*(SIZE(1..256)) (up to 78 bytes for Oracle Tuxedo 8.0 or earlier)

### Access

read-write

### Description

The name of the audit log file for this local domain.

## **tuxDmLcIDmBlockTime**

### **Syntax**

INTEGER (0..32767)

### **Access**

read-write

### **Description**

Specifies the maximum wait time allowed for a blocking call. The value sets a multiplier of the SCANUNIT parameters specified in the tuxTdomain group. The value SCANUNIT \* tuxDmLcIDmBlockTime must be greater than or equal to SCANUNIT and less than 32,768 seconds. If this object is not specified, the default is set to the value of the tuxDmLcIDmBlockTime object specified in the tuxTdomain object. A timeout always implies a failure of the affected request. Notice that the timeout specified for transactions in the tuxTdomain is always used when the request is issued within a transaction.

## **tuxDmLcIDmMaxRapTran**

### **Syntax**

INTEGER (0..32767)

### **Access**

read-write

### **Description**

The maximum number of remote domain access points that can be involved in a single transaction.

## **tuxDmLcIDmMaxTran**

### **Syntax**

INTEGER (0..32767)

### **Access**

read-write

## Description

The maximum number of simultaneous transactions allowed on this local domain access point. This number must be greater than or equal to the `tuxTdomainMaxGTT` object in the `tuxTdomain` group.

## tuxDmLcIDmSecurity

### Syntax

```
INTEGER { none(1) | app-pw(2) | dm-pw(3) | dm-user-pw(4) | te-clear(5) |
te-safe(6) | te-private(7) }
```

### Access

read-write

### Description

The type of security enabled on this domain. This object must be set to one of the following:

`none(1)`

No security is enabled.

`app-pw(2)`

Valid only when `tuxDmRemoteDmType=tdomain`. Application password-based security is enabled.

`dm-pw(3)`

Valid only when `tuxDmRemoteDmType=tdomain`. Domain password-based security is enabled.

`dm-user-pw(4)`

Valid only when `tuxDmRemoteDmType=snax`. Translation of principal names is enabled.

## tuxDmLcIDmTlogDev

### Syntax

*DisplayString* (SIZE(1..256)) (up to 78 bytes for Oracle Tuxedo 8.0 or earlier)

### Access

read-write

## Description

The device (raw slice) or file that contains the domain TLOG for this local domain access point. The TLOG is stored as an Oracle Tuxedo system VTOC table on the device. For reliability, the use of a device (raw slice) is recommended.

## **tuxDmLcIDmTlogName**

### Syntax

```
DisplayString(SIZE(1..30))
```

### Access

read-write

## Description

The domain TLOG name for this local domain access point. If more than one TLOG exists on the same device, each TLOG must have a unique name.

## **tuxDmLcIDmTlogSize**

### Syntax

```
INTEGER
```

### Access

read-write

## Description

The size in pages of the TLOG for this local domain access point. This size is constrained by the amount of space available on the device identified in `tuxDmLc1TlogDev`.

## **tuxDmLcIDmConnectionPolicy**

### Syntax

```
INTEGER { on-demand(1) | on-startup(2) | incoming-only(3) }
```

## Access

read-write

## Description

Attributes available when `tuxDmRemoteDmType=tdomain`.

Specifies the conditions under which a local domain gateway tries to establish a connection to a remote domain. Supported values are:

`on-demand` (1)

Means that a connection is attempted only when requested by either a client request to a remote service or an administrative “connect” command. The default setting for this object is `on-demand`. The `on-demand` policy provides the equivalent behavior to previous releases, in which this object was not explicitly available.

`on-startup` (2)

Means that a domain gateway attempts to establish a connection with its remote domain access points at gateway server initialization time. Remote services, (that is, services advertised by the domain gateway for this local access point) are advertised only if a connection is successfully established to that remote domain access point. Therefore, if there is no active connection to a remote domain access point, the remote services are suspended. By default, this connection policy retries failed connections every 60 seconds; however, you can specify a different value for this interval using the `tuxDmLclMaxRetry` and `tuxDmLclDmRetryInterval` objects.

## tuxDmLclDmRetryInterval

### Syntax

INTEGER

### Access

read-write

### Description

The number of seconds between automatic attempts to establish a connection to remote domain access points. The minimum value is 0 and the maximum value is 2147483647. The default setting is 60. If `tuxDmLclDmMaxRetry` is set to 0, setting `tuxDmLclDmRetryInterval` is not allowed.

This object is valid only when the `tuxDmLclDmConnectionPolicy` object is set to `on-startup`. For other connection policies, automatic retries are disabled.

## **tuxDmLclDmMaxRetry**

### Syntax

INTEGER

### Access

read-write

### Description

The number of times that a domain gateway tries to establish connections to remote domain access points. The minimum value is 0 and the maximum is `MAXLONG`. `MAXLONG` indicates that retry processing is repeated indefinitely, or until a connection is established. For a connection policy of `on-startup`, the default setting for `tuxDmLclMaxRetry` is `MAXLONG`. Setting this object to 0 turns off the auto retry mechanism. For other connection policies, auto retries are disabled.

The `tuxDmLclMaxRetry` object is valid only when the connection policy is `on-startup`.

## **tuxDmLclDmConnPrincipalName**

### Syntax

*DisplayString*(SIZE(1..511))

### Access

read-write

### Description

The connection principal name identifier. This object value is the principal name used for verifying the identity of this local domain access point when establishing a connection to a remote domain access point. This object only applies to domains of type `TDOMAIN` that are running Oracle Tuxedo 7.1 or later software.

This object can contain a maximum of 511 characters (excluding the terminating null character). If this object is not specified, the connection principal name defaults to the `tuxDmLclDmAccessPointId` string for this local domain access point.

For default authentication plug-ins, if a value is assigned to this object for this local domain access point, it must be the same as the value assigned to the `tuxDmLclAccessPointId` object for this local domain access point. If these values do not match, the local domain gateway process

does not boot and the system generates the following `userlog(3c)` message: “ERROR: Unable to acquire credentials”.

## tuxDmLcIDmMachineType

### Syntax

*DisplayString*(SIZE(1..15))

### Access

read-write

### Description

Used for grouping domains so that encoding/decoding of messages between domains can be bypassed. If no value is specified, the default is to turn encoding/decoding on. If the value set for this object is the same in both the `DM_LOCAL` and the `DM_REMOTE` section of a domain configuration file, data encoding/decoding is bypassed. The value set for this object can be any string value up to 15 characters in length. The value is used only for comparison.

This object is valid only when `tuxDmRemoteDmType=tdomain`.

## tuxDmLcIDmBlobShmSize

### Syntax

INTEGER

### Access

read-write

### Description

This object is relevant only to local domain access point entries. It specifies the amount of shared memory allocated to storing binary large object log information specific to `ositp`.

## tuxDmOsitpTable

The `tuxDmOsitpTable` group contains objects that define the OSI TP protocol-related configuration information for a specific local or remote domain access point.

Object Name	Object ID
tuxDmOsiDmAccessPoint	.1.3.6.1.4.1.140.300.130.1.1.10
tuxDmOsiDmState	.1.3.6.1.4.1.140.300.130.1.1.20
tuxDmOsiDmApt	.1.3.6.1.4.1.140.300.130.1.1.30
tuxDmOsiDmAeq	.1.3.6.1.4.1.140.300.130.1.1.40
tuxDmOsiDmNwDevice	.1.3.6.1.4.1.140.300.130.1.1.50
tuxDmOsiDmAcn	.1.3.6.1.4.1.140.300.130.1.1.60
tuxDmOsiDmApid	.1.3.6.1.4.1.140.300.130.1.1.70
tuxDmOsiDmAeid	.1.3.6.1.4.1.140.300.130.1.1.80
tuxDmOsiDmUrch	.1.3.6.1.4.1.140.300.130.1.1.90
tuxDmOsiDmMaxListeningEp	.1.3.6.1.4.1.140.300.130.1.1.100
tuxDmOsiDmXatmiEncoding	.1.3.6.1.4.1.140.300.130.1.1.110

## tuxDmOsiDmAccessPoint

### Syntax

*DisplayString*(SIZE(1..30))

### Access

read-only

### Description

The domain access point name for which this entry provides the protocol-specific configuration information. This object matches the domain access point name given in the `tuxDmLocal` or `tuxDmRemote` entry that defines the protocol-independent configuration of the domain access point.

## tuxDmOsiDmState

### Syntax

```
INTEGER { valid(1) | invalid(2) }
```

### Access

read-write

### Description

This object denotes the current state of the tuxDmOsiTp instance.

GET requests:

valid(1): The object exists.

SET requests:

invalid(2): The object is deleted.

## tuxDmOsiDmApt

### Syntax

```
DisplayString(SIZE(1..78))
```

### Access

read-write

### Description

The application process title of the domain access point in object identifier form.

## tuxDmOsiDmAeq

### Syntax

```
DisplayString(SIZE(1..78))
```

### Access

read-write

### Description

The application entity qualifier of the domain access point in integer form.

## **tuxDmOsiDmNwDevice**

### Syntax

*DisplayString*(SIZE(1..78))

### Access

read-write

### Description

This object, which specifies the network device to be used, is relevant when it defines a local domain access point and ignored for a remote domain access point.

## **tuxDmOsiDmAcn**

### Syntax

INTEGER { atmi(1) | udt(2) }

### Access

read-write

### Description

The application context name to use with this domain access point. When you establish a dialogue to a remote domain access point, use the application context name from the remote domain access point, if it is present. If the application context name from the remote domain access point is absent, use the application context name from the local domain access point. The value `xatmi` selects the use of the X\Open-defined `xatmi` Application Service Element (ASE) and encoding. The value `udt` selects the use of the ISO/IEC 10026-5 User Data Transfer encoding.

## **tuxDmOsiDmApid**

### Syntax

INTEGER

### Access

read-write

## Description

This optional object defines the application process-invocation identifier to be used on this domain access point.

## **tuxDmOsiDmAeid**

### Syntax

INTEGER

### Access

read-write

## Description

This optional object defines the application entity-invocation identifier to be used on this domain access point.

## **tuxDmOsiDmUrch**

### Syntax

*DisplayString*(SIZE(1..30))

### Access

read-write

## Description

This object specifies the user portion of the OSI TP recovery context handle. It can be required by an OSI TP provider in order to perform recovery of distributed transactions after a communication line or system failure.

This object is relevant for defining a local domain access point and ignored for a remote domain access point.

## **tuxDmOsiDmMaxListeningEp**

### Syntax

INTEGER (1..32767)

## Access

read-write

## Description

This object specifies the number of endpoints awaiting incoming OSI TP dialogue. It is relevant for defining a local domain access point and ignored for a remote domain access point.

# tuxDmOsiDmXatmiEncoding

## Syntax

```
INTEGER { cae(1) | preliminary(2) | oltp-tm2200(3) }
```

## Access

read-write

## Description

This object specifies the version of the XATMI protocol used to communicate with a remote system. Valid values are: `cae`, `preliminary`, and `oltp-tm2200`.

This object is relevant for remote domain access points and ignored for local domain access points.

# tuxDmPasswordTable

The `tuxDmPasswordTable` group contains objects that represent configuration information for inter-domain authentication through access points of type `tdomain`.

Object Name	Object ID
<code>tuxDmPasswdDmlAccessPoint</code>	<code>.1.3.6.1.4.1.140.300.140.1.1.10</code>
<code>tuxDmPasswdDmrAccessPoint</code>	<code>.1.3.6.1.4.1.140.300.140.1.1.20</code>
<code>tuxDmPasswdDmlPWD</code>	<code>.1.3.6.1.4.1.140.300.140.1.1.30</code>
<code>tuxDmPasswdDmrPWD</code>	<code>.1.3.6.1.4.1.140.300.140.1.1.40</code>
<code>tuxDmPasswdState</code>	<code>.1.3.6.1.4.1.140.300.140.1.1.50</code>

## **tuxDmPasswdDmlAccessPoint**

### Syntax

*DisplayString*(SIZE(1..24))

### Access

read-only

### Description

The name of the local domain access point to which the password applies.

## **tuxDmPasswdDmrAccessPoint**

### Syntax

*DisplayString*(SIZE(1..24))

### Access

read-only

### Description

The name of the remote domain access point to which the password applies.

## **tuxDmPasswdDmlPWD**

### Syntax

*DisplayString*(SIZE(1..30))

### Access

write-only

### Description

The local password used to authenticate connections between the local domain access point identified by `tuxDmPasswdDmlAccessPoint` and the remote domain access point identified by `tuxDmPasswdDmrAccessPoint`.

## tuxDmPasswdDmrPWD

### Syntax

*DisplayString*(SIZE(1..30))

### Access

write-only

### Description

The remote password used to authenticate connections between the local domain access point identified by `tuxDmPasswdDmlAccessPoint` and the remote domain access point identified by `tuxDmPasswdDmrAccessPoint`.

## tuxDmPasswdState

### Syntax

INTEGER { valid(1) | invalid(2) | reencrypt(3) }

### Access

read-write

### Description

This object denotes the current state of the `tuxDmPassword` instance.

GET requests:

`valid(1)`: The object exists.

SET requests:

`invalid(2)`: The object is deleted.

`reencrypt(3)`: Re-encrypt all passwords using a new encryption key.

## tuxDmPrincipalMapTable

The `tuxDmPrincipalMapTable` group contains objects that represent configuration information for mapping principal names to and from external principal names across access point of type `snax`.

<b>Object Name</b>	<b>Object ID</b>
tuxDmPrinMapDmlAccessPoint	.1.3.6.1.4.1.140.300.150.1.1.10
tuxDmPrinMapDmrAccessPoint	.1.3.6.1.4.1.140.300.150.1.1.20
tuxDmPrinMapDmlPrinName	.1.3.6.1.4.1.140.300.150.1.1.30
tuxDmPrinMapDmrPrinName	.1.3.6.1.4.1.140.300.150.1.1.40
tuxDmPrinMapDirection	.1.3.6.1.4.1.140.300.150.1.1.50
tuxDmPrinMapState	.1.3.6.1.4.1.140.300.150.1.1.60

## **tuxDmPrinMapDmlAccessPoint**

### Syntax

*DisplayString* (SIZE(1..12))

### Access

read-only

### Description

The local domain access point to which the principal mapping applies.

## **tuxDmPrinMapDmrAccessPoint**

### Syntax

*DisplayString* (SIZE(1..12))

### Access

read-only

### Description

The remote domain access point to which the principal mapping applies.

## **tuxDmPrinMapDmlPrinName**

### Syntax

*DisplayString*(SIZE(1..12))

### Access

read-only

### Description

The local principal name in the principal mapping.

## **tuxDmPrinMapDmrPrinName**

### Syntax

*DisplayString*(SIZE(1..12))

### Access

read-only

### Description

The remote principal name in the principal mapping.

## **tuxDmPrinMapDirection**

### Syntax

INTEGER { in(1) | out(2) | both(3) }

### Access

read-write

### Description

The direction to which the principal mapping applies.

in(1)

Is incoming to this Oracle Tuxedo domain through the given remote domain access point and local domain access point.

out (2)

Is outgoing from this Oracle Tuxedo domain through the given local domain access point and remote domain access point.

both (3)

Applies to both incoming to and outgoing from this Oracle Tuxedo domain through the given local domain access point and remote domain access point.

## tuxDmPrinMapState

### Syntax

```
INTEGER { valid(1) | invalid(2) }
```

### Access

read-write

### Description

This object denotes the current state of the `tuxDmPrincipalMap` instance.

GET requests:

`valid(1)`: The object exists.

SET requests:

`invalid(2)`: The object is deleted.

## tuxDmRemoteTable

The `tuxDmRemoteTable` group contains objects that represent remote domain access point configuration information. Local resources that can be exported through one or more local domain access points are made accessible to a remote domain through a remote domain access point. Similarly, remote resources are imported from a remote domain through a remote domain access point.

Object Name	Object ID
<code>tuxDmRemoteDmAccessPoint</code>	.1.3.6.1.4.1.140.300.160.1.1.10
<code>tuxDmRemoteDmAccessPointID</code>	.1.3.6.1.4.1.140.300.160.1.1.20
<code>tuxDmRemoteType</code>	.1.3.6.1.4.1.140.300.160.1.1.30

Object Name	Object ID
tuxDmRemoteState	.1.3.6.1.4.1.140.300.160.1.1.40
tuxDmRemoteDmCodePage	.1.3.6.1.4.1.140.300.160.1.1.50
tuxDmRemoteDmMachineType	.1.3.6.1.4.1.140.300.160.1.1.90

## tuxDmRemoteDmAccessPoint

### Syntax

*DisplayString*(SIZE(1..30))

### Access

read-only

### Description

The name of this `tuxDmRemote` entry. This object value is an identifier unique within the scope of `tuxDmLocal` and `tuxDmRemote` entry names in the domain configuration.

## tuxDmRemoteDmAccessPointID

### Syntax

*DisplayString*(SIZE(1..30))

### Access

read-write

### Description

The access point identifier. This identifier is unique across all local and remote domain access points.

## tuxDmRemoteType

### Syntax

INTEGER { tdomain(1) | ositp(2) | snax(3) }

## Access

read-write

## Description

The type of domain:

`tdomain(1)`

An Oracle Tuxedo domain.

`ositp(2)`

An OSI domain.

`snax(3)`

An SNA domain.

The presence or absence of other objects depends on the value of this object.

## tuxDmRemoteState

### Syntax

```
INTEGER { valid(1) | invalid(2) }
```

### Access

read-write

### Description

This object denotes the current state of the `tuxDmRemote` instance.

GET requests:

`valid(1)`: The object exists.

SET requests:

`invalid(2)`: The object is deleted.

## tuxDmRemoteDmCodePage

### Syntax

```
DisplayString (SIZE(1..20))
```

### Access

read-write

## Description

Attributes available when `tuxDmRemoteDmType=snax`. The name of the default translation tables to use when translating requests and replies that are sent through this access point.

## tuxDmRemoteDmMachineType

### Syntax

*DisplayString*(SIZE(1..15))

### Access

read-write

## Description

Attributes available when `tuxDmRemoteDmType=tomain`.

These objects are used for grouping domains so that encoding/decoding of messages between domains can be bypassed. If it is not specified, the default is to turn encoding/decoding on. If the value set for this object is the same in both the `DM_LOCAL` and the `DM_REMOTE` sections of a domain configuration file, data encoding/decoding is bypassed. The value set for this object can be any string value up to 15 characters in length. The object value is used only for comparison.

## tuxDmResourcesTable

The `tuxDmResourcesTable` group contains an object that represents Domains-specific configuration information.

Object Name	Object ID
<code>tuxDmResourcesDmVersion</code>	<code>.1.3.6.1.4.1.140.300.170.1.1.10</code>

## tuxDmResourcesDmVersion

### Syntax

*DisplayString*(SIZE(1..30))

## Access

read-only

## Description

A user-supplied identifier for the Domains configuration.

# tuxDmRoutingTable

The `tuxDmRoutingTable` group contains objects that represent routing criteria information for routing requests to a domain through a remote domain access point.

Object Name	Object ID
<code>tuxDmRoutingDmRoutingName</code>	.1.3.6.1.4.1.140.300.180.1.1.10
<code>tuxDmRoutingDmBufType</code>	.1.3.6.1.4.1.140.300.180.1.1.20
<code>tuxDmRoutingDmField</code>	.1.3.6.1.4.1.140.300.180.1.1.30
<code>tuxDmRoutingDmFieldType</code>	.1.3.6.1.4.1.140.300.180.1.1.40
<code>tuxDmRoutingDmRanges</code>	.1.3.6.1.4.1.140.300.180.1.1.50
<code>tuxDmRoutingState</code>	.1.3.6.1.4.1.140.300.180.1.1.60

## tuxDmRoutingDmRoutingName

### Syntax

*DisplayString* (SIZE(1..15))

### Access

read-only

### Description

The name of the routing criteria table entry.

## tuxDmRoutingDmBufType

### Syntax

*DisplayString*(SIZE(1..30))

### Access

read-only

### Description

List of types and subtypes of data buffers for which this routing entry is valid.

type1[:subtype1[, subtype2...]][:type2[:subtype3[, subtype4...]]...]

A maximum of 32 type/subtype combinations is allowed. The types are restricted to the following: FML, XML, VIEW, X-C-TYPE, or X\_COMMON. No subtype can be specified for FML or XML; subtypes are required for types VIEW, X\_C\_TYPE, and X\_COMMON (“\*” is not allowed).

Note that subtype names should not contain semicolon (;), colon (:), comma (,), or asterisk (\*) characters. Duplicate type/subtype pairs cannot be specified for the same routing criterion name; more than one routing entry can have the same criterion name as long as the type/subtype pairs are unique. If multiple buffer types are specified for a single routing entry, the data types of the routing field for each buffer type must be the same.

## tuxDmRoutingDmField

### Syntax

*DisplayString*(SIZE(1..30))

### Access

read-write

### Description

This object value is the routing field name. This field is assumed to be an FML buffer, XML buffer, or VIEW field name that is identified in an FML field table (using the FLDTBLDIR and FIELDTBLS environment variables), or an FML VIEW table (using the VIEWDIR and VIEWFILES environment variables), respectively. This information is used to get the associated field value for data-dependent routing to an access point of a remote domain.

For an XML buffer type, this field contains either a routing element type (or name) or a routing element object name.

The syntax of this object for an XML buffer type is as follows:

```
root_element[/child_element][/child_element][/...][/@object_\
name]
```

The element is assumed to be an XML document or datagram element type. Indexing is not supported. Therefore, the Oracle Tuxedo system recognizes only the first occurrence of a given element type when it processes an XML buffer for data-dependent routing. This information is used to get the associated element content for data-dependent routing while sending a message. The content must be a string encoded in UTF-8.

The object is assumed to be an XML document or datagram object of the defined element. This information is used to get the associated object value for data-dependent routing while sending a message. The value must be a string encoded in UTF-8.

The combination of element name and object name can contain up to 30 characters.

## tuxDmRoutingDmFieldType

### Syntax

```
INTEGER { char(1) | short(2) | long(3) | float(4) | double(5) | string(6)}
```

### Access

read-write

### Description

The type can be `char`, `short`, `long`, `float`, `double`, or `string`. Only one type is allowed. This object value is used only for routing XML buffers.

## tuxDmRoutingDmRanges

### Syntax

```
DisplayString(SIZE(1..1000))
```

### Access

read-write

## Description

This object includes the ranges and associated server groups for the `tuxDmRoutingFieldType` routing field. The format of the string is a comma-separated, ordered list of range/group name pairs. A range/group pair has the following format:

```
lower[-upper]:raccesspoint
```

where `lower` and `upper` are assigned numeric values or character strings in single quotes. `lower` must be less than or equal to `upper`. To embed a single quote in a character string value, the quote must be preceded by two backslashes (for example, `'O\\'Brien'`). The value `MIN` can be used to indicate the minimum value for the data type of the associated field on the machine. The value `MAX` can be used to indicate the maximum value for the data type of the associated field on the machine. Thus, `"MIN-5"` is all numbers less than or equal to -5, and `"6-MAX"` is all numbers greater than or equal to 6.

The meta-character `"*"` (wild-card) in the position of a range indicates any values not covered by the other ranges previously seen in the entry. Only one wild-card range is allowed per entry and it should be the last range (ranges that follow it are ignored).

The routing field can be of any data type supported in FML. A numeric routing field must have numeric range values and a string routing field must have string range values.

String range values for `string`, `carray`, and `character` field types must be placed inside a pair of single quotes, and cannot be preceded by a sign. The `short` and `long` integer values are a string of digits, optionally preceded by a plus or minus sign. Floating point numbers are of the form accepted by the C compiler or `atof(3)`: an optional sign, then a string of digits (that optionally contains a decimal point), then an optional `e` or `E` followed by an optional sign or space, followed by an integer.

The `raccesspoint` parameter indicates the remote domain access point to which the request is routed if the field matches the range. A `raccesspoint` of `"*"` indicates that the request can go to any remote domain access point that imports the desired service.

## tuxDmRoutingState

### Syntax

```
INTEGER { valid(1) | invalid(2) }
```

### Access

read-write

## Description

This object denotes the current state of the `tuxDmRouting` instance.

GET requests:

`valid(1)`: The object exists.

SET requests:

`invalid(2)`: The object is deleted.

## tuxDmrPrincipalTable

The `tuxDmrPrincipalTable` group contains objects that represent password configuration information for remote principal names.

Object Name	Object ID
<code>tuxDmrPrincipalDmrAccessPoint</code>	.1.3.6.1.4.1.140.300.190.1.1.10
<code>tuxDmrPrincipalDmrPrinName</code>	.1.3.6.1.4.1.140.300.190.1.1.20
<code>tuxDmrPrincipalDmrPrinPasswd</code>	.1.3.6.1.4.1.140.300.190.1.1.30
<code>tuxDmrPrincipalState</code>	.1.3.6.1.4.1.140.300.190.1.1.40

## tuxDmrPrincipalDmrAccessPoint

### Syntax

*DisplayString*(SIZE(1..24))

### Access

read-only

### Description

The remote domain access point to which the principal is applicable.

## tuxDmrPrincipalDmrPrinName

### Syntax

*DisplayString*(SIZE(1..24))

## Access

read-only

## Description

The remote principal name.

## **tuxDmrPrincipalDmrPrinPasswd**

## Syntax

*DisplayString* (SIZE(1..8))

## Access

write-only

## Description

The remote password used for the principal name when communicating through the remote domain access point identified in `tuxDmrPrincipalDmrAccessPoint`.

## **tuxDmrPrincipalState**

## Syntax

INTEGER { valid(1) | invalid(2) }

## Access

read-write

## Description

This object denotes the current state of the `tuxDmrPrincipal` instance.

GET requests:

valid(1): The object exists.

SET requests:

invalid(2): The object is deleted.

## tuxDmSnaCRMTTable

The tuxDmSnaCRMTTable group defines the SNM-CRM-specific configuration information for the named local domain access point.

