



THE ENTERPRISE MIDDLEWARE SOLUTION

# BEA Manager

## Agent Connection for M3 and TUXEDO Systems Reference Manual

Agent Connection for M3 and TUXEDO 2.0  
Document Edition 2.0  
October 1998

# Copyright

Copyright © 1998 BEA Systems, Inc. All Rights Reserved.

## Restricted Rights Legend

This software and documentation is subject to and made available only pursuant to the terms of the BEA Systems License Agreement and may be used or copied only in accordance with the terms of that agreement. It is against the law to copy the software except as specifically allowed in the agreement. This document may not, in whole or in part, be copied photocopied, reproduced, translated, or reduced to any electronic medium or machine readable form without prior consent, in writing, from BEA Systems, Inc.

Use, duplication or disclosure by the U.S. Government is subject to restrictions set forth in the BEA Systems License Agreement and in subparagraph (c)(1) of the Commercial Computer Software-Restricted Rights Clause at FAR 52.227-19; subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013, subparagraph (d) of the Commercial Computer Software--Licensing clause at NASA FAR supplement 16-52.227-86; or their equivalent.

Information in this document is subject to change without notice and does not represent a commitment on the part of BEA Systems. THE SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. FURTHER, BEA Systems DOES NOT WARRANT, GUARANTEE, OR MAKE ANY REPRESENTATIONS REGARDING THE USE, OR THE RESULTS OF THE USE, OF THE SOFTWARE OR WRITTEN MATERIAL IN TERMS OF CORRECTNESS, ACCURACY, RELIABILITY, OR OTHERWISE.

## Trademarks or Service Marks

BEA, ObjectBroker, TOP END, and TUXEDO are registered trademarks of BEA Systems, Inc. BEA Builder, BEA Connect, BEA Manager, BEA MessageQ, Jolt and M3 are trademarks of BEA Systems, Inc.

All other company names may be trademarks of the respective companies with which they are associated.

### BEA Manager Agent Connection for M3 and TUXEDO Systems Reference Manual

Document Edition	Date	SoftwareVersion
2.0	October 1998	Agent Connection for M3 and TUXEDO 2.0

---

# Contents

## Preface

Purpose of This Manual.....	xxvii
Who Should Read This Manual .....	xxvii
How This Manual Is Organized .....	xxviii
How to Use This Manual.....	xxix
Opening the Manual in a Web Browser .....	xxx
How to Print .....	xxxii
How to Print the Complete Book .....	xxxii
Documentation Conventions .....	xxxii
Related Documentation .....	xxxiv
Contact Information.....	xxxiv
Documentation Support.....	xxxv
Customer Support.....	xxxv

## 1. Overview

Benefits of Network Management Integration .....	1-1
SNMP Overview .....	1-3
The TUXEDO MIB for SNMP .....	1-5
MIB Object Identifiers .....	1-6
ASN.1 File.....	1-8
MIBs Supported by the SNMP Agent Connection .....	1-8
Structure of the MIB Definitions .....	1-9
MIB Object Definitions Format.....	1-9
MIB Event Trap Definitions Format.....	1-10

---

Differences Between the TUXEDO MIB and the SNMP TUXEDO MIB .....	1-11
Managing M3 and TUXEDO Applications Using the SNMP MIB .....	1-12
Querying Non-Existent MIB Objects.....	1-12
Managing TUXEDO Resources from a Management Platform.....	1-13
Getting Started.....	1-15

## 2. Setting Up the Agents

Setting Up the Agent Connection .....	2-1
Advanced Configuration.....	2-4
Starting and Stopping the SNMP Agents .....	2-8
Starting the SNMP Agents on UNIX Platforms .....	2-8
Startup Options.....	2-9
Description of tux_snmpd and m3_snmpd Commands .....	2-10
Starting the SNMP Agents on Windows NT Systems .....	2-11
Startup Options.....	2-13
Stopping the Agents .....	2-14
TUXEDO Master and Non-Master Nodes .....	2-14
Local and Global TUXEDO Information.....	2-15
Updating MIB Objects .....	2-16
Disabling SET Access .....	2-16