Object Name	Object ID
tuxDmSnaCRMDmSNACRM	.1.3.6.1.4.1.140.300.200.1.1.10
tuxDmSnaCRMDmAccessPoint	.1.3.6.1.4.1.140.300.200.1.1.20
tuxDmSnaCRMState	.1.3.6.1.4.1.140.300.200.1.1.30
tuxDmSnaCRMDmNWAddr	.1.3.6.1.4.1.140.300.200.1.1.40
tuxDmSnaCRMDmNWDevice	.1.3.6.1.4.1.140.300.200.1.1.50

### tuxDMSnaCRMDmSNACRM

#### Syntax

*DisplayString*(SIZE(1..30))

#### Access

read-only

#### Description

This object is an identifier, unique within the scope of the SNA CRM entries in the domain configuration, used to identify this SNA CRM entry.

### tuxDMSnaCRMDmAccessPoint

#### Syntax

*DisplayString*(SIZE(1..30))

#### Access

write-only

## Description

The name of the local domain access point entry with which this SNA CRM is used.

## **tuxDMSnaCRMState**

### Syntax

```
INTEGER { valid(1) | invalid(2) }
```

### Access

read-write

### Description

This object denotes the current state of the `tuxDmSnaCRM` instance.

GET requests:

`valid(1)`: The object exists.

SET requests:

`invalid(2)`: The object is deleted.

## **tuxDMSnaCRMDmNWAddr**

### Syntax

*DisplayString*(SIZE(1..256)) (up to 78 bytes for Oracle Tuxedo 8.0 or earlier)

### Access

read-write

### Description

Specifies the network address for communication between the domain gateway of the local domain access point and the SNA CRM.

## **tuxDMSnaCRMDmNWDevice**

### Syntax

*DisplayString*(SIZE(1..78))

## Access

read-write

## Description

Specifies the network device to be used for communication between the domain gateway of the local domain access point and the SNA CRM.

# tuxDmSnaLinkTable

The tuxDmSnaLinkTable group contains objects that represent snax-specific configuration information for a remote domain access point.

Object Name	Object ID
tuxDmSnaLinkDmSNALink	.1.3.6.1.4.1.140.300.210.1.1.10
tuxDmSnaLinkDmSNASStack	.1.3.6.1.4.1.140.300.210.1.1.20
tuxDmSnaLinkDmrAccessPoint	.1.3.6.1.4.1.140.300.210.1.1.30
tuxDmSnaLinkDmlSysID	.1.3.6.1.4.1.140.300.210.1.1.40
tuxDmSnaLinkDmrSysID	.1.3.6.1.4.1.140.300.210.1.1.50
tuxDmSnaLinkDmlUname	.1.3.6.1.4.1.140.300.210.1.1.60
tuxDmSnaLinkDmMinWin	.1.3.6.1.4.1.140.300.210.1.1.70
tuxDmSnaLinkDmModeName	.1.3.6.1.4.1.140.300.210.1.1.80
tuxDmSnaLinkState	.1.3.6.1.4.1.140.300.210.1.1.90
tuxDmSnaLinkDmSecType	.1.3.6.1.4.1.140.300.210.1.1.100
tuxDmSnaLinkDmStartType	.1.3.6.1.4.1.140.300.210.1.1.110
tuxDmSnaLinkDmMaxSNAsess	.1.3.6.1.4.1.140.300.210.1.1.120
tuxDmSnaLinkDmMaxSyncLvl	.1.3.6.1.4.1.140.300.210.1.1.130

## **tuxDmSnaLinkDmSNALink**

### Syntax

*DisplayString* (SIZE(1..30))

### Access

read-only

### Description

This object is an identifier, unique within the scope of the SNA LINK entries within the domain configuration, used to identify rows in this table.

## **tuxDmSnaLinkDmSNASStack**

### Syntax

*DisplayString* (SIZE(1..30))

### Access

read-write

### Description

The name of the *snaX* stack entry to be used to reach this remote domain access point.

## **tuxDmSnaLinkDmrAccessPoint**

### Syntax

*DisplayString* (SIZE(1..30))

### Access

read-write

### Description

Identifies the remote domain access point name for which this entry provides the *snaX* configuration data.

## **tuxDmSnaLinkDmlSysID**

### Syntax

*DisplayString*(SIZE(1..4))

### Access

read-write

### Description

The local SYSID used when establishing an SNA link to the remote logical unit (LU).

## **tuxDmSnaLinkDmrSysID**

### Syntax

*DisplayString*(SIZE(1..4))

### Access

read-write

### Description

The remote SYSID used when establishing an SNA link to the remote logical unit (LU).

## **tuxDmSnaLinkDmlUname**

### Syntax

*DisplayString*(SIZE(1..8))

### Access

read-write

### Description

Specifies the logical unit (LU) name associated with the remote domain access point.

## **tuxDmSnaLinkDmMinWin**

### Syntax

INTEGER

### Access

read-write

### Description

The minimum number of winner sessions to the remote LU.

## **tuxDmSnaLinkDmModeName**

### Syntax

*DisplayString*(SIZE(1..8))

### Access

read-write

### Description

Specifies the name associated with the session characteristics for sessions to the remote LU.

## **tuxDmSnaLinkState**

### Syntax

INTEGER { valid(1) | invalid(2) }

### Access

read-write

### Description

This object denotes the current state of the `tuxDmSnaLink` instance.

GET requests:

valid(1): The object exists.

SET requests:

invalid(2): The object is deleted.

## tuxDmSnaLinkDmSecType

### Syntax

```
INTEGER { local(1) | identify(2) | verify(3) | persistent(4) | mixidpe(5) }
```

### Access

read-write

### Description

Specifies the type of SNA security to be used on sessions to the remote logical unit. Valid values for this object are `local`, `identify`, `verify`, `persistent`, and `mixidpe`.

## tuxDmSnaLinkDmStartType

### Syntax

```
INTEGER { auto(1) | cold(2) }
```

### Access

read-write

### Description

Specifies the type of session start-up for the destination logical unit (LU).

`auto(1)`

The SNACRM, in conjunction with the domain gateway, chooses whether to `COLDSTART` or `WARMSTART` the LU.

`cold(2)`

Forces a `COLDSTART` with the LU.

## tuxDmSnaLinkDmSNAseSS

### Syntax

```
INTEGER (0..32767)
```

### Access

read-write

## Description

Specifies maximum number of sessions to establish with the remote LU.

## tuxDmSnaLinkDmMaxSyncLvl

### Syntax

INTEGER (0..2)

### Access

read-only

## Description

The maximum SYNC LEVEL that can be support to this remote LU.

## tuxDmSnaStackTable

The `tuxDmSnaStackTable` group defines an SNA stack to be used by a specific SNA CRM.

Object Name	Object ID
tuxDmSnaStackDmSnaStack	.1.3.6.1.4.1.140.300.220.1.1.10
tuxDmSnaStackDmSnaCRM	.1.3.6.1.4.1.140.300.220.1.1.20
tuxDmSnaStackDmStackType	.1.3.6.1.4.1.140.300.220.1.1.30
tuxDmSnaStackDmLName	.1.3.6.1.4.1.140.300.220.1.1.40
tuxDmSnaStackDmTpName	.1.3.6.1.4.1.140.300.220.1.1.50
tuxDmSnaStackDmStackParams	.1.3.6.1.4.1.140.300.220.1.1.60
tuxDmSnaStackState	.1.3.6.1.4.1.140.300.220.1.1.70

## tuxDmSnaStackDmSnaStack

### Syntax

*DisplayString*(SIZE(1..30))

## Access

read-only

## Description

The name of this `tuxDmSnaStack` entry. This object is an identifier, unique within the scope of the `tuxDmSnaStackTable` in the domain configuration.

## **tuxDmSnaStackDmSnaCRM**

### Syntax

*DisplayString* (SIZE(1..30))

### Access

read-write

### Description

Identifies the `tuxDmSnaCRM` table entry of the SNA CRM in which this SNA protocol stack definition is used.

## **tuxDmSnaStackDmStackType**

### Syntax

*DisplayString* (SIZE(1..30))

### Access

read-write

### Description

Identifies the protocol stack to be used.

## **tuxDmSnaStackDmIUname**

### Syntax

*DisplayString* (SIZE(1..8))

## Access

read-write

## Description

Specifies the LU name to be used on sessions established using this stack definition.

## **tuxDmSnaStackDmTpName**

### Syntax

*DisplayString* (SIZE(1..8))

### Access

read-write

### Description

Specifies the TP name associated with the SNA stack. A value of "\*" means accept any TP name.

## **tuxDmSnaStackDmStackParams**

### Syntax

*DisplayString* (SIZE(1..128))

### Access

read-write

### Description

Provides protocol stack specific parameters.

## **tuxDmSnaStackState**

### Syntax

INTEGER { valid(1) | invalid(2) }

### Access

read-write

## Description

This object denotes the current state of the `tuxDmSnaStack` instance.

GET requests:

`valid(1)`: The object exists.

SET requests:

`invalid(2)`: The object is deleted.

## tuxDmTdomainTable

The `tuxDmTdomainTable` group defines the domain-specific configuration for a local or remote domain access point.

Object Name	Object ID
<code>tuxDmTdomainDmAccessPoint</code>	<code>.1.3.6.1.4.1.140.300.240.1.1.10</code>
<code>tuxDmTdomainDmNwAddr</code>	<code>.1.3.6.1.4.1.140.300.240.1.1.20</code>
<code>tuxDmTdomainState</code>	<code>.1.3.6.1.4.1.140.300.240.1.1.30</code>
<code>tuxDmTdomainDmNwDevice</code>	<code>.1.3.6.1.4.1.140.300.240.1.1.40</code>
<code>tuxDmTdomainDmCmpLimit</code>	<code>.1.3.6.1.4.1.140.300.240.1.1.50</code>
<code>tuxDmTdomainDmFailOverSeq</code>	<code>.1.3.6.1.4.1.140.300.240.1.1.60</code>
<code>tuxDmTdomainDmMinEncryptBits</code>	<code>.1.3.6.1.4.1.140.300.240.1.1.70</code>
<code>tuxDmTdomainDmMaxEncryptBits</code>	<code>.1.3.6.1.4.1.140.300.240.1.1.80</code>

## tuxDmTdomainDmAccessPoint

### Syntax

*DisplayString*(SIZE(1..24))

### Access

read-only

## Description

The local or remote domain access point name for which this entry provides the TDomain-specific configuration data.

When domain-level failover is in use, more than one `tuxDmTdomainTable` entry can be defined with the same `tuxTdmTdomainDmAccessPoint`.

## **tuxDmTdomainDmNwAddr**

### Syntax

*DisplayString*(SIZE(1..24))

### Access

read-only

### Description

Specifies the network address associated with the access point.

For a local domain access point, this object supplies the address used to listen for incoming connections.

For a remote domain access point, this object supplies the destination used when you connect to a remote domain access point.

The value of this object must be unique across all `tuxDmTdomainTable` objects.

## **tuxDmTdomainState**

### Syntax

INTEGER { valid(1) | invalid(2) }

### Access

read-write

### Description

This object denotes the current state of the `tuxDmTdomain` instance.

GET requests:

`valid(1)`: The object exists.

SET requests:  
    invalid(2): Delete the object.

## **tuxDmTdomainDmNwDevice**

### **Syntax**

*DisplayString*(SIZE(1..78))

### **Access**

read-write

### **Description**

Specifies the network device used.

For a local domain access point, this object specifies the device used for listening.

For a remote domain access point, this object specifies the device used to connect to the remote domain access point.

## **tuxDmTdomainDmCmpLimit**

### **Syntax**

INTEGER

### **Access**

read-write

### **Description**

This object is relevant to remote domain access points only. It specifies the threshold over which compression occurs for traffic on connections to this access point.

## **tuxDmTdomainDmFailOverSeq**

### **Syntax**

INTEGER (0..32767)

## Access

read-write

## Description

This object is relevant to remote domain access points only. It specifies the position of this set of addressing in the failover sequence for this remote domain access point. If no failover sequence number is supplied, the first entry for this remote domain access point is allocated the number 10 greater than the highest failover sequence number known for the remote domain access point. Thus, the first entry gets 10, the second, 20, and so on.

## tuxDmTdomainDmMinEncryptBits

### Syntax

```
INTEGER { enc-0-bit(1) | enc-40-bits(2) | enc-56-bits(3) | enc-128-bits(4) }
```

## Access

read-write

## Description

This object is relevant to remote domain access points only. When establishing a network link to this access point, this object specifies the minimum level of encryption required.

enc-0-bit(1)

No encryption

enc-40-bits(2), enc-56-bits(3), **and** enc-128-bits(4)

These values specify the encryption length (in bits).

If this minimum level of encryption is not met, link establishment fails. The default value is enc-0-bit.

## tuxDmTdomainDmMaxEncryptBits

### Syntax

```
INTEGER { enc-0-bit(1) | enc-40-bits(2) | enc-56-bits(3) | enc-128-bits(4) }
```

## Access

read-write

## Description

This object is relevant to remote domain access points only. When establishing a network link to this access point, this object specifies the maximum level of encryption required.

enc-0-bit(1)

No encryption

enc-40-bits(2), enc-56-bits(3), **and** enc-128-bits(4)

These values specify the encryption length (in bits).

The default value is enc-128-bits.

**Note:** Modifications to this object do not affect established connections.

## tuxDmTransactionTable

The `tuxDmTransactionTable` group contains objects that represent information about transactions that span domains. This object can be used to find out what remote domain access points are involved in the transaction, the parent domain access point, the transaction state, and other information.

Object Name	Object ID
tuxDmTransactionDmlAccessPoint	.1.3.6.1.4.1.140.300.260.1.1.10
tuxDmTransactionDmTpTranID	.1.3.6.1.4.1.140.300.260.1.1.20
tuxDmTransactionState	.1.3.6.1.4.1.140.300.260.1.1.30
tuxDmTransactionDmTxAccessPoint	.1.3.6.1.4.1.140.300.260.1.1.40
tuxDmTransactionDmTxNetTranID	.1.3.6.1.4.1.140.300.260.1.1.50
tuxDmTransactionDmBranchCount	.1.3.6.1.4.1.140.300.260.1.1.60
tuxDmTransactionDmBranchIndex	.1.3.6.1.4.1.140.300.260.1.1.70
tuxDmTransactionDmBranchNo	.1.3.6.1.4.1.140.300.260.1.1.80
tuxDmTransactionDmrAccessPoint	.1.3.6.1.4.1.140.300.260.1.1.90
tuxDmTransactionDmNetTranID	.1.3.6.1.4.1.140.300.260.1.1.100
tuxDmTransactionDmBranchState	.1.3.6.1.4.1.140.300.260.1.1.110

## **tuxDmTransactionDmlAccessPoint**

### Syntax

*DisplayString*(SIZE(1..30))

### Access

read-only

### Description

The name of the local domain access point with which the transaction is associated. This object is a required field for GET operations. For SET operations, this object must be specified.

## **tuxDmTransactionDmTPTranID**

### Syntax

*DisplayString*(SIZE(1..24))

### Access

read-write

### Description

The transaction identifier returned from `tpsuspend(3c)` mapped to a string representation. The data in this field should not be interpreted directly by the user, except for equality comparison. For SET operations, this object must be specified.

## **tuxDmTransactionState**

### Syntax

```
INTEGER { aborted(1) | abortonly(2) | active(3) | comcalled(4) | decided(5)
| done(6) | habort(7) | hcommit(8) | heuristic(9) | ready(10) | unknown(11)
| invalid(12) }
```

### Access

read-write

## Description

This object denotes the current state of the `tuxDmTransaction` instance.

GET requests:

`aborted(1)`: The transaction is being rolled back.

`abortonly(2)`: The transaction has been identified for rollback.

`active(3)`: The transaction is active.

`comcalled(4)`: The transaction has initiated the first phase of commitment.

`decided(5)`: The transaction has initiated the second phase of commitment.

`done(6)`: The transaction has completed the second phase of commitment.

`habort(7)`: The transaction has been heuristically rolled back.

`hcommit(8)`: The transaction has been heuristically committed.

`heuristic(9)`: The transaction commitment or rollback has completed heuristically.

`ready(10)`: The transaction has completed the first phase of a two-phase commit. All the participating groups and remote domains have completed the first phase of commitment and are ready to be committed.

`unknown(11)`: It was not possible to determine the state of the transaction.

SET requests:

`invalid(12)`: Forget the specified table entry. This state change is only valid in states HCOmmit and HABort.

## tuxDmTransactionDmTxAccessPoint

### Syntax

*DisplayString*(SIZE(1..30))

### Access

read-only

## Description

If the transaction originated from a remote domain, this object value is the name of the remote domain access point through which the transaction originated. If the transaction originated within this domain, this value is the name of the local domain access point.

## **tuxDmTransactionDmTxNetTranID**

### Syntax

*DisplayString*(SIZE(1..78))

### Access

read-only

## Description

If the transaction originated from a remote domain, this object value is the external transaction identifier received from the remote domain access point through which the transaction originated. If the transaction originated within this domain, the object contains the same value as the

## **tuxDmTransactionDmBranchCount**

### Syntax

INTEGER

### Access

read-only

## Description

The number of branches to remote domain access points involved in the transaction. For a domain gateway that does not make branch information available, this value is zero.

## **tuxDmTransactionDmBranchIndex**

### Syntax

INTEGER

## Access

read-only

## Description

The index of the first branch-specific object values (`tuxDmTransactionDmBranchNo`, `tuxDmTransactionDmrAccessPoint`, `tuxDmTransactionDmNetTranID`, and `tuxDmTransactionDmBranchState`) corresponding to this object.

## **tuxDmTransactionDmBranchNo**

### Syntax

INTEGER

### Access

read-only

### Description

The branch number of the participating branch (numbered from zero).

## **tuxDmTransactionDmrAccessPoint**

### Syntax

*DisplayString*(SIZE(1..30))

### Access

read-only

### Description

The name of the remote domain access point for this branch.

## **tuxDmTransactionDmNetTranID**

### Syntax

*DisplayString*(SIZE(1..78))

## Access

read-only

## Description

The external transaction identifier used with the remote domain access point for this branch. Some types of domain gateways do not return this information; in this scenario, this object is set to the empty string. For example, TDomains uses the local transaction identifier in `tuxDmTransactionDmTpTranID` for branches to remote domain access points and sets this value to the empty string.

## tuxDmTransactionDmBranchState

### Syntax

```
INTEGER { aborted(1) | abortonly(2) | active(3) | comcalled(4) | decided(5)
| done(6) | habort(7) | hcommit(8) | heuristic-hazard(9) |
heuristic-mixed(10) | ready(11) | unknown(12) }
```

## Access

read-write

## Description

A GET operation retrieves run-time information for the transaction branch (when the information is available for a particular domain gateway type).

GET requests:

**ABorted(1):** The transaction branch is being rolled back.

**ABortonly(2):** The transaction branch has been identified for rollback.

**ACTive(3):** The transaction branch is active.

**COMcalled(4):** The transaction branch has initiated the first phase of commitment.

**DECided(5):** The transaction branch has initiated second phase of commitment.

**DONe(6):** The transaction branch has completed the second phase of commitment.

**HABort(7):** The transaction has been heuristically rolled back.

**HCOMmit(8):** The transaction has been heuristically committed.

Heuristic HaZard(9): Communications for the transaction branch failed and it has not been determined if rollback completed successfully.

Heuristic MIxed(10): The transaction commitment or rollback for the transaction branch has completed and the remote domain has reported that the state of some of the resources used for the commitment or rollback is not consistent with the outcome of the transaction.

REAdy 11): The transaction has completed the first phase of a two-phase commit. All the participating groups and remote domains have completed the first phase of commitment and are ready to be committed.

UNKnown(12): The state of the transaction could not be determined.



# Oracle Domain List MIB

The Oracle Domain List MIB consists of one group name `beaDomainList`. This group contains objects that represent the information about the Oracle Tuxedo domain that the Tuxedo SNMP agent is monitoring, as specified at startup. Note that row creation is not allowed in this MIB group, and that a minimal `tuxconfig` file must exist before you can start the agent.

The `beaDomainList` MIB group consists of the following objects.

Object Name	Object ID
<code>beaDomainKey</code>	.1.3.6.1.4.1.140.305.1.1
<code>beaLogicalAgentName</code>	.1.3.6.1.4.1.140.305.1.2
<code>beaDomainId</code>	.1.3.6.1.4.1.140.305.1.3
<code>beaDomainTuxdir</code>	.1.3.6.1.4.1.140.305.1.4
<code>beaDomainTuxconfig</code>	.1.3.6.1.4.1.140.305.1.5
<code>beaDomainStatus</code>	.1.3.6.1.4.1.140.305.1.6

## `beaDomainKey`

### Syntax

INTEGER (32769..262143)

## Access

read-only

## Description

Numeric key for the well-known address in a Tuxedo system bulletin board. In a single-processor environment, this key names the bulletin board. In a multi-processor environment, this key names the message queue of the Distinguished Bulletin Board Liaison (DBBL). This key is used as the basis for deriving the names of resources other than the well-known address, such as the names for the bulletin boards throughout the application.

## **beaLogicalAgentName**

### Syntax

*DisplayString*(SIZE(1..32))

### Access

read-only

### Description

The logical agent name of the agent as specified in the `-l` option when the Tuxedo SNMP agent was started (UNIX systems). On Windows NT systems, the logical agent name is the name of the Windows NT service used to start the agent. This name is the name of the agent that monitors the domain. If there are multiple SNMP agents running on a managed node, this name needs to be appended to the community string with an `@` sign when sending an SNMP request to the agent. For example, if there are two logical agents `simp_snmpd` and `bank_snmpd`, the default communities used to query values from these agents would be `public@simp_snmpd` and `public@bank_snmpd`, respectively. To run multiple agents on the same managed node, they must be run as subagents (without the `-s` option) with the Oracle SNMP Agent Integrator.

## **beaDomainId**

### Syntax

*DisplayString*(SIZE(1..30))

### Access

read-only

## Description

This object is the Oracle domain identifier of the domain being managed by this agent. This object is optional.

## **beaDomainTuxdir**

### Syntax

*DisplayString* (SIZE(1..256)) (up to 78 bytes for Oracle Tuxedo 8.0 or earlier)

### Access

read-only

### Description

The *tuxdir* value for the domain being managed by this agent. *tuxdir* is the absolute pathname to the directory where the Tuxedo software is found on the master machine.

## **beaDomainTuxconfig**

### Syntax

*DisplayString* (SIZE(1..256)) (up to 64 bytes for Oracle Tuxedo 8.0 or earlier)

### Access

read-only

### Description

The absolute location, including filename, for the configuration file of the domain being managed by this agent.

## **beaDomainStatus**

### Syntax

INTEGER { active(1), inactive(2) }

### Access

read-only

## Description

This object represents the current state of the domain being managed. The states and their interpretation are the same as for `tuxTdomainState`.

# CORBA Interface MIB

The CORBA Interface MIB defines the Common Object Request Broker Architecture (CORBA) interface groups and objects specific to Oracle Tuxedo 8.0 or later applications. The CORBA Interface MIB consists of the following groups:

Group Name	Description
<code>tuxFactoryTable</code>	This group contains objects that represent occurrences of factories registered with the FactoryFinder. The available factories for the Tuxedo 8.0 or later application are reflected in this MIB group.
<code>tuxInterfaceTable</code>	This group contains objects that represent the configuration and run-time characteristics of CORBA interfaces at both the domain and server group levels.
<code>tuxLclInterfaceTable</code>	The object instances in this group return local <code>tuxInterfaceTable</code> objects for the local host on which Oracle SNMP Agent is running.

Group Name	Description
<code>tuxIfQueueTable</code>	The object instances in this group represent the run-time characteristics of an interface for a particular server queue ( <code>tuxTQueue</code> ).
<code>tuxLclIfQueueTable</code>	The object instances in this group represent the local objects of <code>tuxIfQueueTable</code> instances. These values are specific to the host on which Oracle SNMP Agent is running.

In addition to the objects in these groups, the “[Core MIB](#)” on page 2-1 contains the following CORBA interface specific objects:

- `tuxMaxObjects`
- `tuxMaxInterfaces`
- `tuxCurInterfaces`
- `tuxHwInterfaces`
- `tuxMachineMaxObjects`
- `tuxMachineCurObjects`
- `tuxMachineHwObjects`
- `tuxSrvrCurObjsExt`
- `tuxSrvrCurInterfaceExt`
- `tuxTranGstate`

For more information on these objects, see “[Core MIB](#)” on page 2-1.

## tuxFactoryTable

The `tuxFactoryTable` group contains objects that represent occurrences of factories registered with the `FactoryFinder`.

Object Name	Object ID
<code>tuxFactorySerNo</code>	.1.3.6.1.4.1.140.300.48.1.1
<code>tuxFactoryId</code>	.1.3.6.1.4.1.140.300.48.1.2

Object Name	Object ID
tuxFactoryIfName	.1.3.6.1.4.1.140.300.48.1.3
tuxFactoryState	.1.3.6.1.4.1.140.300.48.1.4

## tuxFactorySerNo

### Syntax

INTEGER

### Access

read-only

### Description

This object is the running number used as the index to instances in this table.

## tuxFactoryId

### Syntax

*DisplayString* (SIZE(1..256))

### Access

read-only

### Description

The registered ID for the factory.

## tuxFactoryIfName

### Syntax

*DisplayString* (SIZE(1..128))

### Access

read-only

## Description

The fully qualified interface name used as the interface repository ID for the factory. The format of this name depends on the options specified in the Interface Definition Language (IDL) that generates the interface implementation. For details, consult the CORBA 2.1 specification, section 7.6.

## tuxFactoryState

### Syntax

```
INTEGER { active(1) }
```

### Access

read-only

### Description

A GET operation retrieves run-time information for the selected `tuxFactoryTable` instance or instances. The returned value is 1 (`active`) if the instance is registered with the `FactoryFinder`.

## tuxInterfaceTable

The `tuxInterfaceTable` group contains objects that represent configuration and run-time characteristics of CORBA interfaces at both the domain and server group levels. There are certain semantic differences in the objects of this group between domain level and server group level instances, as explained in the following discussions for `tuxInterfaceTable`.

### tuxInterfaceTable

A domain level `tuxInterfaceTable` instance is not associated with a Server group. In this case, its `tuxIfSrvGrp` object has the invalid value \*.

A server group level instance has an associated Server group. In this case, its `tuxIfSrvGrp` object has a valid server group name for the domain. This server group level representation of an interface also provides a container for managing the interface state (the `tuxIfState` object) and for collecting accumulated statistics.

Every CORBA interface that is activated in a server must have a server group level `tuxInterfaceTable` instance. The activation of interfaces in a server is controlled by the state of a `tuxIfQueue` instance for the interface. Activation of a `tuxIfQueue` instance causes its

objects to be initialized with values specified for the associated server group level `tuxInterfaceTable` instance. If such an instance does not already exist, then one is dynamically created. This dynamically created server group level `tuxInterfaceTable` instance is initialized with the objects of the domain level `tuxInterfaceTable` instance for the interface, if one exists. If an associated domain level instance does not exist, system-specified default configuration values are used. After they are activated, interfaces are always associated with a server group level `tuxInterfaceTable` instance.

The specification of configuration objects for interfaces at any level is optional. Interfaces offered by a server are identified through the ICF file used for generating skeletons. The interfaces are advertised automatically by the system when the server is activated.

## tuxInterfaceTable Objects

The following table lists the objects within the `tuxInterfaceTable`.

Object Name	Object ID
<code>tuxIfSerNo</code>	.1.3.6.1.4.1.140.300.53.1.1.1
<code>tuxIfName</code>	.1.3.6.1.4.1.140.300.53.1.1.2
<code>tuxIfSrvGrp</code>	.1.3.6.1.4.1.140.300.53.1.1.3
<code>tuxIfState</code>	.1.3.6.1.4.1.140.300.53.1.1.4
<code>tuxIfAutoTran</code>	.1.3.6.1.4.1.140.300.53.1.1.5
<code>tuxIfLoad</code>	.1.3.6.1.4.1.140.300.53.1.1.6
<code>tuxIfPrio</code>	.1.3.6.1.4.1.140.300.53.1.1.7
<code>tuxIfTimeout</code>	.1.3.6.1.4.1.140.300.53.1.1.8
<code>tuxIfTranTime</code>	.1.3.6.1.4.1.140.300.53.1.1.9
<code>tuxIfFbRoutingName</code>	.1.3.6.1.4.1.140.300.53.1.1.10
<code>tuxIfLmid</code>	.1.3.6.1.4.1.140.300.53.1.1.11
<code>tuxIfNumServers</code>	.1.3.6.1.4.1.140.300.53.1.1.12

Object Name	Object ID
tuxIfTpPolicy	.1.3.6.1.4.1.140.300.53.1.1.13
tuxIfTxPolicy	.1.3.6.1.4.1.140.300.53.1.1.14

## tuxIfSerNo

### Syntax

INTEGER

### Access

read-only

### Description

This object value is the running number used as an index to instances in this table.

## tuxIfName

### Syntax

*DisplayString*(SIZE(1..128))

### Access

read-only

### Description

The fully qualified interface name used as the interface ID. The format of this name is one of the options specified in the IDL that generates the interface implementation. For details, consult the CORBA 2.1 specification, Section 7.6.

## tuxIfSrvGrp

### Syntax

*DisplayString*(SIZE(1..30))

## Access

read-write

## Description

The server group name. Server group names cannot contain an asterisk, comma, or colon. An asterisk (\*) specified as a value for this object specifies a domain level instance.

**Note:** This object can be SET only during creation of a new row.

## tuxIfState

### Syntax

```
INTEGER { active(1), inactive(2), suspended(3), partitioned(4), invalid(5),
reactivate(6) }
```

## Access

read-write

## Description

The semantics for GET and SET requests differ between server group and domain level instances as noted in the following list.

```
GET: { active(1) | inactive(2) | suspended(3) | partitioned(4) }
```

A GET request retrieves configuration information for the selected tuxInterfaceTable instance or instances. The only states that can be returned are: active, inactive, suspended, partitioned.

```
active(1)
```

The tuxInterfaceTable instance is defined and at least one corresponding tuxIfQueueTable instance is in the active state. For a server group level tuxInterfaceTable instance, corresponding tuxIfQueueTable instances are those with matching tuxIfName and tuxIfSrvGrp objects. For a domain level tuxInterfaceTable instance, corresponding tuxIfQueueTable instances are those with matching tuxIfName value regardless of their tuxIfSrvGrp value.

```
inactive(2)
```

The tuxInterfaceTable instance is defined and there are no corresponding tuxIfQueueTable instances in any active state.

suspended(3)

The `tuxInterfaceTable` instance is defined and among all corresponding `tuxIfQueueTable` instances, there are none in the active state and at least one in the suspended state.

partitioned(4)

The `tuxInterfaceTable` instance is defined and among all the corresponding `tuxIfQueueTable` instances, there are none in the active state, none in the suspended state, and at least one in the partitioned state.

SET: {invalid(5) | active(1) | inactive(2) | reactivate(6) | suspended(3) }

A SET request updates run-time and configuration information for the selected `tuxInterfaceTable` instance. Modifications can affect more than one server group when domain level changes are made, and run-time modifications can affect more than one server if multiple servers are currently offering an interface. Only the following values can be used in a SET request: `invalid`, `active`, `reactivate`, or `suspended`.

invalid(5)

Delete the `tuxInterfaceTable` instance for the application. This state change is allowed only when the instance is in the inactive state.

active(1)

Activate the `tuxInterfaceTable` instance for the application. Setting this state on a domain level instance has the effect of activating all corresponding `tuxIfQueueTable` instances that are currently suspended throughout the domain. Setting this state on a server group level instance affects only servers within the group offering the interface. This state change is allowed only when the instance is in the suspended state. A successful return leaves the object in the `active(1)` state.

reactivate(6)

Reactivates the `tuxInterfaceTable` instance. Setting this state on a domain level instance has the effect of activating all corresponding `tuxIfQueueTable` instances that are currently suspended throughout the domain. Setting this state on a server group level instance affects only servers within the group offering the interface. This state change is allowed only when the instance is in the `active(1)` or `suspended(3)` state. Successful return leaves the instance in the `active(1)` state. Setting this state permits global activation of `tuxIfQueueTable` instances suspended at the server group level without having to individually activate each server group level `tuxInterfaceTable` instance.

suspended(3)

Suspend the `tuxInterfaceTable` instance. Setting this state on the domain level object has the effect of suspending all corresponding `tuxIfQueueTable` instances that are currently active throughout the domain. Setting this state on a server group level instance affects only servers within the group offering the interface. This state change is permitted

only in the `active(1)` state. Successful return leaves the object in the `suspended(3)` state.

**Note:** Dynamic advertisement of interfaces (that is, state change from `inactive(2)` or `invalid(5)` to `active(1)`) is not supported, nor is removal of advertisement (that is, state change from `active(1)` to `inactive(2)`).

## tuxIfAutoTran

### Syntax

```
INTEGER { yes(1), no(2) }
```

### Access

read-write

### Description

Signifies whether a transaction is automatically started for invocations made outside a transaction context.

This object has the following limitations:

- Run-time updates to this object are not reflected in active equivalent `tuxInterfaceTable` instances.
- The `tuxIfTxPolicy` object can override the value specified for this object in the `UBBCONFIG` file. If `tuxIfTxPolicy` is `always(1)`, a `tuxIfAutoTran` value of `no(2)` has no effect at run-time. Behavior is as though the setting were `yes(1)`. If `tuxIfTxPolicy` is `never(2)`, a `tuxIfAutoTran` value of `yes(1)` has no effect. The interface is never involved in a transaction. If `tuxIfTxPolicy` is `ignore(4)`, a `tuxIfAutoTran` value of `yes(1)` has no effect. The interface is never involved in a transaction.

## tuxIfLoad

### Syntax

```
INTEGER (1..32767)
```

### Access

read-write

## Description

This object imposes the indicated load on the system. Interface loads are used for load-balancing. That is, queues with higher enqueued workloads are less likely to be chosen for a new request.

**Note:** Run-time updates to this object for domain level instances do not affect corresponding server group level instances for the same interface.

## tuxIfPrio

### Syntax

INTEGER (1..100)

### Access

read-write

## Description

Dequeuing priority. If multiple interface requests are waiting on a queue for servicing, the higher priority requests are handled first.

**Note:** Run-time updates to this object for domain level instances do not affect corresponding server group level instances for the same interface.

## tuxIfTimeout

### Syntax

INTEGER

### Access

read-write

## Description

The time limit (in seconds) for processing individual method invocations for this interface. Servers that process method invocations for this interface are terminated abortively if they exceed the specified time limit in processing the request. A value of 0 for this object indicates that the server should not be terminated abortively.

**Note:** Run-time updates to this object for domain level instances do not affect corresponding server group level instances for the same interface.

## tuxIfTranTime

### Syntax

INTEGER

### Access

read-write

### Description

Transaction timeout value in seconds for transactions automatically started for this `tuxInterfaceTable` instance. Transactions are started automatically when a request not in transaction mode is received and the `tuxIfAutoTran` object value for the interface is `yes(1)`.

**Note:** Run-time updates to this object for domain level instances do not affect corresponding server group level instances for the same interface.

## tuxIfFbRoutingName

### Syntax

*DisplayString* (SIZE(1..15))

### Access

read-write

### Description

The factory-based routing criteria associated with this interface.

**Note:** This object can be set only for a domain level `tuxInterfaceTable` instance, that is, only if `tuxIfSrvGrp` is `*`.

## tuxIfLmid

### Syntax

*DisplayString* (SIZE(1..30))

### Access

read-only

## Description

Current logical machine with which the active equivalent server group level `tuxInterfaceTable` instance, is associated. This object value is `NULL` for domain level instances.

## **tuxIfNumServers**

### Syntax

```
INTEGER
```

### Access

read-only

### Description

The number of corresponding servers that offer this interface.

## **tuxIfTpPolicy**

### Syntax

```
INTEGER { method(1), transaction(2), process(3) }
```

### Access

read-only

### Description

The TP framework deactivation policy. This object value reflects the policy registered with the framework at server startup. The first server to register with the interface sets the value in `tuxInterfaceTable`. This value cannot be changed.

## **tuxIfTxPolicy**

### Syntax

```
INTEGER { always(1), never(2), optional(3), ignore(4) }
```

### Access

read-only

## Description

The transaction policy for the interface. This object value affects the `tuxIfAutoTran` object. This policy is set by the application developer and is registered when the server starts.

## tuxLclInterfaceTable

The `tuxLclInterfaceTable` group returns values for the local host on which Oracle SNMP Agent is running. The following table lists the columnar objects that comprise each row (instance) in the group.

Object Name	Object ID
<code>tuxLclIfSerNo</code>	.1.3.6.1.4.1.140.300.53.2.1.1
<code>tuxLclIfName</code>	.1.3.6.1.4.1.140.300.53.2.1.2
<code>tuxLclSrvGrp</code>	.1.3.6.1.4.1.140.300.53.2.1.3
<code>tuxLclIfNcompleted</code>	.1.3.6.1.4.1.140.300.53.2.1.4
<code>tuxLclIfNqueued</code>	.1.3.6.1.4.1.140.300.53.2.1.5

## tuxLclIfSerNo

### Syntax

INTEGER

### Access

read-only

### Description

This object value is the running number, which is used as an index into the table.

## tuxLclIfName

### Syntax

*DisplayString*(SIZE(1..128))

## Access

read-only

## Description

The fully qualified interface name used as the interface repository ID for the interface. The format of this name depends on the options specified in the IDL that generates the interface implementation. For details, see the CORBA 2.1 Specification Section 7.6 [CORBA].

## tuxLclIfSrvGrp

### Syntax

*DisplayString*(SIZE(1..30))

## Access

read-only

## Description

The server group name. Server group names cannot contain an asterisk, comma, or colon. A value of \* for this object indicates a domain level interface.

## tuxLclIfNcompleted

### Syntax

INTEGER

## Access

read-only

## Description

The number of method invocations completed for the corresponding `tuxIfQueueTable` instances since they were initially offered. The values returned are for the indicated interface on the local host where Oracle SNMP Agent is running.

**Note:** This object value is returned only when `tuxTdomainLoadBalance` is `yes(1)`.

## tuxLclIfNqueued

### Syntax

INTEGER

### Access

read-only

### Description

The number of requests currently enqueued for this interface. The values returned are for the indicated interface on the local host where Oracle SNMP Agent is running.

**Note:** This object value is returned only when `tuxTdomainLoadBalance` is `yes(1)`.

## tuxIfQueueTable

The `tuxIfQueueTable` group contains objects that represent the run-time characteristics of an interface for a particular server queue. The objects provide access to the inherited configuration characteristics of an interface as well as statistics relating to the interface on the queue. This group gives administrators finer granularity in suspending and activating interfaces. This group provides the link between the interface name and the server processes capable of processing method invocations on the interface. For example, `tuxIfQRqAddr` can be used to identify the corresponding server in the `tuxTsrvrTbl` and `tuxTsrvrTblExt` groups.

Object Name	Object ID
<code>tuxIfQueueSerNo</code>	.1.3.6.1.4.1.140.300.53.3.1.1
<code>tuxIfQueueName</code>	.1.3.6.1.4.1.140.300.53.3.1.2
<code>tuxIfQueueSrvGrp</code>	.1.3.6.1.4.1.140.300.53.3.1.3
<code>tuxIfQueueRqAddr</code>	.1.3.6.1.4.1.140.300.53.3.1.4
<code>tuxIfQueueState</code>	.1.3.6.1.4.1.140.300.53.3.1.5
<code>tuxIfQueueAutoTran</code>	.1.3.6.1.4.1.140.300.53.3.1.6
<code>tuxIfQueueLoad</code>	.1.3.6.1.4.1.140.300.53.3.1.7
<code>tuxIfQueuePrio</code>	.1.3.6.1.4.1.140.300.53.3.1.8

Object Name	Object ID
tuxIfQueueTimeout	.1.3.6.1.4.1.140.300.53.3.1.9
tuxIfQueueTranTime	.1.3.6.1.4.1.140.300.53.3.1.10
tuxIfQueueFbRoutingName	.1.3.6.1.4.1.140.300.53.3.1.11
tuxIfQueueLmid	.1.3.6.1.4.1.140.300.53.3.1.12
tuxIfQueueNumServers	.1.3.6.1.4.1.140.300.53.3.1.13
tuxIfQueueTpPolicy	.1.3.6.1.4.1.140.300.53.3.1.14
tuxIfQueueTxPolicy	.1.3.6.1.4.1.140.300.53.3.1.15

## tuxIfQueueSerNo

### Syntax

INTEGER

### Access

read-only

### Description

The running number used as an index into this table.

## tuxIfQueueName

### Syntax

*DisplayString*(SIZE(1..128))

### Access

read-only

### Description

The fully qualified interface name used as the interface repository ID for the interface. The format of this name is dependent on the options specified in the IDL that generates the interface implementation. See the CORBA 2.1 specification Section 7.6 for details.

## tuxIfQueueSrvGrp

### Syntax

*DisplayString* (SIZE(1..30))

### Access

read-only

### Description

The server group name. Server group names cannot contain an asterisk, comma, or colon.

## tuxIfQueueRqAddr

### Syntax

*DisplayString* (SIZE(1..30))

### Access

read-only

### Description

The symbolic address of the request queue for an active server offering this interface. See `tuxTsrvrRqAddr` for more information about this object.

## tuxIfQueueState

### Syntax

INTEGER { active(1), suspended(2), partitioned(3), unknown(4) }

### Access

read-write

### Description

The values for GET and SET operations are as follows:

GET: { active(1) | suspended(2) | partitioned(3) }

A GET request retrieves configuration information for the selected `tuxIfQueueTable` instances. The meaning of the possible return values are as follows:

active(1)

Represents an available interface in the running system.

suspended(2)

Represents a currently suspended interface in the running system.

partitioned(3)

Represents a currently partitioned interface in the running system.

SET: { active(1) | suspended(2) }

The values for SET are:

active(1)

Activates the `tuxIfQueueTable` instance. This state change is allowed only in the `suspended(2)` state. A successful return leaves instances in the `active(1)` state.

suspended(2)

Suspends the `tuxIfQueueTable` instance. This state change is allowed only in the `active(1)` state. A successful return leaves the object in the `suspended(2)` state.

**Note:** Dynamic advertisement of interfaces (that is, a state change from inactive or invalid to active) is not supported, nor is unadvertisement (that is, a state change from active to inactive).

## tuxIfQueueAutoTran

### Syntax

```
INTEGER { yes(1), no(2) }
```

### Access

read-only

### Description

Signifies whether a transaction is automatically started for invocations made outside a transaction context.

This object has the following limitations:

- Run-time updates to this object are not reflected in active equivalent `tuxInterfaceTable` instances.
- The `tuxIfTxPolicy` object can override the value specified for this object in the `UBBCONFIG` file. If `tuxIfTxPolicy` is always (1), an `tuxIfQueueAutoTran` value of `no(2)` has no effect at run-time. Behavior is as though the setting were `yes(1)`. If

tuxIfTxPolicy is never (2), an tuxIfQueueAutoTran value of yes (1) has no effect. The interface is never involved in a transaction. If tuxIfTxPolicy is ignore (4), an tuxIfQueueAutoTran value of yes (1) has no effect. The interface is never involved in a transaction.

## tuxIfQueueLoad

### Syntax

INTEGER (1..32767)

### Access

read-only

### Description

Load imposed on the system by this instance. Interface loads are used for load-balancing. Queues with higher enqueued workloads are less likely to be chosen for a new request.

## tuxIfQueuePrio

### Syntax

INTEGER (1..101)

### Access

read-only

### Description

Dequeuing priority. If multiple interface requests are waiting on a queue for servicing, the higher priority requests are handled first.

## tuxIfQueueTimeout

### Syntax

INTEGER

### Access

read-only

## Description

The time limit (in seconds) for processing individual method invocations for this interface. Servers processing method invocations for this interface are abortively terminated if they exceed the specified time limit in processing the request. A value of 0 for this object indicates that the server should not be abortively terminated.

## **tuxIfQueueTranTime**

### Syntax

INTEGER

### Access

read-only

## Description

The transaction timeout value in seconds for transactions automatically started for this instance. Transactions are started automatically when a request not in transaction mode is received and the `tuxIfAutoTran` object value for the interface is `yes(1)`.

## **tuxIfQueueFbRoutingName**

### Syntax

*DisplayString*(SIZE(1..15))

### Access

read-only

## Description

The factory-based routing criterion associated with this interface.

## **tuxIfQueueLmid**

### Syntax

*DisplayString*(SIZE(1..30))

## Access

read-only

## Description

The current logical machine on which this queue is offering this interface.

## **tuxIfQueueNumServers**

### Syntax

```
INTEGER
```

### Access

read-only

### Description

The number of corresponding servers that offer this interface on this queue.

## **tuxIfQueueTpPolicy**

### Syntax

```
INTEGER { method(1), transaction(2), process(3) }
```

### Access

read-only

### Description

The TP framework deactivation policy. This value reflects the policy registered with the framework at the server startup. The first server to register the interface sets the value. This value cannot be changed.

## **tuxIfQueueTxPolicy**

### Syntax

```
INTEGER { always(1), never(2), optional(3), ignore(4) }
```

## Access

read-only

## Description

The transaction policy for the interface. This object value affects the effect of the `tuxIfQueueAutoTran` object; see `tuxIfQueueAutoTran` for further explanation. This object value is always read-only, and is set by the developer when the server is built and registered at server startup.

# tuxLclIfQueueTable

The `tuxLclIfQueueTable` group contains objects that represent the local characteristics of the `tuxIfQueueTable`. The object values are specific to the host on which Oracle SNMP Agent is running.

Object Name	Object ID
<code>tuxLclIfQueueSerNo</code>	.1.3.6.1.4.1.140.300.53.4.1.1
<code>tuxLclIfQueueName</code>	.1.3.6.1.4.1.140.300.53.4.1.2
<code>tuxLclIfQueueSrvGrp</code>	.1.3.6.1.4.1.140.300.53.4.1.3
<code>tuxLclIfQueueRqAddr</code>	.1.3.6.1.4.1.140.300.53.4.1.4
<code>tuxLclIfQueueNcompleted</code>	.1.3.6.1.4.1.140.300.53.4.1.5
<code>tuxLclIfQueueNqueued</code>	.1.3.6.1.4.1.140.300.53.4.1.6
<code>tuxLclIfQueueCurObjs</code>	.1.3.6.1.4.1.140.300.53.4.1.7
<code>tuxLclIfQueueCurTrans</code>	.1.3.6.1.4.1.140.300.53.4.1.8

## tuxLclIfQueueSerNo

### Syntax

INTEGER

### Access

read-only

## Description

The running number used as an index into this table.

## **tuxLclIfQueueName**

### Syntax

*DisplayString*(SIZE(1..128))

### Access

read-only

## Description

The fully qualified interface name used as the interface repository ID for this interface. The format of this name is dependent on the options specified in the IDL that generates the interface implementation. For details, see the CORBA 2.1 specification Section 7.6.

## **tuxLclIfQueueSrvGrp**

### Syntax

*DisplayString*(SIZE(1..30))

### Access

read-only

## Description

The server group name. Server group names cannot contain an asterisk, comma, or colon.

## **tuxLclIfQueueRqAddr**

### Syntax

*DisplayString*(SIZE(1..30))

### Access

read-only

## Description

The symbolic address of the request queue for an active server that offers this interface. See `tuxTsrvrRqAddr` for more information about this object.

## **tuxLclIfQueueNcompleted**

### Syntax

INTEGER

### Access

read-only

### Description

The number of interface method invocations completed since the interface was initially offered.

**Note:** This object value is returned only when `tuxTdomainLoadBalance` is equal to `yes(1)`.

## **tuxLclIfQueueNqueued**

### Syntax

INTEGER

### Access

read-only

### Description

The number of requests currently enqueued for this interface.

**Note:** This object value is returned only when `tuxTdomainLoadBalance` is equal to `yes(1)`.

## **tuxLclIfQueueCurObjs**

### Syntax

INTEGER

### Access

read-only

## Description

The number of active objects for this interface for the associated queue. This number represents the number of entries in the active object table for this queue on the associated machine. This number includes objects that are not in memory but were invoked within an active transaction.

## **tuxLclIfQueueCurTrans**

### Syntax

INTEGER

### Access

read-only

### Description

The number of active global transactions associated with this interface for its associated queue.



# Access Control List MIB

An access control list (ACL) specifies who and what is authorized to access Oracle Tuxedo system objects. The Access Control List MIB enables a system manager to administer Tuxedo security by authenticating users, setting permissions, and controlling access. It defines the objects controlled by the ACL facility. These MIB objects are grouped into three major categories.

The Access Control List MIB consists of the following groups.

Group Name	Description
<a href="#">tuxTAclGrpTable</a>	ACL group
<a href="#">tuxTAclPermTable</a>	ACL permissions
<a href="#">tuxTAclPrinTbl</a>	ACL principal (users or domains)

For Tuxedo security, define application security options in the Domain group. This group lets you specify a user identity and security type used by your Tuxedo application. The users and remote domains in an application that need authentication and authorization are collectively known as *principals*. The managed objects for getting or setting the values of principals are defined in the `tuxTAclPrinTbl` group. The managed objects for getting or setting the values of ACL groups are defined in the `tuxTAclGrpTable`. The Access Control List MIB, as a whole, specifies the principals and access control lists for Tuxedo applications services, application queues, and events. You can define these ACL *permissions* for service, event, and application queue names. The managed objects that enable you to do define the ACL permissions are defined in the

tuxTAc1PermTable group. All these ACL MIB groups and their objects are described in the following sections.

## tuxTAc1GrpTable

The tuxTAc1GrpTable group contains objects that represent groups of Tuxedo application users and domains. The following table lists the managed objects that are part of the tuxTAc1GrpTable group. To create a new row in the table, it is necessary to issue a SET request for a non-existing row.

Object Name	Object ID
tuxTAc1GrpName	.1.3.6.1.4.1.140.300.11.1.1.1.1
tuxTAc1GrpId	.1.3.6.1.4.1.140.300.11.1.1.1.2
tuxTAc1GrpState	.1.3.6.1.4.1.140.300.11.1.1.1.3

## tuxTAc1GrpName

### Syntax

*DisplayString* (SIZE(1..30))

### Access

read-write

### Description

Logical name of the group. A group name is a string of printable characters and cannot contain a pound sign, comma, colon, or newline.

**Note:** This object can be set only during row creation.

## tuxTAc1GrpId

### Syntax

INTEGER (0..16384)

## Access

read-write

## Description

Group identifier associated with this user. A value of 0 indicates the default group `other`. If the group identifier is not specified at creation time, it defaults to the next available (unique) identifier greater than 0.

## tuxTAclGrpState

### Syntax

```
INTEGER { valid(1), invalid(2) }
```

### Access

read-write

### Description

The values for GET and SET operations are as follows:

GET: `valid(1)`

A GET operation retrieves configuration information for the selected `tuxTAclGrpTable` instance(s). The following state indicates the meaning of a `tuxTAclGrpState` returned in response to a GET request. States not listed are not returned.

`valid(1)`

`tuxTAclGrpTable` instance is defined and inactive. Note that `valid(1)` is the only valid state for this class. ACL groups are never active.

SET: `invalid(2)`

A SET operation updates configuration information for the selected `tuxTAclGrpTable` instance. The following state indicates the meaning of a `tuxTAclGrpState` set in a SET request. States not listed might not be set.

`invalid(2)`

Delete `tuxTAclGrpTable` instance for application. Successful return removes the instance from the table.

# tuxTAclPermTable

The `tuxTAclPermTable` group indicates what groups are allowed to access Tuxedo system entities. These entities are named by a string. The names currently represent service names, event names, and application queue names. To create a new row in this table, it is necessary to issue a SET request for a non-existing row that specifies at least the values for `tuxTAclPermName` and `tuxTAclPermType`.

Object Name	Object ID
<code>tuxTAclPermName</code>	.1.3.6.1.4.1.140.300.11.2.1.1.1
<code>tuxTAclPermType</code>	.1.3.6.1.4.1.140.300.11.2.1.1.2
<code>tuxTAclPermGrpIds</code>	.1.3.6.1.4.1.140.300.11.2.1.1.3
<code>tuxTAclPermState</code>	.1.3.6.1.4.1.140.300.11.2.1.1.4

## tuxTAclPermName

### Syntax

*DisplayString*(SIZE(1..30))

### Access

read-write

### Description

The name of the entity for which permissions are being granted. The name can represent a service name, an event name, and/or a queue name. An ACL name is a string of printable characters and cannot contain a colon, pound sign, or newline.

**Note:** This object can be set only during row creation.

## tuxTAclPermType

### Syntax

INTEGER { enq(1), deq(2), service(3), postevent(4) }

## Access

read-write

## Description

The type of the entity for which permissions are being granted.

**Note:** This object can be set only during row creation.

## tuxTAclPermGrpIds

### Syntax

*DisplayString* (SIZE(0..800))

## Access

read-write

## Description

A comma-separated list of group identifiers (numbers) that are permitted access to the associated entity.

## tuxTAclPermState

### Syntax

INTEGER { valid(1), invalid(2) }

## Access

read-write

## Description

The values for GET and SET operations are as follows:

GET: valid(1)

A GET operation retrieves configuration information for all selected entities. The following state indicates the meaning of a tuxTAclPermState returned in response to a GET request. States not listed are not returned.

`valid(1)`

`tuxTAclPermState` instance is defined. Note that `valid(1)` is the only valid state for this class. ACL permissions are never active.

SET: `invalid(2)`

A SET operation updates configuration information for the selected `tuxTAclPermState` instance. The following state indicates the meaning of a `tuxTAclPermState` set in a SET request. States not listed might not be set.

`invalid(2)`

Delete `tuxTAclPermState` instance for application. State change allowed only when in the `valid(1)` state. Successful return leaves the object in the `invalid(2)` state.

**Note:** The `tuxTAclPermTable` instance refers to all groupids related to a particular `tuxTAclPermName` in the table.

## tuxTAclPrinTbl

The `tuxTAclPrinTbl` group contains objects that represent users or domains that can access a Tuxedo application and the group with which they are associated. To join the application as a specific user, it is necessary to present a user-specific password. To create a new row in this table, it is necessary to issue a SET request for a non-existing row (instance).

Object Name	Object ID
<code>tuxTAclPrinName</code>	.1.3.6.1.4.1.140.300.11.3.1.1.1
<code>tuxTAclCltName</code>	.1.3.6.1.4.1.140.300.11.3.1.1.2
<code>tuxTAclPrinId</code>	.1.3.6.1.4.1.140.300.11.3.1.1.3
<code>tuxTAclPrinGrp</code>	.1.3.6.1.4.1.140.300.11.3.1.1.4
<code>tuxTAclPrinPasswd</code>	.1.3.6.1.4.1.140.300.11.3.1.1.5
<code>tuxTAclPrinState</code>	.1.3.6.1.4.1.140.300.11.3.1.1.6

## tuxTAclPrinName

### Syntax

*DisplayString(SIZE(1..30))*

## Access

read-write

## Description

Logical name of the user or domain (a principal). A principal name is a string of printable characters and cannot contain a pound sign, colon, or newline.

**Note:** This object can be set only during row creation.

## tuxTAclCltName

### Syntax

*DisplayString* (SIZE(1..30))

## Access

read-write

## Description

The client name associated with the user. It generally describes the role of the associated user and provides a further qualifier on the user entry. If the client name is not specified at creation time, the default is the wildcard asterisk (\*). A client name is a string of printable characters and cannot contain a colon or newline.

## tuxTAclPrinId

### Syntax

INTEGER (1..131072)

## Access

read-write

## Description

Unique user identification number. If not specified at creation time, it defaults to the next available (unique) identifier greater than 0.

**Note:** This object can be set only during row creation.

## **tuxTAclPrinGrp**

### Syntax

INTEGER (0..16384)

### Access

read-write

### Description

Group identifier associated with this user. A value of 0 indicates the default group `other`. If the group identifier is not specified at creation time, the default value 0 is assigned.

## **tuxTAclPrinPasswd**

### Syntax

*DisplayString*

### Access

read-write

### Description

The clear-text authentication password for the associated user. Note that the system automatically encrypts this information on behalf of the administrator.

## **tuxTAclPrinState**

### Syntax

INTEGER { valid(1), invalid(2) }

### Access

read-write

### Description

The values for `GET` and `SET` operations are as follows:

GET: valid(1)

A GET operation retrieves configuration information for the selected tuxTAclPrinTbl instance(s). The following state indicates the meaning of tuxTAclPrinState:

valid(1)

tuxTAclPrinTbl instance is defined and inactive. Note that valid(1) is the only valid state for this class. ACL principals are never active.

SET: invalid(2)

A SET operation updates configuration information for the selected tuxTAclPrinTbl instance. The following state indicates the meaning of a tuxTAclPrinState set in a SET request. States not listed might not be set.

invalid(2)

Delete tuxTAclPrinTbl instance for application. State change is allowed only when in the valid(1) state. Successful return leaves the object in the invalid(2) state.



# Workstation MIB

Oracle Tuxedo systems can require clients to run on a workstation for purposes of security, performance, and convenience. A network administrator can define the environment required to control workstation clients using the Workstation MIB. This MIB is an extension of the Core MIB and specifies the information required to control access to a Tuxedo application from multiple workstations.

The Tuxedo Workstation subsystem consists of a workstation clients (WSC) library, the workstation listener (WSL) executable, and the workstation handler (WSH) executable. The Workstation MIB specifies information about workstation listeners and workstation handlers. The following table lists the two WSL and WSH groups through which you can manage a workstation listener and its associated workstation handler processes.

The Workstation MIB consists of the following groups.

Group Name	Description
<a href="#">tuxTwshTbl</a>	Workstation Handler
<a href="#">tuxTws1Tbl</a>	Workstation Listener

You can define new workstation listeners in the `tuxTws1Tbl` group, and you can obtain information about active workstation handlers from the `tuxTwshTbl` group.

# tuxTwshTbl

The tuxTwshTbl table represents run-time characteristics of WSH client processes. These objects characterize workstation statistics specific to a particular WSH client process. Objects in this table are only accessible through a Tuxedo SNMP agent installed on the local machine. Objects are only accessible when the corresponding WSH is active.

<b>Object Name</b>	<b>Object ID</b>
tuxTwshTaClientId	.1.3.6.1.4.1.140.300.1.1.1.1
tuxTwshTaWshClientId	.1.3.6.1.4.1.140.300.1.1.1.2
tuxTwshTaSrvGrp	.1.3.6.1.4.1.140.300.1.1.1.3
tuxTwshTaSrvId	.1.3.6.1.4.1.140.300.1.1.1.4
tuxTwshTaGrpNo	.1.3.6.1.4.1.140.300.1.1.1.5
tuxTwshTaState	.1.3.6.1.4.1.140.300.1.1.1.6
tuxTwshTaLmid	.1.3.6.1.4.1.140.300.1.1.1.7
tuxTwshTaPid	.1.3.6.1.4.1.140.300.1.1.1.8
tuxTwshTaNaddr	.1.3.6.1.4.1.140.300.1.1.1.9
tuxTwshTaHwClients	.1.3.6.1.4.1.140.300.1.1.1.10
tuxTwshTaMultiplex	.1.3.6.1.4.1.140.300.1.1.1.11
tuxTwshTaCurClients	.1.3.6.1.4.1.140.300.1.1.1.12
tuxTwshTaTimeleft	.1.3.6.1.4.1.140.300.1.1.1.13
tuxTwshTaActive	.1.3.6.1.4.1.140.300.1.1.1.14
tuxTwshTaTotacttime	.1.3.6.1.4.1.140.300.1.1.1.15
tuxTwshTaTotidltime	.1.3.6.1.4.1.140.300.1.1.1.16
tuxTwshTaCurwork	.1.3.6.1.4.1.140.300.1.1.1.17
tuxTwshTaFlowcnt	.1.3.6.1.4.1.140.300.1.1.1.18
tuxTwshTaNumblockQ	.1.3.6.1.4.1.140.300.1.1.1.19

Object Name	Object ID
tuxTwshTaRcvdByt	.1.3.6.1.4.1.140.300.1.1.1.20
tuxTwshTaRcvdNum	.1.3.6.1.4.1.140.300.1.1.1.21
tuxTwshTaSentByt	.1.3.6.1.4.1.140.300.1.1.1.22
tuxTwshTaSentNum	.1.3.6.1.4.1.140.300.1.1.1.23

## tuxTwshTaClientId

### Syntax

*DisplayString*(SIZE(1..78))

### Access

read-only

### Description

Client identifier for this WSH. The data in this field should not be interpreted directly by the end user except for equality comparison.

## tuxTwshTaWshClientId

### Syntax

*DisplayString*(SIZE(1..78))

### Access

read-only

### Description

Client identifier for this WSH. The data in this field should not be interpreted directly by the end user except for equality comparison. Value is same as tuxTwshTaClientId.

## **tuxTwshTaSrvGrp**

### Syntax

*DisplayString*( SIZE (1..30) )

### Access

read-only

### Description

Logical name of the server group for the associated WSL.

## **tuxTwshTaSrvId**

### Syntax

INTEGER (1..30001)

### Access

read-only

### Description

Unique (within the server group) server identification number for the associated WSL.

## **tuxTwshTaGrpNo**

### Syntax

INTEGER (1..30000)

### Access

read-only

### Description

Group number.

## tuxTwshTaState

### Syntax

```
INTEGER { active(1), suspended(2), dead(3) }
```

### Access

read-write

### Description

State for the WSH client within the application. Any state defined for the `tuxTclientTbl` group can be returned or set. State changes to the `suspended(2)` state are transitive to all clients associated with this WSH as is the resetting of a `suspended(2)` WSH to `active(1)`.

Additionally, `suspended(2)` WSH clients are not assigned any additional incoming clients by the WSL.

Note that the state of a WSH client might not be set to `dead(3)` when accessing the `tuxTclientTbl` group. However, the state transition to `dead(3)` is allowed via the `tuxTwshTbl` group and results in all connections handled by the targeted WSH being dropped abortively.

## tuxTwshTaLmid

### Syntax

```
DisplayString( SIZE(1..30) )
```

### Access

read-only

### Description

Current logical machine on which the WSH is running.

## tuxTwshTaPid

### Syntax

```
INTEGER
```

### Access

read-only

## Description

Native operating system process identifier for the WSH client.

## **tuxTwshTaNaddr**

### Syntax

*DisplayString*( SIZE (1..78) )

### Access

read-only

## Description

Network address of workstation handler. Hexadecimal addresses are converted to an ASCII format with a leading 0x.

## **tuxTwshTaHwClients**

### Syntax

INTEGER (1..32767)

### Access

read-only

## Description

High water number of clients accessing application through this WSH.

## **tuxTwshTaMultiplex**

### Syntax

INTEGER (1..32767)

### Access

read-only

## Description

Maximum number of clients that can access the application through this WSH.

## **tuxTwshTaCurClients**

### Syntax

INTEGER (1..32767)

### Access

read-only

## Description

Current number of clients accessing the application through this WSH.

## **tuxTwshTaTimeleft**

### Syntax

INTEGER

### Access

read-only

## Description

A non-0 value for this object indicates that the WSH has been assigned a newly connecting workstation client that has the indicated amount of time, in seconds, to complete the initialization process with the WSH.

## **tuxTwshTaActive**

### Syntax

INTEGER { yes(1), no(2), unknown(3) }

### Access

read-only

## Description

A value of `yes` (1) indicates that the WSH is currently performing work on behalf of one of its associated workstation clients. A value of `no` (2) indicates that the WSH is currently waiting for work to perform on behalf of one of its associated workstation clients.

## **tuxTwshTaTotacttime**

### Syntax

INTEGER

### Access

read-only

### Description

Time, in seconds, that the WSH has been active since it started processing.

## **tuxTwshTaTotidlttime**

### Syntax

INTEGER

### Access

read-only

### Description

Time, in seconds, that the WSH has been idle since it started processing.

## **tuxTwshTaCurwork**

### Syntax

INTEGER

### Access

read-only

## Description

Amount of work processed by this WSH since the last WSH assignment by the WSL. This value is used by the WSL to load balance new incoming connections among a set of WSH processes.

## **tuxTwshTaFlowcnt**

### Syntax

INTEGER

### Access

read-only

## Description

Number of times flow control has been encountered by this WSH. This object should be considered only in relation to recent past values because it might wrap around during the lifetime of the WSH.

## **tuxTwshTaNumblockQ**

### Syntax

INTEGER

### Access

read-only

## Description

Number of times this WSH has been unable to enqueue a message to a local UNIX system message queue due to queue blocking conditions. This object should be considered only in relation to recent past values because it might wrap around during the lifetime of the WSH.

## **tuxTwshTaRcvdByt**

### Syntax

INTEGER

## Access

read-only

## Description

Number of bytes received from the network by this WSH from all its present and past workstation clients. This object should be considered only in relation to recent past values because it might wrap around during the lifetime of the WSH.

## **tuxTwshTaRcvdNum**

## Syntax

INTEGER

## Access

read-only

## Description

Number of Tuxedo system messages received from the network by this WSH from all its present and past workstation clients. This object should be considered only in relation to recent past values because it might wrap around during the lifetime of the WSH.

## **tuxTwshTaSentByt**

## Syntax

INTEGER

## Access

read-only

## Description

Number of bytes sent to the network by this WSH to all its present and past workstation clients. This object should be considered only in relation to recent past values because it might wrap around during the lifetime of the WSH.

## tuxTwshTaSentNum

### Syntax

INTEGER

### Access

read-only

### Description

Number of Tuxedo system messages sent to the network by this WSH to all its present and past workstation clients. This object should be considered only in relation to recent past values because it might wrap around during the lifetime of the WSH.

## tuxTwsITbl

The tuxTwsITbl table represents configuration and run-time characteristics of WSL server processes configured to manage workstation groups. These object values identify and characterize workstation-specific configuration objects for WSL tuxTsrvrTbl objects within the application. To create a new row in this table, use a SET request that specifies the values for at least tuxTwsITaSrvGrp, tuxTwsITaSrvId, and tuxTwsITaNaddr.

Object Name	Object ID
tuxTwsITaSrvGrp	.1.3.6.1.4.1.140.300.1.2.1.1
tuxTwsITaSrvId	.1.3.6.1.4.1.140.300.1.2.1.2
tuxTwsITaGrpNo	.1.3.6.1.4.1.140.300.1.2.1.3
tuxTwsITaState	.1.3.6.1.4.1.140.300.1.2.1.4
tuxTwsITaLmid	.1.3.6.1.4.1.140.300.1.2.1.5
tuxTwsITaPid	.1.3.6.1.4.1.140.300.1.2.1.6
tuxTwsITaDevice	.1.3.6.1.4.1.140.300.1.2.1.7
tuxTwsITaNaddr	.1.3.6.1.4.1.140.300.1.2.1.8
tuxTwsITaWshName	.1.3.6.1.4.1.140.300.1.2.1.9

<b>Object Name</b>	<b>Object ID</b>
tuxTws1TaMinHandlers	.1.3.6.1.4.1.140.300.1.2.1.10
tuxTws1TaMaxHandlers	.1.3.6.1.4.1.140.300.1.2.1.11
tuxTws1TaMultiplex	.1.3.6.1.4.1.140.300.1.2.1.12
tuxTws1TaMaxIdleTime	.1.3.6.1.4.1.140.300.1.2.1.13
tuxTws1TaMaxInitTime	.1.3.6.1.4.1.140.300.1.2.1.14
tuxTws1TaClOpt	.1.3.6.1.4.1.140.300.1.2.1.15
tuxTws1TaEnvFile	.1.3.6.1.4.1.140.300.1.2.1.16
tuxTws1TaGrace	.1.3.6.1.4.1.140.300.1.2.1.17
tuxTws1TaMaxGen	.1.3.6.1.4.1.140.300.1.2.1.18
tuxTws1TaRcmd	.1.3.6.1.4.1.140.300.1.2.1.19
tuxTws1TaRestart	.1.3.6.1.4.1.140.300.1.2.1.20
tuxTws1TaSequence	.1.3.6.1.4.1.140.300.1.2.1.21
tuxTws1TaCurHandlers	.1.3.6.1.4.1.140.300.1.2.1.22
tuxTws1TaHwHandlers	.1.3.6.1.4.1.140.300.1.2.1.23
tuxTws1TaWsProto	.1.3.6.1.4.1.140.300.1.2.1.24
tuxTws1TaSuspended	.1.3.6.1.4.1.140.300.1.2.1.25
tuxTws1TaViewRefresh	.1.3.6.1.4.1.140.300.1.2.1.26
tuxTws1TaKeepAlive	.1.3.6.1.4.1.140.300.1.2.1.28
tuxTws1TaNetTimeOut	.1.3.6.1.4.1.140.300.1.2.1.29

## **tuxTws1TaSrvGrp**

### **Syntax**

*DisplayString*( SIZE(1..30) )

## Access

read-write

## Description

Logical name of the server group. Server group names cannot contain an asterisk (\*), comma, or colon.

**Note:** This object can be updated only during row creation.

## tuxTwsITaSrvId

### Syntax

INTEGER (1..30001)

### Access

read-write

### Description

Unique (within the server group) server identification number.

**Note:** This object can be updated only during row creation.

## tuxTwsITaGrpNo

### Syntax

INTEGER (1..30001)

### Access

read-only

### Description

Group number associated with this servers group.

## **tuxTwsITaState**

### Syntax

```
INTEGER { active(1), inactive(2), migrating(3), cleaning(4), restarting(5),  
suspended(6), partitioned(7), dead(8), invalid(9) }
```

### Access

read-write

### Description

State for the WSL server within the application. Any state defined for the `tuxTsrvrTbl` group can be returned or set as indicated.

## **tuxTwsITaLmid**

### Syntax

```
DisplayString( SIZE (1..30) )
```

### Access

read-only

### Description

Current logical machine on which the server is running.

## **tuxTwsITaPid**

### Syntax

```
INTEGER
```

### Access

read-only

### Description

Native operating system process identifier for the WSL server.

## tuxTwsITaDevice

### Syntax

*DisplayString*( SIZE(0..78) )

### Access

read-write

### Description

Device name to be used by the WSL process to access the network. This object value is a required value for access to a network through a TLI-based Tuxedo system binary. This object value is not needed for sockets-based Tuxedo system binaries.

## tuxTwsITaNaddr

### Syntax

*DisplayString*( SIZE(1..78) )

### Access

read-write

### Description

Specifies the complete network address to be used by the WSL process as its listening address. The listening address for a WSL is the means by which it is contacted by workstation client processes participating in the application.

If string has the form 0xhex-digits or \\xhex-digits, it must contain an even number of valid hexadecimal digits. These forms are translated internally into a character array containing the hexadecimal representations of the string specified.

## tuxTwsITaWshName

### Syntax

*DisplayString*( SIZE(1..78) )

### Access

read-write

## Description

The name of the executable that provides workstation handler services for this workstation listener. The default value for this object is WSH, which corresponds to the system provided workstation handler. Workstation handlers can be customized using the command `buildwsh`.

## **tuxTwsITaMinHandlers**

### Syntax

INTEGER (0..256)

### Access

read-write

## Description

The minimum number of handlers that should be available in conjunction with this WSL at any given time. Upon being activated, the WSL starts this many WSHs immediately and does not deplete the supply of WSHs below this number until the administrator issues a shutdown to the WSL. Modifications to this object for a running WSL might cause additional handlers to be activated.

## **tuxTwsITaMaxHandlers**

### Syntax

INTEGER (0..32767)

### Access

read-write

## Description

The maximum number of handlers that should be available in conjunction with this WSL at any given time. Handlers are started as necessary to meet the demand of workstation clients attempting to access the system. This object must be greater than or equal to the setting for the minimum number of handlers.

## **tuxTwsITaMultiplex**

### Syntax

INTEGER (0..32767)

### Access

read-write

### Description

Maximum number of clients that are supported by any one handler process concurrently.

## **tuxTwsITaMaxIdleTime**

### Syntax

INTEGER

### Access

read-write

### Description

Maximum amount of time, in minutes, that a workstation client is permitted to be idle before it is abortively disconnected from the application by the handler. A value of 0 allows clients to be idle as long as is necessary without being timed out.

## **tuxTwsITaMaxInitTime**

### Syntax

INTEGER

### Access

read-write

### Description

The minimum number of seconds that should be allowed for a workstation client to complete initialization processing through the WSH before being timed out by the WSL.

## tuxTwsITaCIOpt

### Syntax

*DisplayString*( SIZE (0..128) )

### Access

read-write

### Description

Command-line options to be passed to the WSL server when it is activated. For details, see reference page [servopts\(5\)](#) in *Oracle Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*.

**Note:** Run-time modifications to this object do not affect a running WSL server. Server-specific options (that is, those after a double-dash "--") cannot be set and are not returned.

## tuxTwsITaEnvFile

### Syntax

*DisplayString*( SIZE (0..78) )

### Access

read-write

### Description

WSL server-specific environment file. See `tuxTmachineEnvFile` for a complete discussion of how this file is used to modify the environment.

**Note:** Run-time modifications to this object do not affect a running WSL server.

## tuxTwsITaGrace

### Syntax

INTEGER

### Access

read-write

## Description

The period of time, in seconds, during which the `tuxTwslTaMaxGen` limit applies. This object value is meaningful only for restartable WSL servers, that is, if the `tuxTwslTaRestart` object is set to `yes(1)`. When a restarting server would exceed the `tuxTwslTaMaxGen` limit but the `tuxTwslTaGrace` period has expired, the system resets the current generation (`tuxTsrvrGeneration`) to 1 and resets the initial boot time (`tuxTsrvrTimeStart`) to the current time. A value of 0 for this object indicates that the WSL server should always be restarted.

## tuxTwslTaMaxGen

### Syntax

INTEGER (0..256)

### Access

read-write

### Description

Number of generations allowed for a restartable WSL server (`tuxTwslTaRestart == yes(1)`) over the specified grace period (`tuxTwslTaGrace`). The initial activation of the WSL server counts as one generation and each restart also counts as one. Processing after the maximum number of generations is exceeded is discussed above with respect to `tuxTwslTaGrace`.

## tuxTwslTaRcmd

### Syntax

*DisplayString*(SIZE(0..78))

### Access

read-write

### Description

Application specified command to be executed in parallel with the system restart of an application server. This command must be an executable file in the native operating system.

## tuxTwsITaRestart

### Syntax

INTEGER { yes(1), no(2) }

### Access

read-write

### Description

Restartable (yes(1)) or non-restartable (no(2)) WSL server. If server migration is specified for this server group (tuxTdomainOptions = migrate(2) and tuxTgroupLMID with alternate site), then this object must be set to yes(1).

## tuxTwsITaSequence

### Syntax

INTEGER (1..10000)

### Access

read-write

### Description

Specifies when this server should be booted (tmboot(1)) or shutdown (tmshutdown(1)) relative to other servers. If two servers are given the same sequence number, it is possible for tmboot(1) to boot them in parallel and for tmshutdown(1) to shut them down in parallel. tuxTwsITb1 instances added without a tuxTwsITaSequence object specified or with an invalid value have a value generated for them that is 10,000 or more and is higher than any other automatically selected default value. Servers are booted by tmboot(1) in increasing order of sequence number and shutdown by tmshutdown(1) in decreasing order. Run-time modifications to this object affect only tmboot(1) and tmshutdown(1) and affect the order in which running servers can be shutdown by a subsequent invocation of tmshutdown(1).

## tuxTwsITaCurHandlers

### Syntax

INTEGER

## Access

read-only

## Description

Number of currently active handlers associated with this WSL.

## **tuxTwsITaHwHandlers**

### Syntax

INTEGER

### Access

read-only

### Description

Maximum number of currently active handlers associated with this WSL at any one time.

## **tuxTwsITaWsProto**

### Syntax

INTEGER

### Access

read-only

### Description

The Tuxedo system /WS protocol version number for this /WS group. Note that /WS clients connecting to this group might themselves have a different protocol version number associated with them.

## **tuxTwsITaSuspended**

### Syntax

INTEGER { new(1), all(2), none(3) }

## Access

read-write

## Description

A value of `new(1)` indicates that new incoming clients cannot connect through this `tuxTws1Tbl` instance. A value of `all(2)` indicates that workstation clients already connected to the application through this WSL have been `suspended(2)` (see `tuxTclientState`) in addition to new incoming connections being disallowed. A value of `none(3)` indicates that no suspension characteristics are in effect.

## **tuxTws1TaViewRefresh**

### Syntax

```
INTEGER { yes(1), no-value-returned(2) }
```

## Access

read-write

## Description

Setting a value of `yes(1)` causes all active WSHs in the /WS group to refresh their VIEW buffer type cache. A `GET` request on this object always returns `no-value-returned(2)` and does not mean anything. This object has meaning only for `SET` requests.

## **tuxTws1TaKeepAlive**

### Syntax

```
INTEGER { client(1), handler(2), both(3), none(4), not-available(5) }
```

## Access

read-write

## Description

The network “keep alive” option is configured for the client, the handler, or both the client and the handler, or not on either side of the connection. Changing this value only affects future connections.

## **tuxTwsITaNetTimeOut**

### Syntax

INTEGER (0..35204650)

### Access

read-write

### Description

The minimum number of seconds that should be allowed for a workstation client to wait for a response from WSL/WSH. A value of 0 indicates no network time-out. Changing this value affects only future connections. This object is supported only on Tuxedo 6.4. -1 is returned if the object is not available.



# Application Queue MIB

Oracle Tuxedo systems incorporate the capability to use application queues for time-independent communication. The Tuxedo Application Queue MIB provides the administrative environment required for managing and controlling access to application queues. The Application Queue MIB defines the structure of the application queues.

In Tuxedo applications, messages are stored on a queue, and queues are defined within a particular queue space. Queueing and dequeuing is done within a transaction. The Application Queue MIB consists of five different groups for defining queue access, queues, messages, queue spaces, and queue transactions.

The Application Queue MIB consists of the following groups.

Group Name	Description
<a href="#">tuxTAppQctrl</a>	Access control to application queues
<a href="#">tuxTAppQTbl</a>	Application queues within a queue space
<a href="#">tuxTAppQmsgTbl</a>	Messages within an application queue
<a href="#">tuxTQspaceTbl</a>	Application queue spaces
<a href="#">tuxTQtransTbl</a>	Transactions associated with application queues

# tuxTAppQctrl

The `tuxTAppQctrl` group enables controlled access to all Application Queue related MIB groups.

Object Name	Object ID
<code>tuxTAppQctrlLmid</code>	<code>.1.3.6.1.4.1.140.300.12.5.1</code>
<code>tuxTAppQctrlQmConfig</code>	<code>.1.3.6.1.4.1.140.300.12.5.2</code>
<code>tuxTAppQctrlSpaceName</code>	<code>.1.3.6.1.4.1.140.300.12.5.3</code>
<code>tuxTAppQctrlQname</code>	<code>.1.3.6.1.4.1.140.300.12.5.4</code>
<code>tuxTAppQctrlMsgLoPrio</code>	<code>.1.3.6.1.4.1.140.300.12.5.5</code>
<code>tuxTAppQctrlMsgHiPrio</code>	<code>.1.3.6.1.4.1.140.300.12.5.6</code>
<code>tuxTAppQctrlMsgEndTime</code>	<code>.1.3.6.1.4.1.140.300.12.5.7</code>
<code>tuxTAppQctrlMsgStartTime</code>	<code>.1.3.6.1.4.1.140.300.12.5.8</code>
<code>tuxTAppQctrlMsgExpireEndTime</code>	<code>.1.3.6.1.4.1.140.300.12.5.20</code>
<code>tuxTAppQctrlMsgExpireStartTime</code>	<code>.1.3.6.1.4.1.140.300.12.5.30</code>

## tuxTAppQctrlLmid

### Syntax

```
INTEGER { local(1), all(2) }
```

### Access

read-write

### Description

This applies to all Application Queue related MIB groups. This object value controls the machines for which the values are returned.

If the value is `local(1)`, only the local host where Oracle SNMP Agent is running is considered; alternatively, all LMIDs known to the application are considered if the value is `all(2)`.

The default for this object is `local(1)`.

## tuxTAppQctrlQmConfig

### Syntax

*DisplayString*(SIZE(1..78))

### Access

read-write

### Description

This applies to all Application Queue related MIB groups. This object value controls the device for which the values are returned.

The default for this object is “\*”, in which case all known devices (which are a part of some group) are considered.

## tuxTAppQctrlSpaceName

### Syntax

*DisplayString*(SIZE(1..15))

### Access

read-write

### Description

This applies to all Application Queue related MIB groups. This object value controls the queue space for which the values are returned.

The default for this object is “\*”, in which case all queue spaces for the devices (qualified by `tuxTAppQctrlQmConfig`) are considered.

## tuxTAppQctrlQname

### Syntax

*DisplayString*(SIZE(1..127))

## Access

read-write

## Description

This object value controls the queue for which the values are returned. This applies to `tuxTAppQTbl` and `tuxTAppQmsgTbl`.

The default for this object is “\*”, in which case all queues for the devices (qualified by `tuxTAppQctrlQmConfig`) and queue spaces (qualified by `tuxTAppQctrlSpaceName`) are considered.

## **tuxTAppQctrlMsgLoPrio**

### Syntax

INTEGER

### Access

read-write

### Description

This object applies only to `tuxTAppQmsgTbl`. The lowest priority within which to search for occurrences of `tuxTAppQmsgTbl` instances. This object value is valid only for **PRIO**-based queues. By default, the minimum value of priority is considered. To revert to the default setting, set this object to 0.

## **tuxTAppQctrlMsgHiPrio**

### Syntax

INTEGER

### Access

read-write

### Description

This object applies only to `tuxTAppQmsgTbl`. The highest priority within which to search for occurrences of `tuxTAppQmsgTbl` instances. This object value is valid only for **PRIO**-based

queues. By default, the maximum value of priority is considered. To revert to the default setting, set this object to 0.

## tuxTAppQctrlMsgEndTime

### Syntax

*DisplayString*(SIZE(1..15))

### Access

read-write

### Description

This object applies only to tuxTAppQmsgTbl. The end time within which to search for occurrences of tuxTAppQmsgTbl instances. The range is inclusive. This object value is valid only for TIME-based queues. The default value is the maximum number possible on that machine. To use the default setting, set this object to “\*”.

YY[MM[DD[hh[mm[ss]]]]]

Specifies the year, month, date, hour, minute, and second respectively. Any value which is not specified defaults to its minimum value (e.g., 9506 is taken as 950601000000). The years 00 through 37 are treated as 2000 through 2037, 70 through 99 as 1970 through 1999, and 38 through 69 are invalid.

## tuxTAppQctrlMsgStartTime

### Syntax

*DisplayString*(SIZE(1..15))

### Access

read-write

### Description

This object applies only to tuxTAppQmsgTbl. The start time within which to search for occurrences of tuxTAppQmsgTbl instances. The range is inclusive. This object value is valid only for TIME-based queues. By default, the minimum time value is considered to be 0. To use the default setting, set this object to “\*”.

YY[MM[DD[hh[mm[ss]]]]]

Specifies the year, month, date, hour, minute, and second respectively. Any value which is not specified defaults to its minimum value (e.g., 9506 is taken as 950601000000). The years 00 through 37 are treated as 2000 through 2037, 70 through 99 as 1970 through 1999, and 38 through 69 are invalid.

## tuxTAppQctrlMsgExpireEndTime

### Syntax

*DisplayString*(SIZE(1..12))

### Access

read-write

### Description

This object applies only to `tuxTAppQmsgTbl`. The expire end time within which to search for occurrences of `tuxTAppQmsgTbl` instances. The range is inclusive. This object value is valid only for TIME-based queues. The default value is the maximum number possible on that machine. To use the default setting, set this object to “\*”.

YY[MM[DD[hh[mm[ss]]]]]

Specifies the year, month, date, hour, minute, and second respectively. Any value which is not specified defaults to its minimum value (e.g., 9506 is taken as 950601000000). The years 00 through 37 are treated as 2000 through 2037, 70 through 99 as 1970 through 1999, and 38 through 69 are invalid.

## tuxTAppQctrlMsgExpireStartTime

### Syntax

*DisplayString*(SIZE(1..12))

### Access

read-write

### Description

This object applies only to `tuxTAppQmsgTbl`. The Expire start time within which to search for occurrences of `tuxTAppQmsgTbl` instances. The range is inclusive. This object value is valid only

for TIME-based queues. By default, the minimum time value is considered to be 0. To use the default setting, set this object to “\*”.

YY[MM[DD[hh[mm[ss]]]]]

Specifies the year, month, date, hour, minute, and second respectively. Any value which is not specified defaults to its minimum value (e.g., 9506 is taken as 950601000000). The years 00 through 37 are treated as 2000 through 2037, 70 through 99 as 1970 through 1999, and 38 through 69 are invalid.

## tuxTAppQTbl

The tuxTAppQTbl group contains objects that represent application queues. One or more application queues can exist in a single application queue space. Objects in this table are only accessible through a Tuxedo SNMP agent installed on the local machine.

**Creation of a New Queue** — For creating a new queue(row), in this group the SET request should have the value of tuxTAppQname, tuxTAppQspaceName, and tuxTAppQmConfig. Also the value of tuxTAppQgrpNo (which is a part of the index) should be the corresponding group number for that queue space or “40000” (if no such group exists).

**Note:** For this and all other Application Queue related MIB groups, there is a control MIB which can be used to filter the data returned as a part of all Application Queue related MIB groups. Refer to tuxTAppQctrl.

To create a new row in this table, issue a SET request that specifies at least the values for tuxTAppQname, tuxTAppQspaceName, and tuxTAppQmConfig.

Object Name	Object ID
tuxTAppQname	.1.3.6.1.4.1.140.300.12.1.1.1
tuxTAppQspaceName	.1.3.6.1.4.1.140.300.12.1.1.2
tuxTAppQmConfig	.1.3.6.1.4.1.140.300.12.1.1.3
tuxTAppQlmid	.1.3.6.1.4.1.140.300.12.1.1.4
tuxTAppQgrpNo	.1.3.6.1.4.1.140.300.12.1.1.5
tuxTAppQstate	.1.3.6.1.4.1.140.300.12.1.1.6
tuxTAppQorder	.1.3.6.1.4.1.140.300.12.1.1.7

<b>Object Name</b>	<b>Object ID</b>
tuxTAppQcmd	.1.3.6.1.4.1.140.300.12.1.1.8
tuxTAppQcmdHw	.1.3.6.1.4.1.140.300.12.1.1.9
tuxTAppQcmdLw	.1.3.6.1.4.1.140.300.12.1.1.10
tuxTAppQmaxRetries	.1.3.6.1.4.1.140.300.12.1.1.11
tuxTAppQoutOfOrder	.1.3.6.1.4.1.140.300.12.1.1.12
tuxTAppQretryDelay	.1.3.6.1.4.1.140.300.12.1.1.13
tuxTAppQcurBlocks	.1.3.6.1.4.1.140.300.12.1.1.14
tuxTAppQcurMsg	.1.3.6.1.4.1.140.300.12.1.1.15
tuxTAppQDefExpirationTime	.1.3.6.1.4.1.140.300.12.1.1.30
tuxTAppQDefDeliveryPolicy	.1.3.6.1.4.1.140.300.12.1.1.40
tuxTAppQCmdNonPersist	.1.3.6.1.4.1.140.300.12.1.1.50
tuxTAppQCmdNonPersistHw	.1.3.6.1.4.1.140.300.12.1.1.60
tuxTAppQCmdNonPersistLw	.1.3.6.1.4.1.140.300.12.1.1.70

Object Name	Object ID
tuxTAppQCurNonPersistBytes	.1.3.6.1.4.1.140.300.12.1.1.80
tuxTAppQCurNonPersistMsg	.1.3.6.1.4.1.140.300.12.1.1.90

## tuxTAppQname

### Syntax

*DisplayString* (SIZE(1..127))

### Access

read-write

### Description

Name of the application queue.

**Note:** This object can be updated only during row creation.

## tuxTAppQspaceName

### Syntax

*DisplayString* (SIZE(1..15))

### Access

read-write

### Description

Name of the application queue space containing the application queue.

**Note:** This object can be updated only during row creation.

## tuxTAppQmConfig

### Syntax

*DisplayString* (SIZE(1..78))

## Access

read-write

## Description

Absolute pathname of the file or device where the application queue space is located.

**Note:** This object can be updated only during row creation.

## **tuxTAppQlmid**

### Syntax

*DisplayString* (SIZE(1..30))

## Access

read-write

## Description

Identifier of the logical machine where the application queue space is located.

**Note:** This object can be updated only during row creation.

## **tuxTAppQgrpNo**

### Syntax

INTEGER (1..29999)

## Access

read-write

## Description

Group number of any server group for which this queue is a resource manager, in other words that group's openinfo string `tuxTgroupOpenInfo` contains the device name and queue space name for this queue.

**Note:** This object can be updated only during row creation.

## tuxTAppQstate

### Syntax

```
INTEGER { valid(1), invalid(2) }
```

### Access

read-write

### Description

The values for GET and SET operations are as follows:

GET: valid(1)

A GET operation retrieves information about the selected application queues. The following list describes the meaning of the tuxTAppQstate object returned in response to a GET request. States not listed are not returned.

valid(1)

The specified queue exists.

SET: invalid(2)

A SET operation changes characteristics of the selected application queue or creates a new queue. The following list describes the meaning of the tuxTAppQstate object returned by a SET request. States not listed cannot be set.

invalid(2)

Delete the specified queue. If the queue space has processes attached to it, the queue is not deleted. In addition, if the queue has messages in it, it is not deleted. Successful return leaves the object in the invalid(2) state.

## tuxTAppQorder

### Syntax

```
DisplayString (SIZE(1..30))
```

### Access

read-write

### Description

The order in which messages in the queue are to be processed. Legal values are PRIORITY or TIME, followed by a comma, optionally followed by another occurrence of PRIORITY or TIME, followed by

one of the values `LIFO` or `FIFO`. If neither `FIFO` nor `LIFO` is specified, `FIFO` is assumed. If nothing is specified when a queue is created, the default is `FIFO`. For example, these are some legal settings:

```
PRIO
PRIO, TIME, LIFO
TIME, PRIO, FIFO
TIME, FIFO
```

## **tuxTAppQcmd**

### Syntax

*DisplayString*(*SIZE*(0..127)) (up to 78 bytes for Oracle Tuxedo 8.0 or earlier)

### Access

read-write

### Description

The command to be automatically executed when the high water mark, `tuxTAppQcmdHw`, is reached. The command is re-executed when the high water mark is reached again after the low water mark, `tuxTAppQcmdLw`, has been reached.

## **tuxTAppQcmdHw**

### Syntax

*DisplayString*

### Access

read-write

### Description

The high water mark. Refer to `tuxTAppQcmdLw` for further information.

## tuxTAppQcmdLw

### Syntax

*DisplayString*

### Access

read-write

### Description

The low water marks that control the automatic execution of the command specified in the `tuxTAppQcmd` object. Each is an integer greater than or equal to zero optionally followed by one of the following keyletters. The keyletters must be consistent for `tuxTAppQcmdHw` and `tuxTAppQcmdLw`.

b

The high and low water marks pertain to the number of bytes used by messages in the queue.

B

The high and low water marks pertain to the number of blocks used by messages in the queue.

m

The high and low water marks pertain to the number of messages in the queue.

%

The high and low water marks are expressed in terms of a percentage of queue capacity.

For example, if `tuxTAppQcmdLw` is 50m and `tuxTAppQcmdHw` is 100m, then the command specified in `tuxTAppQcmd` is executed when 100 messages are on the queue, and it is not executed again until the queue is drained below 50 messages and is filled again to 100 messages.

## tuxTAppQmaxRetries

### Syntax

INTEGER

### Access

read-write

## Description

The maximum number of retries for a failed queue message. When the number of retries is exhausted, the message is placed on the error queue of the associated application queue space. If there is no error queue, the message is dropped. The default is zero.

## **tuxTAppQoutOfOrder**

### Syntax

```
INTEGER { none(1), top(2), msgid(3) }
```

### Access

read-write

## Description

The way in which out-of-order message processing is to be handled. The default is `none(1)`.

## **tuxTAppQretryDelay**

### Syntax

```
INTEGER
```

### Access

read-write

## Description

The delay, in seconds, between retries for a failed queue message. The default is zero.

## **tuxTAppQcurBlocks**

### Syntax

```
INTEGER
```

### Access

read-only

## Description

The number of disk pages currently consumed by the queue.

## **tuxTAppQcurMsg**

### Syntax

INTEGER

### Access

read-only

## Description

The number of messages currently in the queue.

## **tuxTAppQDefExpirationTime**

### Syntax

DisplayString

### Access

read-write

## Description

This object specifies an expiration time for messages enqueued with no explicit expiration time. The expiration time can be either a relative expiration time or none. The relative expiration time is determined by associating a fixed amount of time with a message after the message arrives at the queue manager process. When a message's expiration time is reached and the message has not been dequeued or administratively deleted, all resources associated with the message are reclaimed by the system and statistics are updated. If a message expires during a transaction, the expiration does not cause the transaction to fail. Messages that expire while being enqueued or dequeued within a transaction are removed from the queue when the transaction ends. There is no notification that the message has expired. If no default expiration time is specified for a queue, the message without an explicit expiration time does not expire. When the queue's expiration time is modified, the expiration times of messages that were in the queue before the modification are not changed.

The format is `+seconds`, where `seconds` is the number of seconds allowed to lapse between the time that the queue manager successfully completes the operation and the time that the message is to expire. If `seconds` is set to zero (0), the message expires immediately.

The value of this object may also be set to the string “none.” The `none` string indicates that messages enqueued to the queue with no explicit expiration time do not expire. You can change the expiration time for messages already in a queue with the `tuxTAppQmsgExpireTime` object of the `tuxTAppQmsgTbl` group.

## **tuxTAppQDefDeliveryPolicy**

### **Syntax**

```
INTEGER { persist(1), non-persist(2) }
```

### **Access**

read-write

### **Description**

This object specifies the default delivery policy for the queue when no delivery mode is specified for a message enqueued to the queue. When the value is “`persist`,” messages enqueued to the queue without an explicitly specified delivery mode are delivered using the persistent (disk-based) delivery method. When the value is `non-persist`, messages enqueued to the queue without an explicitly specified delivery method are delivered using the non-persistent (in memory) delivery method. When a queue’s default delivery policy is modified, the delivery quality of service of messages that are in the queue before the modification are not changed. If the queue being modified is the reply queue named for any messages currently in the queue space, the reply quality of service is not changed for those messages as a result of changing the default delivery policy of the queue.

For non-persistent delivery, if the memory area is exhausted or fragmented so that a message cannot be enqueued, the enqueueing operation fails, even if there is sufficient persistent storage for the message. Similarly, if the persistent storage area is exhausted or fragmented so that a message cannot be enqueued, the enqueueing operation fails, even if there is sufficient non-persistent storage for the message. If the `tuxTQspaceMemNonPersist` object of the `tuxTQspaceTbl` group is zero (0) for a queue space, no space is reserved for non-persistent messages. In such a case, any attempt to enqueue a non-persistent message fails. This type of failure results, for example, when no delivery quality of service has been specified for a message and the `tuxTAppQDefDeliveryPolicy` object for the target queue has been set to “`non-persist`.”

## tuxTAppQCmdNonPersist

### Syntax

*DisplayString*(SIZE(1..127)) (up to 78 bytes for Oracle Tuxedo 8.0 or earlier)

### Access

read-write

### Description

This object specifies the command to be executed automatically when the high-water mark for non-persistent (memory-based delivery) messages, `tuxTAppQCmdNonPersistHw`, is reached. The command is re-executed when the high-water mark is reached again after the low-water mark for non-persistent (memory-based delivery) messages, `tuxTAppQCmdNonPersistLw`, has been reached.

## tuxTAppQCmdNonPersistHw

### Syntax

*DisplayString*

### Access

read-write

### Description

These objects specify the high- and low-water marks that control the automatic execution of the command specified in the `tuxTAppQCmdNonPersist` object. Each is an integer greater than or equal to zero, followed by one of the following keyletters. The keyletters must be consistent for `tuxTAppQCmdNonPersistHw` and `tuxTAppQCmdNonPersistLw`.

b

The high- and low-water marks are expressed as the number of bytes used by non-persistent (in-memory) messages in the queue.

B

The high- and low-water marks are expressed as the number of blocks used by non-persistent (in-memory) messages in the queue.

%

The high- and low-water marks are expressed as a percentage of the shared memory capacity reserved for non-persistent messages in the queue space used by the queue.

The messages threshold type specified through the `tuxTAppQCmdHw` and `tuxTAppQCmdLw` objects (when followed by an `m`) applies to all messages in a queue, including both persistent and non-persistent messages, and therefore is not available as a threshold type for `tuxTAppQCmdNonPersistHw` and `tuxTAppQCmdNonPersistLw`.

## **tuxTAppQCmdNonPersistLw**

### Syntax

*DisplayString*

### Access

read-write

### Description

These objects specify the high- and low-water marks that control the automatic execution of the command specified in the `tuxTAppQCmdNonPersist` object. Each is an integer greater than or equal to zero, followed by one of the following keyletters. The keyletters must be consistent for `tuxTAppQCmdNonPersistHw` and `tuxTAppQCmdNonPersistLw`.

`b`

The high- and low-water marks are expressed as the number of bytes used by non-persistent (in-memory) messages in the queue.

`B`

The high- and low-water marks are expressed as the number of blocks used by non-persistent (in-memory) messages in the queue.

%

The high- and low-water marks are expressed as a percentage of the shared memory capacity reserved for non-persistent messages in the queue space used by the queue.

The messages threshold type specified through the `tuxTAppQCmdHw` and `tuxTAppQCmdLw` objects (when followed by an `m`) applies to all messages in a queue, including both persistent and non-persistent messages, and therefore is not available as a threshold type for `tuxTAppQCmdNonPersistHw` and `tuxTAppQCmdNonPersistLw`.

## tuxTAppQCurNonPersistBytes

### Syntax

INTEGER

### Access

read-write

### Description

This object specifies the number of shared memory bytes currently consumed by the non-persistent messages on the queue.

## tuxTAppQCurNonPersistMsg

### Syntax

INTEGER

### Access

read-write

### Description

This object specifies the number of non-persistent messages currently in the queue. To determine the total number of messages in the queue, add the value of `tuxTAppQCurMsg` to this value.

## tuxTAppQmsgTbl

The `tuxTAppQmsgTbl` group contains objects that represent messages stored in application queues. A message is not created by an administrator; instead, it comes into existence as a result of a call to `topenqueue(3)`. A message can be destroyed either by a call to `tpdequeue(3)` or by an administrator. In addition, certain objects of a message can be modified by an administrator. For example, an administrator can move a message from one queue to another queue within the same queue space or change its priority.

Objects in this table are only accessible through a Tuxedo SNMP agent installed on the local machine.

<b>Object Name</b>	<b>Object ID</b>
tuxTAppQmsgId	.1.3.6.1.4.1.140.300.12.2.1.1
tuxTAppQmsgSerNo	.1.3.6.1.4.1.140.300.12.2.1.2
tuxTAppQmsgGrpNo	.1.3.6.1.4.1.140.300.12.2.1.3
tuxTAppQmsgQname	.1.3.6.1.4.1.140.300.12.2.1.4
tuxTAppQmsgQmConfig	.1.3.6.1.4.1.140.300.12.2.1.5
tuxTAppQmsgQspaceName	.1.3.6.1.4.1.140.300.12.2.1.6
tuxTAppQmsgLmid	.1.3.6.1.4.1.140.300.12.2.1.7
tuxTAppQmsgState	.1.3.6.1.4.1.140.300.12.2.1.8
tuxTAppQmsgNewQname	.1.3.6.1.4.1.140.300.12.2.1.9
tuxTAppQmsgPrior	.1.3.6.1.4.1.140.300.12.2.1.10
tuxTAppQmsgTime	.1.3.6.1.4.1.140.300.12.2.1.11
tuxTAppQmsgCorId	.1.3.6.1.4.1.140.300.12.2.1.12
tuxTAppQmsgCurRetries	.1.3.6.1.4.1.140.300.12.2.1.13
tuxTAppQmsgSize	.1.3.6.1.4.1.140.300.12.2.1.14
tuxTAppQmsgExpireTime	.1.3.6.1.4.1.140.300.12.2.1.20
tuxTAppQmsgPersistent	.1.3.6.1.4.1.140.300.12.2.1.30
tuxTAppQmsgReplyPersistent	.1.3.6.1.4.1.140.300.12.2.1.40

## tuxTAppQmsgId

### Syntax

*DisplayString*(SIZE(1..32))

### Access

read-only

## Description

A unique identifier for the queue message, which can be used to select the message for GET or SET operations. No significance should be placed on this value beyond using it for equality comparisons.

## **tuxTAppQmsgSerNo**

### Syntax

INTEGER

### Access

read-only

## Description

A running number corresponding to `tuxTAppQmsgId` for the queue message, which is a part of the composite index of this table.

## **tuxTAppQmsgGrpNo**

### Syntax

INTEGER

### Access

read-only

## Description

Group number of any server group for which this queue is a resource manager, in other words that group's `openinfo` string `tuxTgroupOpenInfo` contains the device name and queue space name for this queue.

## **tuxTAppQmsgQname**

### Syntax

*DisplayString*(SIZE(1..127))

## Access

read-only

## Description

Name of the application queue in which the message is stored.

## **tuxTAppQmsgQmConfig**

### Syntax

*DisplayString* (SIZE(1..78))

### Access

read-only

### Description

Absolute pathname of the file or device where the application queue space for the queue containing this message is located.

## **tuxTAppQmsgQspaceName**

### Syntax

*DisplayString* (SIZE(1..15))

### Access

read-only

### Description

Name of the application queue space containing the application queue in which this message is located.

## **tuxTAppQmsgLmid**

### Syntax

*DisplayString* (SIZE(1..30))

## Access

read-only

## Description

Logical machine id for the machine on which the queue containing this message is located.

## **tuxTAppQmsgState**

### Syntax

```
INTEGER { valid(1), invalid(2) }
```

### Access

read-write

### Description

The values for GET and SET operations are as follows:

GET: `valid(1)`

A GET operation retrieves information about the selected messages. The following list describes the meaning of the `tuxTAppQmsgState` object returned in response to a GET request. States not listed are not returned.

`valid(1)`

The message exists.

SET: `invalid(2)`

A SET operation changes characteristics of the selected message. The following list describes the meaning of the `tuxTAppQmsgState` object returned by a SET request. States not listed cannot be set.

`invalid(2)`

The message is deleted from its queue space. The message must be in state `valid(1)` before attempting this operation. Successful return leaves the object in the `invalid(2)` state.

## **tuxTAppQmsgNewQname**

### Syntax

```
DisplayString (SIZE(1..15))
```

## Access

read-write

## Description

Name of the queue into which to move the selected message. This queue must be an existing queue in the same queue space. The message must be in state `valid(1)` for this operation to succeed. This object value is not returned by a `GET` operation.

## **tuxTAppQmsgPrior**

### Syntax

INTEGER

## Access

read-write

## Description

The priority of the message. This object value is valid only for `PRIO`-based queues. The value `-1` is returned by a `GET` operation if the queue is not `PRIO`-based.

## **tuxTAppQmsgTime**

### Syntax

*DisplayString*(SIZE(1..15))

## Access

read-write

## Description

The time when the message is processed. This object value is valid only for `TIME`-based queues. The empty string is returned by a `GET` operation if the queue is not `TIME`-based. The format is one of the following:

*+seconds*

Specifies that the message is processed *seconds* in the future. The value zero specifies that the message should be processed immediately.

*YY[MM[DD[hh[mm[ss]]]]]*

Specifies the year, month, day, hour, minute, and second when the message should be processed. Omitted units default to their minimum possible values. For example, 9506 is equivalent to 950601000000. The years 00 through 37 are treated as 2000 through 20037, 70 through 99 are treated as 1970 through 1999, and 38 through 69 are invalid.

## **tuxTAppQmsgCorId**

### Syntax

*DisplayString(SIZE(0..32))*

### Access

read-only

### Description

The correlation identifier for this message provided by the application in the `tpenqueue(3)` request. The empty string indicates that a correlation identifier is not present.

## **tuxTAppQmsgCurRetries**

### Syntax

INTEGER

### Access

read-only

### Description

The number of retries that have been attempted so far on this message.

## **tuxTAppQmsgSize**

### Syntax

INTEGER

### Access

read-only

## Description

The size of the message, in bytes.

## tuxTAppQmsgExpireTime

### Syntax

*DisplayString*(SIZE(1..15))

### Access

read-write

## Description

This object specifies the time at which a message expires (that is, the time at which the message should be removed from the queue if it has not already been dequeued or administratively deleted). When a message expires, all resources it uses are reclaimed by the system and statistics are updated. If a message expires during a transaction, the expiration does not cause the transaction to fail. Messages that expire while being enqueued or dequeued within a transaction are removed from the queue when the transaction ends. There is no notification that the message has expired. Expiration times cannot be added to messages enqueued by versions of the Oracle Tuxedo system that do not support message expiration, even when the queue manager responsible for changing this value supports message expiration. Attempts to add an expiration time fail.

The empty string is returned by a `GET` operation if the expiration time is not set. The expiration time format is one of the following:

+seconds

Specifies that the message will be removed after the specified number of seconds. If the value of seconds is set to zero (0), the message is removed immediately from the queue. Relative expiration time is calculated on the basis of the time at which the MIB request arrives and has been processed by the corresponding queue manager.

YY[MM[DD[hh]MM[SS]]]

Specifies the year, month, day, hour, minute, and second when the message will be removed if it has not already been dequeued or administratively deleted. Omitted units default to their minimum possible values. For example, 9506 is equivalent to 950601000000. The years 00 through 37 are treated as 2000 through 2037, 70 through 99 are treated as 1970 through 1999, and 38 through 69 are invalid. An absolute expiration time is determined by the clock on the machine where the queue manager process resides.

none

Specifies that the message will never expire.

## tuxTAppQmsgPersistent

### Syntax

INTEGER { yes(1), no(2) }

### Access

read-only

### Description

This read-only state is set to “no” for non-persistent messages and “yes” for persistent messages. It is the delivery quality of service for the message.

## tuxTAppQmsgReplyPersistent

### Syntax

INTEGER { yes(1), no(2) }

### Access

read-only

### Description

This read-only state is set to “no” for non-persistent messages and “yes” for persistent messages. It is the delivery quality that replies to the message.

## tuxTQspaceTbl

The `tuxTQspaceTbl` group contains objects that represent application queue spaces. Objects in this table are only accessible through a Tuxedo SNMP agent installed on the local machine.

**Note:** The values returned by this MIB are controlled by `tuxTAppQctrl`. For details, see the description of the above group.

To create a new row in this table, a `SET` request should be issued with an index (`tuxTQspaceGrpNo`) of 40000, which is a reserved value for row creation in the table. The `SET` request also needs to specify values for at least `tuxTQspaceQmConfig`, `tuxTQspaceName`,

tuxTQspaceLmid, tuxTQspaceIpckey, tuxTQspaceMaxMsg, tuxTQspaceMaxPages, tuxTQspaceMaxProc, tuxTQspaceMaxQueues, and tuxTQspaceMaxTrans. The newly created instance (row) is not visible until it is attached to some server group.

<b>Object Name</b>	<b>Object ID</b>
tuxTQspaceName	.1.3.6.1.4.1.140.300.12.3.1.1
tuxTQspaceQmConfig	.1.3.6.1.4.1.140.300.12.3.1.2
tuxTQspaceLmid	.1.3.6.1.4.1.140.300.12.3.1.3
tuxTQspaceGrpNo	.1.3.6.1.4.1.140.300.12.3.1.4
tuxTQspaceState	.1.3.6.1.4.1.140.300.12.3.1.5
tuxTQspaceBlocking	.1.3.6.1.4.1.140.300.12.3.1.6
tuxTQspaceErrQname	.1.3.6.1.4.1.140.300.12.3.1.7
tuxTQspaceForceInit	.1.3.6.1.4.1.140.300.12.3.1.8
tuxTQspaceIpckey	.1.3.6.1.4.1.140.300.12.3.1.9
tuxTQspaceMaxMsg	.1.3.6.1.4.1.140.300.12.3.1.10
tuxTQspaceMaxPages	.1.3.6.1.4.1.140.300.12.3.1.11
tuxTQspaceMaxProc	.1.3.6.1.4.1.140.300.12.3.1.12
tuxTQspaceMaxQueues	.1.3.6.1.4.1.140.300.12.3.1.13
tuxTQspaceMaxTrans	.1.3.6.1.4.1.140.300.12.3.1.14
tuxTQspaceCurExtent	.1.3.6.1.4.1.140.300.12.3.1.15
tuxTQspaceCurMsg	.1.3.6.1.4.1.140.300.12.3.1.16
tuxTQspaceCurProc	.1.3.6.1.4.1.140.300.12.3.1.17
tuxTQspaceCurQueues	.1.3.6.1.4.1.140.300.12.3.1.18
tuxTQspaceCurTrans	.1.3.6.1.4.1.140.300.12.3.1.19
tuxTQspaceHwMsg	.1.3.6.1.4.1.140.300.12.3.1.20
tuxTQspaceHwProc	.1.3.6.1.4.1.140.300.12.3.1.21

<b>Object Name</b>	<b>Object ID</b>
tuxTQspaceHwQueues	.1.3.6.1.4.1.140.300.12.3.1.22
tuxTQspaceHwTrans	.1.3.6.1.4.1.140.300.12.3.1.23
tuxTQspacePercentInit	.1.3.6.1.4.1.140.300.12.3.1.24
tuxTQspaceMaxActions	.1.3.6.1.4.1.140.300.12.3.1.40
tuxTQspaceMaxHandles	.1.3.6.1.4.1.140.300.12.3.1.50
tuxTQspaceMaxOwners	.1.3.6.1.4.1.140.300.12.3.1.60
tuxTQspaceMaxTmpQueues	.1.3.6.1.4.1.140.300.12.3.1.70
tuxTQspaceMaxCursors	.1.3.6.1.4.1.140.300.12.3.1.80
tuxTQspaceMemNonPersist	.1.3.6.1.4.1.140.300.12.3.1.90
tuxTQspaceMemFilters	.1.3.6.1.4.1.140.300.12.3.1.100
tuxTQspaceMemOverFlow	.1.3.6.1.4.1.140.300.12.3.1.110
tuxTQspaceMemSystemReserved	.1.3.6.1.4.1.140.300.12.3.1.120
tuxTQspaceMemTotalAllocated	.1.3.6.1.4.1.140.300.12.3.1.130
tuxTQspaceCurActions	.1.3.6.1.4.1.140.300.12.3.1.140
tuxTQspaceCurHandles	.1.3.6.1.4.1.140.300.12.3.1.150
tuxTQspaceCurOwners	.1.3.6.1.4.1.140.300.12.3.1.160
tuxTQspaceCurTmpQueues	.1.3.6.1.4.1.140.300.12.3.1.170
tuxTQspaceCurCursors	.1.3.6.1.4.1.140.300.12.3.1.180
tuxTQspaceCurMemNonPersist	.1.3.6.1.4.1.140.300.12.3.1.190
tuxTQspaceCurMemFilters	.1.3.6.1.4.1.140.300.12.3.1.200
tuxTQspaceCurMemOverFlow	.1.3.6.1.4.1.140.300.12.3.1.210
tuxTQspaceHwActions	.1.3.6.1.4.1.140.300.12.3.1.220
tuxTQspaceHwHandles	.1.3.6.1.4.1.140.300.12.3.1.230

Object Name	Object ID
tuxTQspaceHwOwners	.1.3.6.1.4.1.140.300.12.3.1.240
tuxTQspaceHwTmpQueues	.1.3.6.1.4.1.140.300.12.3.1.250
tuxTQspaceHwCursors	.1.3.6.1.4.1.140.300.12.3.1.260
tuxTQspaceHwMemNonPersist	.1.3.6.1.4.1.140.300.12.3.1.270
tuxTQspaceHwMemFilters	.1.3.6.1.4.1.140.300.12.3.1.280
tuxTQspaceHwMemOverFlow	.1.3.6.1.4.1.140.300.12.3.1.290

## tuxTQspaceName

### Syntax

*DisplayString*(SIZE(1..15))

### Access

read-write

### Description

Name of the application queue space.

**Note:** This object can be updated only during row creation.

## tuxTQspaceQmConfig

### Syntax

*DisplayString*(SIZE(1..78))

### Access

read-write

### Description

Absolute pathname of the file or device where the application queue space is located.

**Note:** This object can be updated only during row creation.

## tuxTQspaceLmid

### Syntax

*DisplayString*(SIZE(1..30))

### Access

read-write

### Description

Identifier of the logical machine where the application queue space is located.

**Note:** This object can be updated only during row creation.

## tuxTQspaceGrpNo

### Syntax

INTEGER (1..29999)

### Access

read-write

### Description

Group number of any server group for which this queue space is a resource manager, in other words that group's `openinfo` string `tuxTgroupOpenInfo` contains the device name and queue space name for this queue space.

**Note:** This object can be updated only during row creation.

## tuxTQspaceState

### Syntax

INTEGER { inactive(1), initializing(2), open(3), active(4), cleaning(5),  
invalid(6) }

### Access

read-write

## Description

The values for GET and SET operations are as follows:

GET: `inactive(1) | initializing(2) | open(3) | active(4)`

A GET operation retrieves information about the selected application queue space. The following list describes the meaning of the `tuxTQspaceState` object returned in response to a GET request. States not listed are not returned.

`inactive(1)`

The queue space exists; i.e., disk space for it has been reserved in a device and the space has been initialized (if requested or if necessary).

`initializing(2)`

Disk space for the queue space is currently being initialized.

`open(3)`

Shared memory and other IPC resources for the queue space have been allocated and initialized, but no processes are currently attached to the shared memory.

`active(4)`

Shared memory and other IPC resources for the queue space have been allocated and initialized, and at least one process is currently attached to the shared memory. These processes can be the queue servers (`TMS_QM`, `TMQUEUE`, and perhaps `TMQFORWARD`) associated with the queue space, or they can be administrative processes such as `qmadmin(1)`, or they can be processes associated with another application.

SET: `open(3) | cleaning(5) | invalid(6)`

A SET operation changes the selected application queue space or creates a new one. The following list describes the meaning of the `tuxTQspaceState` object returned by a SET request. States not listed cannot be set.

`open(3)`

Allocate and initialize shared memory and other IPC resources for the queue space, which is allowed only if the queue space is in the `inactive(1)` state.

`cleaning(5)`

Remove the shared memory and other IPC resources for the queue space, which is allowed only when the queue space is in the `active(4)` or `open(3)` state. Successful return leaves the object in the `inactive(1)` state.

`invalid(6)`

Delete the queue space. An error is reported if the state is `active(4)` or if messages exist on any queues in the queue space. Successful return leaves the object in the `invalid(6)` state.

## tuxTQspaceBlocking

### Syntax

INTEGER

### Access

read-write

### Description

The blocking factor used for disk space management of the queue space. The default when a new queue space is created is 16.

## tuxTQspaceErrQname

### Syntax

*DisplayString* (SIZE(0..127))

### Access

read-write

### Description

Name of the error queue associated with the queue space. If there is no error queue, an empty string is returned by a GET request.

## tuxTQspaceForcelnit

### Syntax

INTEGER { yes(1), no(2) }

### Access

read-write

### Description

This object value determines whether or not to initialize disk pages on new extents for the queue space. The default is not to initialize. Depending on the device type (e.g., regular file or raw slice), initialization can occur even if not requested.

## **tuxTQspaceIpckey**

### Syntax

INTEGER (32769..262143)

### Access

read-write

### Description

The IPC key used to access queue space shared memory.

## **tuxTQspaceMaxMsg**

### Syntax

INTEGER

### Access

read-write

### Description

The maximum number of messages that the queue space can contain.

## **tuxTQspaceMaxPages**

### Syntax

INTEGER

### Access

read-write

### Description

The maximum number of disk pages for all queues in the queue space. Each time the `tuxTQspaceMaxPages` object is increased, a new extent is allocated (see `tuxTQspaceCurExtent`). It is not possible to decrease the number of pages by setting this object to a lower number; an error is reported in this case.

## **tuxTQspaceMaxProc**

### Syntax

INTEGER

### Access

read-write

### Description

The maximum number of processes that can attach to the queue space.

## **tuxTQspaceMaxQueues**

### Syntax

INTEGER

### Access

read-write

### Description

The maximum number of queues that the queue space can contain.

## **tuxTQspaceMaxTrans**

### Syntax

INTEGER

### Access

read-write

### Description

The maximum number of simultaneously active transactions allowed by the queue space.

## **tuxTQspaceCurExtent**

### Syntax

INTEGER

### Access

read-only

### Description

The current number of extents used by the queue space. The largest number allowed is 100. Each time the `tuxTQspaceMaxPages` object is increased, a new extent is allocated.

## **tuxTQspaceCurMsg**

### Syntax

INTEGER

### Access

read-only

### Description

The current number of messages in the queue space. This number can be determined only if the queue space is `open (3)` or `active (4)`, or if the queue space is newly created. If none of the conditions apply, the value -1 is returned.

## **tuxTQspaceCurProc**

### Syntax

INTEGER

### Access

read-only

### Description

The current number of processes accessing the queue space.

## tuxTQspaceCurQueues

### Syntax

INTEGER

### Access

read-only

### Description

The current number of queues existing in the queue space. This number can be determined only if the queue space is `open` (3) or `active` (4), or if the queue space is newly created. If none of these conditions apply, the value -1 is returned.

## tuxTQspaceCurTrans

### Syntax

INTEGER

### Access

read-only

### Description

The current number of outstanding transactions involving the queue space.

## tuxTQspaceHwMsg

### Syntax

INTEGER

### Access

read-only

### Description

The highest number of messages in the queue space since the queue space was last opened. The number is reset to 0 when the queue space state is set to `cleaning` (5).

## **tuxTQspaceHwProc**

### Syntax

INTEGER

### Access

read-only

### Description

The highest number of processes simultaneously attached to the queue space since the queue space was last opened. The number is reset to 0 when the queue space state is set to `cleaning(5)`.

## **tuxTQspaceHwQueues**

### Syntax

INTEGER

### Access

read-only

### Description

The highest number of queues existing in the queue space since the queue space was last opened. The number is reset to 0 when the queue space state is set to `cleaning(5)`.

## **tuxTQspaceHwTrans**

### Syntax

INTEGER

### Access

read-only

### Description

The highest number of outstanding transactions involving the queue space since the queue space was last opened. If the queue space is accessed by more than one application, this number reflects

all applications — not just the application represented by the TUXCONFIG environment variable. The number is reset to 0 when the queue space state is set to `cleaning(5)`.

## **tuxTQspacePercentInit**

### Syntax

INTEGER (0..100)

### Access

read-only

### Description

The percentage (as an integer between 0 and 100 inclusive) of disk space that has been initialized for the queue space.

## **tuxTQspaceMaxActions**

### Syntax

INTEGER

### Access

read-write

### Description

This object specifies the number of additional actions that the Queuing Services component of the Oracle Engine can handle concurrently. When a blocking operation is encountered and additional actions are available, the blocking operation is set aside until it can be satisfied. After setting aside the blocking operation, another operation request can be handled. When the blocking operation is completed, the action associated with the operation is made available for a subsequent operation. The system reserve actions are equivalent to the number of processes that can attach to a queue space, so that each queue manager process can have at least one blocking action. Beyond the system-reserved number of blocking actions, the administrator can configure the system to enable it to accommodate additional blocking actions beyond the reserve. An operation fails if a blocking operation is requested and cannot be immediately satisfied and there are no actions available.

## **tuxTQspaceMaxHandles**

### Syntax

INTEGER

### Access

read-write

### Description

This object specifies the number of handles that users of the Queuing Services component of the Oracle Engine can use concurrently. Objects manipulated by the queuing services API require handles to access the objects. When an object is opened by a call to the Queuing Services API, a new handle is created and returned to the user. When an object handle is closed, the handle is made available for subsequent open object operations. When the Queuing Services API is used by an application, the administrator must configure the system to accommodate the maximum number of handles that are opened concurrently. An operation fails if a user attempts to open a queuing services object and there are no handles available. Adjusting this value has no effect on Tuxedo applications other than unnecessarily consuming shared memory resources.

## **tuxTQspaceMaxOwners**

### Syntax

INTEGER

### Access

read-write

### Description

This object specifies the number of additional Oracle Engine authenticated users that can concurrently use Queuing Services resources. There is one owner record per user, regardless of the number of open handles for the user. When there are no open handles for a user, the owner record is made available to subsequent users. The system reserves a number of owners equivalent to the number of actions, so that each action can be initiated by a different owner. Beyond the system-reserved number of owners that can concurrently use queuing services resources, the administrator can configure the system to accommodate additional owners beyond the reserved number. An operation fails if a user attempts to open a handle when there currently are no open handles, and there are no owners available. Adjusting this value has no effect on Oracle Tuxedo

applications other than unnecessarily consuming shared memory resources. Adjusting this value has no effect on Tuxedo applications other than unnecessarily consuming shared memory resources.

## **tuxTQspaceMaxTmpQueues**

### Syntax

INTEGER

### Access

read-write

### Description

This object specifies the number of temporary queues that can be opened concurrently in the Queuing Services component of the Oracle Engine. Temporary queues are used by dynamic, self-configuring applications and reduce the need for administrators to configure each queue used by an application. Messages enqueued to temporary queues are not persistent. When all handles to a temporary queue are closed, the temporary queue resources are made available for subsequent temporary queue creation. When the temporary queues are used by an application, the administrator must configure the system to accommodate the maximum number of temporary queues that are active concurrently. An open operation fails if a user attempts to open a temporary queue and there are no temporary queue resources available. This object specifies the number of additional Oracle Engine authenticated users that can concurrently use Queuing Services

## **tuxTQspaceMaxCursors**

### Syntax

INTEGER

### Access

read-write

### Description

This object specifies the number of cursors that user of the Queuing Services component of the Oracle Engine can use concurrently. CURsors are used to navigate a queue. When a cursor is destroyed, the cursor resources are made available for subsequent cursor creation operations.

When the cursors are used by an application, the administrator must configure the system to accommodate the maximum number of cursors that are allocated concurrently. An operation fails if a user attempts to create a cursor and there are no cursor resources available. This object specifies the number of additional Oracle Engine authenticated users that can concurrently use Queuing Services.

## **tuxTQspaceMemNonPersist**

### Syntax

*DisplayString*

### Access

read-write

### Description

This object specifies the size of the area reserved in shared memory to hold non-persistent messages for all queues in the queue space. The memory size can be specified in bytes (b) or blocks (B). (The size of a block in this context is equivalent to the size of a disk block.)

The [bB] suffix is optional and, if not specified, the default is blocks. Note that the number of bytes requested can be rounded up to the next internal data size. When read, the value is always the actual amount of memory allocated in bytes (b).

All non-persistent messages in the specified queue space are permanently lost when this variable is successfully changed.

If the variable for a queue space is zero (0), no queue space is reserved. for non-persistent messages. In this case, any attempt to enqueue a non-persistent message fails. This type of failure results, for example, when no delivery quality of service has been specified for a message and the `tuxTAppQDefDeliverPolicy` object of the `tuxTAppTbl` group for the target queue has been set to `NONPERSIST`. For non-persistent delivery, if the memory area is exhausted or fragmented so that a message cannot be enqueued, the enqueueing operation fails, even if there is sufficient persistent storage for the message. Similarly, if the persistent storage area is exhausted or fragmented so that a message cannot be enqueued, the enqueueing operation fails, even if there is sufficient non-persistent storage for the message.

## tuxTQspaceMemFilters

### Syntax

INTEGER

### Access

read-write

### Description

This object specifies the size of the memory area to reserve in shared memory to hold the compiled representation of user-defined filters. The memory size is specified in bytes. Filters are used by the Queuing Services component of the Oracle Engine for message selection in dequeuing and cursor operations. Filters can be specified using various grammars, but are compiled into an engine normal form and stored in shared memory. Filters are referenced by a handle that is returned when they are compiled. When a filter is destroyed, the memory used by the filter is made available for subsequent compiled filters. When the filters are defined by an application, the administrator must configure the system to accommodate the maximum number of filters that will be concurrently compiled. An operation fails if a user attempts to create a new filter and there is not enough memory allocated for the compiled version of the filter. Adjusting this value has no effect on Tuxedo applications other than unnecessarily consuming shared memory resources.

## tuxTQspaceMemOverflow

### Syntax

INTEGER

### Access

read-write

### Description

This object specifies the size of the memory area to reserve in shared memory to accommodate peak load situations where some or all of the allocated shared memory resources are exhausted. The memory size is specified in bytes. Additional objects are allocated from this additional memory on a first-come, first-served basis. When an object created in the additional memory is closed or destroyed, the memory is released for subsequent overflow situations. This additional memory space can yield more objects than the configured number, but there is no guarantee that

additional memory is available for any particular object at any given point in time. Currently, only actions, handles, cursors, owners, temporary queues, timers, and filters use the overflow.

## **tuxTQspaceMemSystemReserved**

### Syntax

INTEGER

### Access

read-only

### Description

This object specifies the total amount of memory (in bytes) reserved from shared memory for queuing services system use.

## **tuxTQspaceMemTotalAllocated**

### Syntax

INTEGER

### Access

read-only

### Description

This object specifies the total amount of memory (in bytes) allocated from shared memory for all queuing services objects.

## **tuxTQspaceCurActions**

### Syntax

INTEGER

### Access

read-only

## Description

This object specifies the current number of actions in use in the queue space. This number can be determined if the queue space is `OPEn` or `ACTive`, or if the queue space is newly created. If none of the conditions apply, the value `-1` is returned.

## **tuxTQspaceCurHandles**

### Syntax

INTEGER

### Access

read-only

## Description

This object specifies the current number of cursors in use in the queue space. This number can be determined if the queue space is `OPEn` or `ACTive`, or if the queue space is newly created. If none of the conditions apply, the value `-1` is returned.

## **tuxTQspaceCurOwners**

### Syntax

INTEGER

### Access

read-only

## Description

This object specifies the current number of owners in use in the queue space. This number can be determined if the queue space is `OPEn` or `ACTive`, or if the queue space is newly created. If none of the conditions apply, the value `-1` is returned.

## **tuxTQspaceCurTmpQueues**

### Syntax

INTEGER

## Access

read-only

## Description

This object specifies the current number of temporary queues in use in the queue space. This number can be determined if the queue space is `OPEN` or `ACTIVE`, or if the queue space is newly created. If none of the conditions apply, the value `-1` is returned.

## **tuxTQspaceCurCursors**

### Syntax

INTEGER

### Access

read-only

### Description

This object specifies the current number of cursors in use in the queue space. This number can be determined if the queue space is `OPEN` or `ACTIVE`, or if the queue space is newly created. If none of the conditions apply, the value `-1` is returned.

## **tuxTQspaceCurMemNonPersist**

### Syntax

INTEGER

### Access

read-only

### Description

This object specifies the current amount of memory, in bytes, consumed by non-persistent messages in the queue space. This number can be determined if the queue space is `OPEN` or `ACTIVE`, or if the queue space is newly created. If none of the conditions apply, the value `-1` is returned.

## tuxTQspaceCurMemFilters

### Syntax

INTEGER

### Access

read-only

### Description

This object specifies the current number of bytes in use for filters in the queue space. This number can be determined if the queue space is **OPEn** or **ACTive**, or if the queue space is newly created. If none of the conditions apply, the value **-1** is returned.

## tuxTQspaceCurMemOverFlow

### Syntax

INTEGER

### Access

read-only

### Description

This object specifies the current number of bytes of overflow memory in use in the queue space. This number can be determined if the queue space is **OPEn** or **ACTive**, or if the queue space is newly created. If none of the conditions apply, the value **-1** is returned.

## tuxTQspaceHwActions

### Syntax

INTEGER (0..100)

### Access

read-only

## Description

This object specifies the highest number of concurrent actions reached in the queue space since the queue space was last opened. The number is reset to 0 when the queue space is set to CLEaning.

## **tuxTQspaceHwHandles**

### Syntax

INTEGER (0..100)

### Access

read-only

## Description

This object specifies the highest number of concurrent handles opened in the queue space since the queue space was last opened. The number is reset to 0 when the queue space is set to CLEaning.

## **tuxTQspaceHwOwners**

### Syntax

INTEGER

### Access

read-only

## Description

This object specifies the highest number of concurrent owners reached in the queue space since the queue space was last opened. The number is reset to 0 when the queue space is set to CLEaning.

## **tuxTQspaceHwTmpQueues**

### Syntax

INTEGER

## Access

read-only

## Description

This object specifies the highest number of concurrent temporary queues opened in the queue space since the queue space was last opened. The number is reset to 0 when the queue space is set to CLEaning.

## **tuxTQspaceHwCursors**

### Syntax

INTEGER

### Access

read-only

### Description

This object specifies the highest number of concurrent cursors created in the queue space since the queue space was last opened. The number is reset to 0 when the queue space is set to CLEaning.

## **tuxTQspaceHwMemNonPersist**

### Syntax

INTEGER

### Access

read-only

### Description

This object specifies the largest amount of memory in bytes consumed by non-persistent messages since the queue space was last opened. The number is reset to 0 when the queue space is set to CLEaning.

## tuxTQspaceHwMemFilters

### Syntax

INTEGER

### Access

read-only

### Description

This object specifies the highest number of bytes used for filters in the queue space since the queue space was last opened. The number is reset to 0 when the queue space is set to CLEaning.

## tuxTQspaceHwMemOverflow

### Syntax

INTEGER

### Access

read-only

### Description

This object specifies the highest number of bytes used in the overflow memory in the queue space since the queue space was last opened. The number is reset to 0 when the queue space is set to CLEaning.

## tuxTQtransTbl

The tuxTQtransTbl group contains objects that represent run-time characteristics of transactions associated with application queue spaces. Objects in this table are only accessible through a Tuxedo SNMP agent installed on the local machine.

Object Name	Object ID
tuxTQtransXid	.1.3.6.1.4.1.140.300.12.4.1.1
tuxTQtransIndx1	.1.3.6.1.4.1.140.300.12.4.1.2

Object Name	Object ID
tuxTQtransIndx2	.1.3.6.1.4.1.140.300.12.4.1.3
tuxTQtransIndx3	.1.3.6.1.4.1.140.300.12.4.1.4
tuxTQtransIndx4	.1.3.6.1.4.1.140.300.12.4.1.5
tuxTQtransIndx5	.1.3.6.1.4.1.140.300.12.4.1.6
tuxTQtransGrpNo	.1.3.6.1.4.1.140.300.12.4.1.7
tuxTQtranSpaceName	.1.3.6.1.4.1.140.300.12.4.1.8
tuxTQtransQmConfig	.1.3.6.1.4.1.140.300.12.4.1.9
tuxTQtransLmid	.1.3.6.1.4.1.140.300.12.4.1.10
tuxTQtransState	.1.3.6.1.4.1.140.300.12.4.1.11

## tuxTQtransXid

### Syntax

*DisplayString* (SIZE(1..78))

### Access

read-only

### Description

Transaction identifier as returned by `tx_info(3)` and mapped to a string representation. The data in this field should not be interpreted directly by the user except for equality comparison.

## tuxTQtransIndx1

### Syntax

INTEGER

### Access

read-only

## Description

An integer index for `tuxTQtransTbl`. This should not be interpreted by the user. It is used only for uniquely identifying a particular row in this table by the combination of `Indx1` through `Indx5`.

## **tuxTQtransIndx2**

### Syntax

INTEGER

### Access

read-only

## Description

An integer index for `tuxTQtransTbl`. This should not be interpreted by the user. It is used only for uniquely identifying a particular row in this table by the combination of `Indx1` through `Indx5`.

## **tuxTQtransIndx3**

### Syntax

INTEGER

### Access

read-only

## Description

An integer index for `tuxTQtransTbl`. This should not be interpreted by the user. It is used only for uniquely identifying a particular row in this table by the combination of `Indx1` through `Indx5`.

## **tuxTQtransIndx4**

### Syntax

INTEGER

### Access

read-only

## Description

An integer index for `tuxTQtransTbl`. This should not be interpreted by the user. It is used only for uniquely identifying a particular row in this table by the combination of `Indx1` through `Indx5`.

## **tuxTQtransIndx5**

### Syntax

INTEGER

### Access

read-only

## Description

An integer index for `tuxTQtransTbl`. This should not be interpreted by the user. It is used only for uniquely identifying a particular row in this table by the combination of `Indx1` through `Indx5`.

## **tuxTQtransGrpNo**

### Syntax

INTEGER

### Access

read-only

## Description

Group number of any server group for which the queue space concerning this transaction is a resource manager, in other words that group's `openinfo` string `tuxTgroupOpenInfo` contains the device name and queue space name for the queue space concerning this transaction.

## **tuxTQtranSpaceName**

### Syntax

*DisplayString*(SIZE(1..15))

## Access

read-only

## Description

Name of the application queue space associated with the transaction.

## **tuxTQtransQmConfig**

### Syntax

*DisplayString* (SIZE(1..78))

## Access

read-only

## Description

Absolute pathname of the file or device where the application queue space is located.

## **tuxTQtransLmid**

### Syntax

*DisplayString* (SIZE(1..30))

## Access

read-only

## Description

Identifier of the logical machine where the application queue space is located.

## **tuxTQtransState**

### Syntax

INTEGER { active(1), abort-only(2), aborted(3), com-called(4), ready(5),  
decided(6), suspended(7), habort(8), hcommit(9) }

## Access

read-write

## Description

The values for GET and SET operations are as follows:

GET: {active(1) | abort-only(2) | aborted(3) | com-called(4) | ready(5) |  
decided(6) | suspended(7) }

A GET operation retrieves run-time information about the selected transactions. The following list describes the meaning of the `tuxTQtransState` object returned in response to a GET request. States not listed are not returned.

active(1)

The transaction is active.

abort-only(2)

The transaction has been identified for rollback.

aborted(3)

The transaction has been identified for rollback and rollback has been initiated.

com-called(4)

The initiator of the transaction has called `tpcommit(3)` and the first phase of two-phase commit has begun.

ready(5)

All of the participating groups on the retrieval site have successfully completed the first phase of the two-phase commit and are ready to be committed.

decided(6)

The second phase of the two-phase commit has begun.

suspended(7)

The initiator of the transaction has suspended processing on the transaction.

SET: {habort(8) | hcommit(9) }

A SET operation updates the state of the selected transactions. The following list describes the meaning of the `tuxTQtransState` object returned by a SET request. States not listed cannot be set.

habort(8)

Heuristically abort the transaction. Successful return leaves the object in the `habort(8)` state.

hcommit(9)

Heuristically commit the transaction. Successful return leaves the object in the `hcommit(9)` state.



# EventBroker MIB

There are two types of Oracle Tuxedo events: application events and system events. Application events are usually controlled or trapped by the application code. System events are generated by the Tuxedo run-time system when important changes in that system are detected. Application programs (clients or services) can subscribe to these system events.

The EventBroker MIB defines the characteristics of an event subscription. You can use the EventBroker MIB to obtain the characteristics of current event subscriptions, define new subscriptions, or invalidate subscriptions. To enable both system event and application event notification, you need to define the system event broker and the application event broker in the Core MIB.

Event subscriptions can be temporary or persistent. Persistent subscriptions survive across application activations and can be removed through the EventBroker MIB. The Tuxedo EventBroker MIB contains five groups of event subscriptions through which the EventBroker can be managed.

The EventBroker MIB consists of the following subscription groups.

Group Name	Description
<a href="#">tuxEventClientTbl</a>	Subscriptions that trigger unsolicited notification
<a href="#">tuxEventCmdTbl</a>	Subscriptions that trigger system commands
<a href="#">tuxEventQueTbl</a>	Subscriptions for queue-based notification

Group Name	Description
<a href="#">tuxEventSvcTbl</a>	Subscriptions for server-based notification
<a href="#">tuxEventUlogTbl</a>	Subscriptions for writing userlog messages

Each object in these groups represents a single subscription request. Client Notifications ([tuxEventClientTbl](#) group) indicate which events trigger an unsolicited message to a client. Service Notifications ([tuxEventSvcTbl](#) group) indicate which events trigger a request to an application service. Application Queue Notifications ([tuxEventQueTbl](#) group) indicate which events send a message to an application queue. System Command Notifications ([tuxEventCmdTbl](#) group) indicate which events trigger an operating system command. Log File Notifications ([tuxEventUlogTbl](#) group) indicate which events generate a record in the central event log (ulog). The EventBroker automatically removes temporary subscriptions when it detects that the corresponding target is no longer active.

Event subscriptions and the ability to change the Tuxedo MIB enables system administrators and application designers to write event-adaptive applications. When a failure is detected through a system event notification, a management framework program can perform the corrective measures. For example, a management framework task can be triggered to activate servers on a backup machine when it receives an event notification about a failure on a primary machine.

## tuxEventClientTbl

This group contains objects that represent subscriptions registered with the EventBroker for client-based notification.

When an event is detected, it is compared to each `tuxEventClientTbl` instance. If the event name matches the value in the event expression and the optional filter rule is true, then the event buffer is sent to the specified client's unsolicited message handling routine. To create a new row in this table, it is necessary to issue a SET request that at least specifies the values for `tuxEventClientExpr` and `tuxEventClientId`.

Object Name	Object ID
<code>tuxEventClientIndx</code>	.1.3.6.1.4.1.140.300.2.1.1.1.1
<code>tuxEventClientExpr</code>	.1.3.6.1.4.1.140.300.2.1.1.1.2

Object Name	Object ID
tuxEventClientFilter	.1.3.6.1.4.1.140.300.2.1.1.1.3
tuxEventClientState	.1.3.6.1.4.1.140.300.2.1.1.1.4
tuxEventClientId	.1.3.6.1.4.1.140.300.2.1.1.1.5

## tuxEventClientIndx

### Syntax

INTEGER

### Access

read-only

### Description

A running number as the unique identifier for a row in the table.

## tuxEventClientExpr

### Syntax

*DisplayString*(SIZE(1..255))

### Access

read-only

### Description

Event pattern expression. This expression, which is a regular expression, controls which event names match this subscription. For the format of regular expressions, see reference page [tpsubscribe\(3c\)](#) in *Oracle Tuxedo ATMI C Function Reference*.

**Note:** This object can be updated only during row creation.

## tuxEventClientFilter

### Syntax

```
DisplayString(SIZE(1..255))
```

### Access

read-only

### Description

Event filter expression. This expression, if present, is evaluated with respect to the posted buffer's contents. It must evaluate to TRUE or this subscription is not matched. If the value of this is "-", it means that the filter expression is in binary format.

**Note:** This object can be updated only during row creation.

## tuxEventClientState

### Syntax

```
INTEGER { active(1), invalid(2) }
```

### Access

read-write

### Description

The values for GET and SET operations are as follows:

GET: active(1)

A GET operation retrieves configuration information for the matching tuxEventClientTbl row(s).

SET: invalid(2)

A SET operation updates configuration information for the row in tuxEventClientTbl. The following state indicates the meaning of a tuxEventClientState set in a SET request. States not listed cannot be set.

invalid(2)

Delete row. Successful return leaves the row in the invalid(2) state.

## tuxEventClientId

### Syntax

*DisplayString*(SIZE(1..78))

### Access

read-only

### Description

Send an unsolicited notification message to this client when a matching event is detected.

**Note:** This object can be updated only during row creation.

## tuxEventCmdTbl

This group contains objects that represent subscriptions registered with the EventBroker that trigger execution of system commands.

When an event is detected, it is compared to each row in this table. If the event name matches the value in the event expression and the optional filter rule is true, then the event buffer is formatted and passed to the system's command interpreter.

**Create a new Row:** To create a new instance of `tuxEventCmdTbl` the user must specify at least `tuxEventCmdExpr` and `tuxEventCmd`. All objects except `tuxEventCmdState` can be updated only during creation of a new instance.

Object Name	Object ID
<code>tuxEventCmdIndx</code>	.1.3.6.1.4.1.140.300.2.2.1.1.1
<code>tuxEventCmdExpr</code>	.1.3.6.1.4.1.140.300.2.2.1.1.2
<code>tuxEventCmdFilter</code>	.1.3.6.1.4.1.140.300.2.2.1.1.3
<code>tuxEventCmdState</code>	.1.3.6.1.4.1.140.300.2.2.1.1.4
<code>tuxEventCmd</code>	.1.3.6.1.4.1.140.300.2.2.1.1.5

## tuxEventCmdIndx

### Syntax

INTEGER

### Access

read-only

### Description

A running number as the unique identifier for a row in the table.

## tuxEventCmdExpr

### Syntax

*DisplayString*(SIZE(1..255))

### Access

read-write

### Description

Event pattern expression. This expression, which is a regular expression, controls which event names match this subscription. For the format of regular expressions, see reference page [tsubscribe\(3c\)](#) in *Oracle Tuxedo ATMI C Function Reference*.

**Note:** This object can be updated only during row creation.

## tuxEventCmdFilter

### Syntax

*DisplayString*(SIZE(1..255))

### Access

read-write

## Description

Event filter expression. This expression, if present, is evaluated with respect to the posted buffer's contents. It must evaluate to TRUE or this subscription is not matched. If the value of the filter is "-", it means that the filter is in a binary format.

**Note:** This object can be updated only during row creation.

## tuxEventCmdState

### Syntax

```
INTEGER { active(1), invalid(2) }
```

### Access

read-write

### Description

The values for GET and SET operations are as follows:

GET: active(1)

A GET operation retrieves configuration information for the tuxEventCmdTbl instance(s).

SET: invalid(2)

A SET operation updates configuration information for the tuxEventCmdTbl instance. The following state indicates the meaning of a tuxEventCmdState set in a SET request. States not listed cannot be set.

invalid(2)

Delete tuxEventCmdTbl instance. Successful return leaves the object in the invalid(2) state.

## tuxEventCmd

### Syntax

```
DisplayString(SIZE(1..255))
```

### Access

read-write

## Description

Execute this system command when an event matching this object is detected. For UNIX system platforms, the command is executed in the background using `system(3)`.

**Note:** This object can be updated only during row creation.

## tuxEventQueTbl

This group contains objects that represent subscriptions registered with the EventBroker for queue-based notification.

When an event is detected, it is compared to each `tuxEventQueTbl` instance. If the event name matches the value in the event expression and the optional filter rule is true, then the event buffer is stored in the specified reliable queue. To create a new row in this table, it is necessary to issue a SET request that at least specifies `tuxEventQueExpr`, `tuxEventQspace`, and `tuxEventQname`.

Object Name	Object ID
<code>tuxEventQueIndx</code>	.1.3.6.1.4.1.140.300.2.3.1.1.1
<code>tuxEventQueExpr</code>	.1.3.6.1.4.1.140.300.2.3.1.1.2
<code>tuxEventQueFilter</code>	.1.3.6.1.4.1.140.300.2.3.1.1.3
<code>tuxEventQueState</code>	.1.3.6.1.4.1.140.300.2.3.1.1.4
<code>tuxEventQspace</code>	.1.3.6.1.4.1.140.300.2.3.1.1.5
<code>tuxEventQname</code>	.1.3.6.1.4.1.140.300.2.3.1.1.6
<code>tuxEventQctlQtop</code>	.1.3.6.1.4.1.140.300.2.3.1.1.7
<code>tuxEventQctlBeforeMsgid</code>	.1.3.6.1.4.1.140.300.2.3.1.1.8
<code>tuxEventQctlQtimeAbs</code>	.1.3.6.1.4.1.140.300.2.3.1.1.9
<code>tuxEventQctlQtimeRel</code>	.1.3.6.1.4.1.140.300.2.3.1.1.10
<code>tuxEventQctlDeqTime</code>	.1.3.6.1.4.1.140.300.2.3.1.1.11
<code>tuxEventQctlPrior</code>	.1.3.6.1.4.1.140.300.2.3.1.1.12
<code>tuxEventQctlMsgId</code>	.1.3.6.1.4.1.140.300.2.3.1.1.13

Object Name	Object ID
tuxEventQctlCorrId	.1.3.6.1.4.1.140.300.2.3.1.1.14
tuxEventQctlReplyQ	.1.3.6.1.4.1.140.300.2.3.1.1.15
tuxEventQctlFailQ	.1.3.6.1.4.1.140.300.2.3.1.1.16
tuxEventPersist	.1.3.6.1.4.1.140.300.2.3.1.1.17
tuxEventTran	.1.3.6.1.4.1.140.300.2.3.1.1.18

## tuxEventQueIndx

### Syntax

INTEGER

### Access

read-only

### Description

Running number which is the unique identifier for an event in this table.

## tuxEventQueExpr

### Syntax

*DisplayString* (SIZE(1..255))

### Access

read-write

### Description

Event pattern expression. This expression, which is a regular expression, controls which event names match this subscription. For the format of regular expressions, see reference page [tpsubscribe\(3c\)](#) in *Oracle Tuxedo ATMI C Function Reference*.

**Note:** This object can be updated only during row creation.

## tuxEventQueFilter

### Syntax

```
DisplayString(SIZE(1..255))
```

### Access

read-write

### Description

Event filter expression. This expression, if present, is evaluated with respect to the posted buffer's contents. It must evaluate to TRUE or this subscription is not matched. If the value of this object is "-", it means the filter is in binary format.

**Note:** This object can be updated only during row creation.

## tuxEventQueState

### Syntax

```
INTEGER { active(1), invalid(2) }
```

### Access

read-write

### Description

The values for GET and SET operations are as follows:

GET: active(1)

A GET operation retrieves configuration information for the matching tuxEventQueTbl row(s).

SET: invalid(2)

A SET operation updates configuration information for the tuxEventQueTbl instance. The following state indicates the meaning of a tuxEventQueState set in a SET request. States not listed cannot be set.

invalid(2)

Delete tuxEventQueTbl row. Successful return leaves the object in the invalid(2) state.

## tuxEventQspace

### Syntax

*DisplayString*(SIZE(1..127))

### Access

read-write

### Description

Enqueue a notification message to a reliable queue in this queue space when a matching event is detected.

**Note:** This object can be updated only during row creation.

## tuxEventQname

### Syntax

*DisplayString*(SIZE(1..127))

### Access

read-write

### Description

Enqueue a notification message to this reliable queue when a matching event is detected.

**Note:** This object can be updated only during row creation.

## tuxEventQctlQtop

### Syntax

INTEGER

### Access

read-write

## Description

This value, if present, is passed in to `topenqueue(3)`'s TPQCTL control structure to request notification via the /Q subsystem with the message to be placed at the top of the queue.

**Note:** This object can be updated only during row creation.

## **tuxEventQctlBeforeMsgid**

### Syntax

INTEGER

### Access

read-write

## Description

This value, if present, is passed in to `topenqueue(3)`'s TPQCTL control structure to request notification via the /Q subsystem with the message to be placed on the queue ahead of the specified message.

**Note:** This object can be updated only during row creation.

## **tuxEventQctlQtimeAbs**

### Syntax

INTEGER

### Access

read-write

## Description

This value, if present, is passed in to `topenqueue(3)`'s TPQCTL control structure to request notification via the /Q subsystem with the message to be processed at the specified time.

**Note:** This object can be updated only during row creation.

## tuxEventQctIqtimeRel

### Syntax

INTEGER

### Access

read-write

### Description

This value, if present, is passed in to `topenqueue(3)`'s TPQCTL control structure to request notification via the /Q subsystem with the message to be processed relative to the dequeue time.

**Note:** This object can be updated only during row creation.

## tuxEventQctIDeqTime

### Syntax

INTEGER

### Access

read-write

### Description

This value, if present, is passed in to `topenqueue(3)`'s TPQCTL control structure.

**Note:** This object can be updated only during row creation.

## tuxEventQctIPrior

### Syntax

INTEGER

### Access

read-write

### Description

This value, if present, is passed in to `topenqueue(3)`'s TPQCTL control structure.

**Note:** This object can be updated only during row creation.

## tuxEventQctlMsgId

### Syntax

*DisplayString*(SIZE(1..31))

### Access

read-write

### Description

This value, if present, is passed in to `topenqueue(3)`'s TPQCTL control structure.

**Note:** This object can be updated only during row creation.

## tuxEventQctlCorrId

### Syntax

*DisplayString*(SIZE(1..31))

### Access

read-write

### Description

This value, if present, is passed in to `topenqueue(3)`'s TPQCTL control structure.

**Note:** This object can be updated only during row creation.

## tuxEventQctlReplyQ

### Syntax

*DisplayString*(SIZE(1..127))

### Access

read-write

## Description

This value, if present, is passed in to `topenqueue(3)`'s TPQCTL control structure.

**Note:** This object can be updated only during row creation.

## tuxEventQctIFailQ

### Syntax

*DisplayString(SIZE(1..127))*

### Access

read-write

## Description

This value, if present, is passed in to `topenqueue(3)`'s TPQCTL control structure.

**Note:** This object can be updated only during row creation.

## tuxEventPersist

### Syntax

INTEGER

### Access

read-write

## Description

If non-zero, do not cancel this subscription if the designated queue is no longer available.

**Note:** This object can be updated only during row creation.

## tuxEventTran

### Syntax

INTEGER

## Access

read-write

## Description

If non-zero and the client's `tppost(3)` call is transactional, include the `tpenqueue(3)` call in the client's transaction.

**Note:** This object can be updated only during row creation.

# tuxEventSvcTbl

This group contains objects that represent subscriptions registered with the EventBroker for service-based notification.

When an event is detected, it is compared to each `tuxEventSvcTbl` instance. If the event name matches the value in the event expression and the optional filter rule is true, then the event buffer is sent to the specified Tuxedo service routine.

To create a new row in this table, a `SET` request must be issued that specifies values for at least `tuxEventSvcExpr` and `tuxEventSvcName`.

Object Name	Object ID
<code>tuxEventSvcIndx</code>	.1.3.6.1.4.1.140.300.2.4.1.1.1
<code>tuxEventSvcExpr</code>	.1.3.6.1.4.1.140.300.2.4.1.1.2
<code>tuxEventSvcFilter</code>	.1.3.6.1.4.1.140.300.2.4.1.1.3
<code>tuxEventSvcState</code>	.1.3.6.1.4.1.140.300.2.4.1.1.4
<code>tuxEventSvcName</code>	.1.3.6.1.4.1.140.300.2.4.1.1.5
<code>tuxEventSvcPersist</code>	.1.3.6.1.4.1.140.300.2.4.1.1.6
<code>tuxEventSvcTran</code>	.1.3.6.1.4.1.140.300.2.4.1.1.7

## tuxEventSvcIndx

### Syntax

INTEGER

## Access

read-only

## Description

A running number which is a unique key for a row in this table.

## tuxEventSvcExpr

### Syntax

*DisplayString* (SIZE(1..255))

### Access

read-only

### Description

Event pattern expression. This expression, which is a regular expression, controls which event names match this subscription. For the format of regular expressions, see reference page [tsubscribe\(3c\)](#) in *Oracle Tuxedo ATMI C Function Reference*.

**Note:** This object can be updated only during row creation.

## tuxEventSvcFilter

### Syntax

*DisplayString* (SIZE(1..255))

### Access

read-only

### Description

Event filter expression. This expression, if present, is evaluated with respect to the posted buffer's contents. It must evaluate to TRUE or this subscription is not matched. If this is "-", it means the filter is in binary format.

**Note:** This object can be updated only during row creation.

## tuxEventSvcState

### Syntax

```
INTEGER { active(1), invalid(2) }
```

### Access

read-write

### Description

The values for GET and SET operations are as follows:

GET: active(1)

A GET operation retrieves configuration information for the matching tuxEventSvcTbl instance(s).

SET: invalid(2)

A SET operation updates configuration information for the tuxEventSvcTbl instance. The following state indicates the meaning of a tuxEventSvcState set in a SET request. States not listed cannot be set.

invalid(2)

Delete tuxEventSvcTbl row. Successful return leaves the object in the invalid(2) state.

## tuxEventSvcName

### Syntax

```
DisplayString(SIZE(1..127))
```

### Access

read-only

### Description

Call this Tuxedo service when a matching event is detected.

**Note:** This object can be updated only during row creation.

## tuxEventSvcPersist

### Syntax

INTEGER

### Access

read-write

### Description

If non-zero, do not cancel this subscription if the `tuxEventSvcName` service is no longer available.

**Note:** This object can be updated only during row creation.

## tuxEventSvcTran

### Syntax

INTEGER

### Access

read-write

### Description

If non-zero and the client's `tppost(3)` call is transactional, include the `tuxEventSvcName` service call in the client's transaction.

**Note:** This object can be updated only during row creation.

## tuxEventUlogTbl

This group contains objects that represent subscriptions registered with the EventBroker for writing system `userlog(3)` messages.

When an event is detected, it is compared to each `tuxEventUlogTbl` instance. If the event name matches the value in the event expression and the optional filter rule is true, then the event buffer is formatted and passed to the Tuxedo `userlog(3)` function.

Create a new Row: To create a new instance of `tuxEventUlogTbl` the user must at least specify values for `tuxEventUlogExpr` and `tuxEventUserlog`. All objects except `tuxEventUlogState` can be updated only during creation of a new instance.

Object Name	Object ID
<code>tuxEventUlogIndx</code>	.1.3.6.1.4.1.140.300.2.5.1.1.1
<code>tuxEventUlogExpr</code>	.1.3.6.1.4.1.140.300.2.5.1.1.2
<code>tuxEventUlogFilter</code>	.1.3.6.1.4.1.140.300.2.5.1.1.3
<code>tuxEventUlogState</code>	.1.3.6.1.4.1.140.300.2.5.1.1.4
<code>tuxEventUserlog</code>	.1.3.6.1.4.1.140.300.2.5.1.1.5

## **tuxEventUlogIndx**

### Syntax

INTEGER

### Access

read-only

### Description

A running number which is a unique key in this table.

## **tuxEventUlogExpr**

### Syntax

*DisplayString*(SIZE(1..255))

### Access

read-write

## Description

Event pattern expression. This expression, which is a regular expression, controls which event names match this subscription. For the format of regular expressions, see reference page [tsubscribe\(3c\)](#) in *Oracle Tuxedo ATMI C Function Reference*.

**Note:** This object can be updated only during row creation.

## tuxEventUlogFilter

### Syntax

```
DisplayString(SIZE(1..255))
```

### Access

read-write

### Description

Event filter expression. This expression, if present, is evaluated with respect to the posted buffer's contents. It must evaluate to TRUE or this subscription is not matched. If this is "-", it means the filter is in binary form.

**Note:** This object can be updated only during row creation.

## tuxEventUlogState

### Syntax

```
INTEGER { active(1), invalid(2) }
```

### Access

read-write

### Description

The values for GET and SET operations are as follows:

GET: active(1)

A GET operation retrieves configuration information for the matching tuxEventUlogTbl instance(s).

SET: `invalid(2)`

A SET operation updates configuration information for the `tuxEventUlogTbl` instance. The following state indicates the meaning of a `tuxEventUlogState` set in a SET request. States not listed cannot be set.

`invalid(2)`

Delete `tuxEventUlogTbl` row. Successful return leaves the object in the `invalid(2)` state.

## **tuxEventUserlog**

### Syntax

*DisplayString*(SIZE(1..255))

### Access

read-write

### Description

Write a `userlog(3)` message when a matching event is detected.

**Note:** This object can be updated only during row creation.

# Traps MIB

The event monitor feature of Oracle Tuxedo systems detects and reports certain predefined events—primarily failures of which a system operator should be aware. Oracle SNMP Agent on the master node subscribes to all system events and generates a corresponding SNMP trap notification whenever any of these events occur. The enterprise ID used for these traps is `.1.3.6.1.4.1.140.tuxedo`, where `tuxedo` is 300. For Oracle SNMP Agent to receive Tuxedo system events, the Tuxedo system EventBroker (`TMSYSEVT`) must be running because that is the entity that generates the system events.

The Event Traps MIB defines all the traps that are generated and the objects that are passed in the variable bindings for these traps. The following sections describe the cause and recommended action for each event:

- [Specific Trap Number](#)
- [Variable Bindings](#)
- [Trap Definitions](#)

## Specific Trap Number

Each enterprise-specific trap notification generated by Oracle SNMP Agent has a value in the specific trap ID field of the SNMP trap packet that identifies the Tuxedo event. For each trap listed in this chapter, “Trap ID” is the specific trap number that is sent in the trap packet.

## Variable Bindings

SNMP trap notifications generated by Oracle SNMP Agent contain 12 variables (variable/value pairs) in the variable bindings of the trap packet:

`beaEventsDomainId`

The ID of the domain that generated the Tuxedo event notification.

`beaEventsIpcKey`

The IPC key of the Tuxedo domain.

`beaLogicalAgentName`

The logical agent name of the SNMP agent for Oracle SNMP Agent generating the trap. The executable name is the default logical agent name.

The `tuxEventTrapVars` group contains all objects that are sent as a part of the variable bindings of the traps generated in relation to Tuxedo system events, as defined in EVENTS.

### **tuxEventsName**

#### Syntax

*DisplayString*

#### Access

not-accessible

#### Description

A string that uniquely identifies this event. All system-generated events begin with `.Sys`.

### **tuxEventsSeverity**

#### Syntax

INTEGER { Error(1), Warn(2), Infor(3) }

#### Access

not-accessible

#### Description

Indicates the severity of the system event.

## **tuxEventsLmid**

### Syntax

```
DisplayString( SIZE(1..30) )
```

### Access

not-accessible

### Description

A string that identifies the machine where the event was detected.

## **tuxEventsTime**

### Syntax

```
INTEGER
```

### Access

not-accessible

### Description

A long integer containing the event detection time, in seconds, according to the clock on the machine where detection took place.

## **tuxEventsUsec**

### Syntax

```
INTEGER
```

### Access

not-accessible

### Description

A long integer containing the event detection time, in microseconds, according to the clock on the machine where detection took place. While the units of this value are always microseconds, the actual resolution depends on the underlying operating system and hardware.

## **tuxEventsDescription**

### Syntax

*DisplayString*

### Access

not-accessible

### Description

A one-line string summarizing the event.

## **tuxEventsClass**

### Syntax

*DisplayString*

### Access

not-accessible

### Description

The class of the object associated with the event. Depending on TA\_CLASS, the event notification buffer can contain additional fields specific to an object of this class.

## **tuxEventsUlogCat**

### Syntax

*DisplayString*

### Access

not-accessible

### Description

Catalog name from which the message was derived, if any.

## **tuxEventsUlogMsgNum**

### Syntax

INTEGER

### Access

not-accessible

### Description

Catalog message number, if the message was derived from a catalog.

## **tuxTdomainID**

### Syntax

*DisplayString*(SIZE (0..30))

### Access

not-accessible

### Description

Domain identification string. Refer to [Chapter 2, “Core MIB.”](#)

## **tuxTdomainKey**

### Syntax

INTEGER (32769..262143)

### Access

not-accessible

### Description

Numeric key for the well-known address in a Tuxedo system bulletin board. In a single processor environment, this key “names” the bulletin board. In a multiple processor or LAN environment, this key names the message queue of the DBBL. In addition, this key is used as a basis for deriving the names of resources other than the well-known address, such as the names for bulletin boards throughout the application. Refer to [Chapter 2, “Core MIB.”](#)

## beaLogicalAgentName

### Syntax

*DisplayString*

### Access

not-accessible

### Description

The logical name of the agent as provided in the `-l` option (service name in case of Windows NT) when the agent was started. This object value is the agent that is monitoring this domain. If there are multiple SNMP agents running on a managed node, this name needs to be appended to the community with an `@` sign to get the MIB values from the appropriate agent. For example, if there are two logical agents, `simp_snmpd` and `bank_snmpd`, the communities used to query values from these agents would be `public@simp_snmpd` and `public@bank_snmpd` respectively. The component after the `@` sign is used to identify the agent to which the MIB query is to be sent.

This object is passed in the variable binding of all SNMP traps generated on behalf of Tuxedo system events.

**Note:** To run multiple SNMP agents on the same managed node, they must be started as subagents (without `-s` option) and run after starting the Oracle SNMP Agent Integrator.

## Trap Definitions

The following sections define all the traps generated by Oracle SNMP Agent when system events occur:

- [DOMAIN Traps](#)
- [MACHINE Traps](#)
- [BRIDGE Traps](#)
- [SERVER Event Traps](#)
- [CLIENT Traps](#)
- [TRANSACTION Traps](#)
- [EVENT Traps](#)

## DOMAIN Traps

The Domain Traps group defines the Tuxedo domain specific event traps.

### resourceConfigTrap

Enterprise

tuxedo

Variables

```
{ tuxEventsName, tuxEventsSeverity, tuxEventsLmid, tuxEventsTime,
  tuxEventsUsec, tuxEventsDescription, tuxEventsClass, tuxEventsUlogCat,
  tuxEventsUlogMsgNum, beaDomainId, beaDomainKey, beaLogicalAgentName }
```

Description

This trap is generated when `.SysResourceConfig` occurs. It denotes a system configuration change.

Action

This message is informational.

Trap ID

1

## MACHINE Traps

The Machine Traps group defines the Tuxedo machine specific event traps.

### machineBroadcastTrap

Enterprise

tuxedo

Variables

```
{ tuxEventsName, tuxEventsSeverity, tuxEventsLmid, tuxEventsTime,
  tuxEventsUsec, tuxEventsDescription, tuxEventsClass, tuxEventsUlogCat,
  tuxEventsUlogMsgNum, beaDomainId, beaDomainKey, beaLogicalAgentName }
```

## Description

This trap is generated when `.SysMachineBroadcast` occurs. It denotes broadcast delivery failure. This message indicates that `tpbroadcast()` failed for at least one accessor on the LMID of the application.

## Action

Since the broadcast messages are sent in no-blocking mode, it is possible that the process doing the broadcasting encountered a blocking condition and dropped a message. Configure larger message queues or load-balance clients and servers such that excessive load is not put on some machines.

## Trap ID

2

## **machineConfigTrap**

### Enterprise

tuxedo

### Variables

```
{ tuxEventsName, tuxEventsSeverity, tuxEventsLmid, tuxEventsTime,  
tuxEventsUsec, tuxEventsDescription, tuxEventsClass, tuxEventsUlogCat,  
tuxEventsUlogMsgNum, beaDomainId, beaDomainKey, beaLogicalAgentName }
```

## Description

This trap is generated when `.SysMachineConfig` occurs. It denotes a change in a particular machine configuration.

## Action

This message is informational.

## Trap ID

3

## machineFullMaxAccessersTrap

### Enterprise

tuxedo

### Variables

```
{ tuxEventsName, tuxEventsSeverity, tuxEventsLmid, tuxEventsTime,  
tuxEventsUsec, tuxEventsDescription, tuxEventsClass, tuxEventsUlogCat,  
tuxEventsUlogMsgNum, beaDomainId, beaDomainKey, beaLogicalAgentName }
```

### Description

This trap is generated when `.SysMachineFullMaxaccessers` occurs. This message indicates that the given LMID reached the capacity limit on the number of accessers.

### Action

Increase the MAXACCESSERS value for the particular machine. Or, if the hardware/software limits have been reached for the maximum number of users on the machine, move additional users to other machines.

### Trap ID

4

## machineFullMaxConvTrap

### Enterprise

tuxedo

### Variables

```
{ tuxEventsName, tuxEventsSeverity, tuxEventsLmid, tuxEventsTime,  
tuxEventsUsec, tuxEventsDescription, tuxEventsClass, tuxEventsUlogCat,  
tuxEventsUlogMsgNum, beaDomainId, beaDomainKey, beaLogicalAgentName }
```

### Description

This trap is generated when `.SysMachineFullMaxconv` occurs. This message indicates that the given LMID reached the capacity limit on the number of concurrent conversations.

## Action

Increase the value of MAXCONV for the particular machine to the point that this event is not generated.

## Trap ID

5

## **machineFullMaxGttTrap**

### Enterprise

tuxedo

### Variables

```
{ tuxEventsName, tuxEventsSeverity, tuxEventsLmid, tuxEventsTime,  
tuxEventsUsec, tuxEventsDescription, tuxEventsClass, tuxEventsUlogCat,  
tuxEventsUlogMsgNum, beaDomainId, beaDomainKey, beaLogicalAgentName }
```

### Description

This trap is generated when `.SysMachineFullMaxgtt` is raised. This message indicates that the given machine reached the capacity limit on the number of concurrent transactions.

## Action

Increase the value of MAXGTT for the particular machine to the point that this event is not generated.

## Trap ID

6

## **machineFullMaxWsClientsTrap**

### Enterprise

tuxedo

### Variables

```
{ tuxEventsName, tuxEventsSeverity, tuxEventsLmid, tuxEventsTime,  
tuxEventsUsec, tuxEventsDescription, tuxEventsClass, tuxEventsUlogCat,  
tuxEventsUlogMsgNum, beaDomainId, beaDomainKey, beaLogicalAgentName }
```

## Description

This trap is generated when `.SysMachineFullMaxwsclients` is raised. This message indicates that the given machine reached the capacity limit on the number of workstation clients.

## Action

Increase the value of `MAXWSCLIENTS` for the particular machine to the point that this event is not generated.

## Trap ID

7

## machineMsgQTrap

### Enterprise

tuxedo

### Variables

```
{ tuxEventsName, tuxEventsSeverity, tuxEventsLmid, tuxEventsTime,
  tuxEventsUsec, tuxEventsDescription, tuxEventsClass, tuxEventsUlogCat,
  tuxEventsUlogMsgNum, beaDomainId, beaDomainKey, beaLogicalAgentName }
```

## Description

This trap is generated when `.SysMachineMsgq` occurs. This message indicates that the server posting a message encountered a blocking condition while putting a message on the message queue.

## Action

Configure larger message queues and/or distribute the load equally on all the machines.

## Trap ID

8

## machinePartitionedTrap

### Enterprise

tuxedo

## Variables

```
{ tuxEventsName, tuxEventsSeverity, tuxEventsLmid, tuxEventsTime,  
tuxEventsUsec, tuxEventsDescription, tuxEventsClass, tuxEventsUlogCat,  
tuxEventsUlogMsgNum, beaDomainId, beaDomainKey, beaLogicalAgentName}
```

## Description

This trap is generated when `.SysMachinePartitioned` occurs. This message indicates that DBBL partitioned the stated machine either because the BBL on the machine is slow or the network link between the master and the machine is broken.

## Action

This can occur due to several reasons:

- The entire network might be bogged down due to heavy traffic.
- The BBL or BRIDGE on the non-master is either dead or slow.
- The BRIDGE process on the non-master is extremely busy.

The software is capable of unpartitioning the machine if things stabilize.

## Trap ID

9

## machineSlowTrap

### Enterprise

tuxedo

## Variables

```
{ tuxEventsName, tuxEventsSeverity, tuxEventsLmid, tuxEventsTime,  
tuxEventsUsec, tuxEventsDescription, tuxEventsClass, tuxEventsUlogCat,  
tuxEventsUlogMsgNum, beaDomainId, beaDomainKey, beaLogicalAgentName}
```

## Description

This trap is generated when `.SysMachineSlow` occurs. This message indicates that BBL on the non-master machine is slow in generating IAMOK messages. These messages are sent periodically from BBLs to the DBBL that helps the DBBL maintain the pulse of the system.

## Action

This can occur due to several reasons:

- The entire network might be bogged down due to heavy traffic.
- The BBL on the non-master might be either dead or slow.
- The BRIDGE process on the non-master is extremely busy.

This problem can be intermittent.

## Trap ID

10

## machineStateTrap

### Enterprise

tuxedo

### Variables

```
{ tuxEventsName, tuxEventsSeverity, tuxEventsLmid, tuxEventsTime,
  tuxEventsUsec, tuxEventsDescription, tuxEventsClass, tuxEventsUlogCat,
  tuxEventsUlogMsgNum, beaDomainId, beaDomainKey, beaLogicalAgentName }
```

### Description

This trap is generated when `.SysMachineState` occurs. This denotes that a particular machine changed its state.

### Action

This message is informational.

## Trap ID

11

## BRIDGE Traps

The Bridge Traps group defines the Tuxedo bridge specific traps.

## networkConfigTrap

### Enterprise

tuxedo

### Variables

```
{tuxEventsName, tuxEventsSeverity, tuxEventsLmid, tuxEventsTime,  
tuxEventsUsec, tuxEventsDescription, tuxEventsClass, tuxEventsUlogCat,  
tuxEventsUlogMsgNum, beaDomainId, beaDomainKey, beaLogicalAgentName }
```

### Description

This trap is generated when `.SysNetworkConfig` occurs. This message indicates the network link between the two machines specified changed to a new state.

### Action

This message is informational.

### Trap ID

12

## networkDroppedTrap

### Enterprise

tuxedo

### Variables

```
{tuxEventsName, tuxEventsSeverity, tuxEventsLmid, tuxEventsTime,  
tuxEventsUsec, tuxEventsDescription, tuxEventsClass, tuxEventsUlogCat,  
tuxEventsUlogMsgNum, beaDomainId, beaDomainKey, beaLogicalAgentName }
```

### Description

This trap is generated when `.SysNetworkDropped` occurs. This message indicates the network link between the two machines specified was dropped abnormally.

### Action

This can happen either because the BRIDGE on either machine died or one of the machines crashed.

## Trap ID

13

### **networkFailureTrap**

#### Enterprise

tuxedo

#### Variables

```
{ tuxEventsName, tuxEventsSeverity, tuxEventsLmid, tuxEventsTime,  
tuxEventsUsec, tuxEventsDescription, tuxEventsClass, tuxEventsUlogCat,  
tuxEventsUlogMsgNum, beaDomainId, beaDomainKey, beaLogicalAgentName }
```

#### Description

This trap is generated when `.SysNetworkFailure` occurs. This indicates a network connection failure between BRIDGE processes.

#### Action

This can happen either because the BRIDGE on remote machine died or the remote machine itself crashed.

## Trap ID

14

### **networkFlowTrap**

#### Enterprise

tuxedo

#### Variables

```
{ tuxEventsName, tuxEventsSeverity, tuxEventsLmid, tuxEventsTime,  
tuxEventsUsec, tuxEventsDescription, tuxEventsClass, tuxEventsUlogCat,  
tuxEventsUlogMsgNum, beaDomainId, beaDomainKey, beaLogicalAgentName }
```

#### Description

This trap is generated when `.SysNetworkFlow` occurs. This message states that the virtual circuit between machines changed to a new state.

## Action

This message is informational.

## Trap ID

15

## networkStateTrap

### Enterprise

tuxedo

### Variables

```
{ tuxEventsName, tuxEventsSeverity, tuxEventsLmid, tuxEventsTime,  
tuxEventsUsec, tuxEventsDescription, tuxEventsClass, tuxEventsUlogCat,  
tuxEventsUlogMsgNum, beaDomainId, beaDomainKey, beaLogicalAgentName}
```

### Description

This trap is generated when `.SysNetworkState` occurs. This message indicates that the server died abnormally and BBL cleaned up the slot allocated by the server.

## Action

Debug the server and fix the problem before the server is restarted.

## Trap ID

16

## SERVER Event Traps

The Server Traps group defines the Tuxedo server specific traps.

## serverCleaningTrap

### Enterprise

tuxedo

## Variables

```
{ tuxEventsName, tuxEventsSeverity, tuxEventsLmid, tuxEventsTime,
  tuxEventsUsec, tuxEventsDescription, tuxEventsClass, tuxEventsUlogCat,
  tuxEventsUlogMsgNum, beaDomainId, beaDomainKey, beaLogicalAgentName }
```

## Description

This trap is generated when `.SysServerCleaning` occurs. This message indicates that the server died abnormally and BBL cleaned up the slot allocated by the server.

## Action

Debug the server and fix the problem before the server is restarted.

## Trap ID

17

## serverConfigTrap

### Enterprise

tuxedo

## Variables

```
{ tuxEventsName, tuxEventsSeverity, tuxEventsLmid, tuxEventsTime,
  tuxEventsUsec, tuxEventsDescription, tuxEventsClass, tuxEventsUlogCat,
  tuxEventsUlogMsgNum, beaDomainId, beaDomainKey, beaLogicalAgentName }
```

## Description

This trap is generated when `.SysServerConfig` occurs. This message indicates that the configuration parameters for the server have been updated.

## Action

This message is informational.

## Trap ID

18

## serverDiedTrap

### Enterprise

tuxedo

### Variables

```
{ tuxEventsName, tuxEventsSeverity, tuxEventsLmid, tuxEventsTime,  
tuxEventsUsec, tuxEventsDescription, tuxEventsClass, tuxEventsUlogCat,  
tuxEventsUlogMsgNum, beaDomainId, beaDomainKey, beaLogicalAgentName }
```

### Description

This trap is generated when `.SysServerDied` occurs. This message indicates that the server died abnormally and the BBL detected this condition in its periodic scan of the BB.

### Action

Debug the server and fix the problem before the server is restarted.

### Trap ID

19

## serverInitTrap

### Enterprise

tuxedo

### Variables

```
{ tuxEventsName, tuxEventsSeverity, tuxEventsLmid, tuxEventsTime,  
tuxEventsUsec, tuxEventsDescription, tuxEventsClass, tuxEventsUlogCat,  
tuxEventsUlogMsgNum, beaDomainId, beaDomainKey, beaLogicalAgentName}
```

### Description

This trap is generated when `.SysServerInit` occurs. This message indicates that the server specified above failed in `tpsivrinit()` and therefore could not be booted.

### Action

Fix the problem and then reboot the server. This problem might be due to a Tuxedo resource limit or an application-specific problem.

## Trap ID

20

**serverMaxgenTrap**

## Enterprise

tuxedo

## Variables

```
{ tuxEventsName, tuxEventsSeverity, tuxEventsLmid, tuxEventsTime,  
tuxEventsUsec, tuxEventsDescription, tuxEventsClass, tuxEventsUlogCat,  
tuxEventsUlogMsgNum, beaDomainId, beaDomainKey, beaLogicalAgentName }
```

## Description

This trap is generated when `.SysServerMaxgen` occurs. This message indicates that the server died abnormally. Since the server has been marked as restartable, it has been restarted `MAXGEN-1` times in the specified GRACE period.

## Action

Tuxedo application servers should not die abnormally. If this happens, it is most likely due to an application-specific problem. Debug the server and resolve the problem before restarting the server.

## Trap ID

21

**serverRestartingTrap**

## Enterprise

tuxedo

## Variables

```
{ tuxEventsName, tuxEventsSeverity, tuxEventsLmid, tuxEventsTime,  
tuxEventsUsec, tuxEventsDescription, tuxEventsClass, tuxEventsUlogCat,  
tuxEventsUlogMsgNum, beaDomainId, beaDomainKey, beaLogicalAgentName }
```

## Description

This trap is generated when `.SysServerRestarting` occurs.. This message indicates that the server died abnormally. Since this has been marked as a restartable server, it has been restarted.

## Action

Tuxedo application servers should not die abnormally. If this happens, it is most likely due to an application-specific problem. Debug the server and resolve the problem before restarting the server.

## Trap ID

22

## **serverStateTrap**

### Enterprise

tuxedo

### Variables

```
{tuxEventsName, tuxEventsSeverity, tuxEventsLmid, tuxEventsTime,  
tuxEventsUsec, tuxEventsDescription, tuxEventsClass, tuxEventsUlogCat,  
tuxEventsUlogMsgNum, beaDomainId, beaDomainKey, beaLogicalAgentName }
```

## Description

This trap is generated when `.SysServerState` occurs. This message indicates that the server changed state.

## Action

This message is informational.

## Trap ID

23

## **serverTpExitTrap**

### Enterprise

tuxedo

## Variables

```
{ tuxEventsName, tuxEventsSeverity, tuxEventsLmid, tuxEventsTime,
tuxEventsUsec, tuxEventsDescription, tuxEventsClass, tuxEventsUlogCat,
tuxEventsUlogMsgNum, beaDomainId, beaDomainKey, beaLogicalAgentName }
```

## Description

This trap is generated when `.SysServerTpexit` occurs. This message indicates that the server received a request and the service routine code did a `tpretreturn(TPEXIT)` while the server was executing application-specific code.

## Action

This message is informational.

## Trap ID

24

# CLIENT Traps

The Client Traps group defines the Tuxedo client-specific traps.

## clientConfigTrap

### Enterprise

tuxedo

## Variables

```
{ tuxEventsName, tuxEventsSeverity, tuxEventsLmid, tuxEventsTime,
tuxEventsUsec, tuxEventsDescription, tuxEventsClass, tuxEventsUlogCat,
tuxEventsUlogMsgNum, beaDomainId, beaDomainKey, beaLogicalAgentName }
```

## Description

This trap is generated when `.SysClientConfig` is raised. This denotes that a particular user on a machine changed its configuration.

## Action

This message is informational.

## Trap ID

25

### **clientDiedTrap**

#### Enterprise

tuxedo

#### Variables

```
{ tuxEventsName, tuxEventsSeverity, tuxEventsLmid, tuxEventsTime,  
tuxEventsUsec, tuxEventsDescription, tuxEventsClass, tuxEventsUlogCat,  
tuxEventsUlogMsgNum, beaDomainId, beaDomainKey, beaLogicalAgentName }
```

#### Description

This trap is generated when `.SysClientDied` occurs. This message indicates that the client exited the application without doing a `tpterm()`. Normally, clients should do a `tpterm()` before exiting the application.

#### Action

This message is informational.

## Trap ID

26

### **clientSecurityTrap**

#### Enterprise

tuxedo

#### Variables

```
{ tuxEventsName, tuxEventsSeverity, tuxEventsLmid, tuxEventsTime,  
tuxEventsUsec, tuxEventsDescription, tuxEventsClass,  
tuxEventsUlogCat, tuxEventsUlogMsgNum, beaDomainId, beaDomainKey,  
beaLogicalAgentName }
```

## Description

This trap is generated when `.SysClientSecurity` occurs. This message indicates that the client failed security validation when trying to join the application.

## Action

Check to make sure that this is not an unauthorized user trying to gain access to your application data.

## Trap ID

27

## clientStateTrap

### Enterprise

tuxedo

### Variables

```
{ tuxEventsName, tuxEventsSeverity, tuxEventsLmid, tuxEventsTime,
  tuxEventsUsec, tuxEventsDescription, tuxEventsClass, tuxEventsUlogCat,
  tuxEventsUlogMsgNum, beaDomainId, beaDomainKey, beaLogicalAgentName}
```

## Description

This trap is generated when `.SysClientState` occurs. This message indicates that a particular client on a machine changed state.

## Action

This message is informational.

## Trap ID

28

## TRANSACTION Traps

The Transaction Traps group defines the Tuxedo transaction-specific traps.

## transHeuristicAbortTrap

### Enterprise

tuxedo

### Variables

```
{ tuxEventsName, tuxEventsSeverity, tuxEventsLmid, tuxEventsTime,  
tuxEventsUsec, tuxEventsDescription, tuxEventsClass, tuxEventsUlogCat,  
tuxEventsUlogMsgNum, beaDomainId, beaDomainKey, beaLogicalAgentName }
```

### Description

This trap is generated when `.SysTransactionHeuristicAbort` occurs. This message indicates that the database in a particular group performed an heuristic abort on a transaction.

### Action

Check to make sure that the coordinator of the transaction is still running.

### Trap ID

29

## transHeuristicCommitTrap

### Enterprise

tuxedo

### Variables

```
{ tuxEventsName, tuxEventsSeverity, tuxEventsLmid, tuxEventsTime,  
tuxEventsUsec, tuxEventsDescription, tuxEventsClass, tuxEventsUlogCat,  
tuxEventsUlogMsgNum, beaDomainId, beaDomainKey, beaLogicalAgentName }
```

### Description

This trap is generated when `.SysTransactionHeuristicCommit` occurs. This message indicates that the database in a particular group performed an heuristic commit on a transaction.

### Action

Check to make sure that the coordinator of the transaction is still running.

Trap ID

30

## EVENT Traps

The Event Traps group defines the Tuxedo event specific traps.

### eventDeliveryTrap

Enterprise

tuxedo

Variables

```
{ tuxEventsName, tuxEventsSeverity, tuxEventsLmid, tuxEventsTime,
  tuxEventsUsec, tuxEventsDescription, tuxEventsClass, tuxEventsUlogCat,
  tuxEventsUlogMsgNum, beaDomainId, beaDomainKey, beaLogicalAgentName }
```

Description

This trap is generated when `.SysEventDelivery` occurs. This message indicates that the event server failed to perform at least one notification for a posted event.

Action

Check to make sure that the notifications specified in the subscriptions that match the posted event are doable.

Trap ID

31

### eventFailureTrap

Enterprise

tuxedo

Variables

```
{ tuxEventsName, tuxEventsSeverity, tuxEventsLmid, tuxEventsTime,
  tuxEventsUsec, tuxEventsDescription, tuxEventsClass, tuxEventsUlogCat,
  tuxEventsUlogMsgNum, beaDomainId, beaDomainKey, beaLogicalAgentName }
```

## Description

This trap is generated when `.SysEventFailure` occurs. The system event server periodically sends a message to itself to detect blocking conditions on the message queues. This event is generated if the server cannot put a message on the queue in no-block mode. It can also be generated if the received message does not match what was sent out earlier. The second possible case is very unlikely. This denotes a system event monitor subsystem failure on a particular host.

## Action

Configure larger message queues or distribute the load in the application equally among all the machines.

## Trap ID

32