## 3. Integrating Agent Connection with a Management System

Integrating M3 and TUXEDO Event Notifications.....	3-3
Retrieving or Modifying Object Values when Managing Multiple Domains.....	3-6
Integrating Events Generated by Agent Integrator Polling .....	3-7
Integrating the Agent Connection with the Tivoli TME Platform .....	3-8
Distributed Monitoring of TUXEDO or M3 Applications.....	3-10

## 4. TUXEDO Core MIB

tuxTBridgeTbl .....	4-3
tuxTBridgeLmid .....	4-4
tuxTBridgeState .....	4-4
tuxTBridgeCurTime .....	4-6
tuxTBridgeConTime .....	4-6
tuxTBridgeSuspTime .....	4-7

---

tuxTBridgeRcvdByte .....	4-7
tuxTBridgeSentByte.....	4-7
tuxTBridgeRcvdNum.....	4-7
tuxTBridgeSentNum .....	4-8
tuxTBridgeFlowCnt .....	4-8
tuxTBridgeCurEncryptBits .....	4-8
tuxTBridgeNetworkGrpNo .....	4-8
tuxTBridgeNetworkGrpName .....	4-9
tuxTclientTbl .....	4-10
tuxTclientState .....	4-12
tuxTclientBirthTime.....	4-14
tuxTclientMachineId .....	4-14
tuxTclientReg .....	4-14
tuxTclientClntName.....	4-14
tuxTclientIdleTime.....	4-15
tuxTclientPid .....	4-15
tuxTclientSrvGrp .....	4-15
tuxTclientUsrName .....	4-15
tuxTclientWsc .....	4-16
tuxTclientWsh .....	4-16
tuxTclientWshClientId.....	4-16
tuxTclientRelease.....	4-16
tuxTclientWsProto .....	4-17
tuxTclientNumConv.....	4-17
tuxTclientNumDeque.....	4-17
tuxTclientNumEnque .....	4-17
tuxTclientNumPost .....	4-17
tuxTclientNumReq.....	4-18
tuxTclientNumSubscribe .....	4-18
tuxTclientNumTran.....	4-18
tuxTclientNumTranAbt.....	4-18
tuxTclientNumTranCmt.....	4-18
tuxTclientCmtRet.....	4-19
tuxTclientCurConv.....	4-19
tuxTclientCurReq.....	4-19

---

tuxTclientCurTime .....	4-19
tuxTclientLastGrp .....	4-20
tuxTclientNaddr .....	4-20
tuxTclientNotify .....	4-20
tuxTclientNumUnSol .....	4-20
tuxTclientRpid.....	4-21
tuxTclientTimeLeft .....	4-21
tuxTclientTimeStart .....	4-21
tuxTclientTranLev .....	4-21
tuxTclientId .....	4-22
tuxTconnTable .....	4-23
tuxTconnSerNo .....	4-24
tuxTconnState.....	4-24
tuxTconnSvcName .....	4-24
tuxTconnClientId .....	4-25
tuxTconnOgrpNo .....	4-25
tuxTconnOlmid .....	4-25
tuxTconnOpid.....	4-25
tuxTconnOsndcnt .....	4-25
tuxTconnOsrvId .....	4-26
tuxTconnSgrpNo .....	4-26
tuxTconnSlmid .....	4-26
tuxTconnSpid .....	4-26
tuxTconnSsndcnt .....	4-26
tuxTconnSsrvId .....	4-27
tuxTdeviceTbl.....	4-28
tuxTdevLmid .....	4-28
tuxTdevCfgDev .....	4-29
tuxTdeviceName .....	4-29
tuxTdevOffset.....	4-29
tuxTdevSize.....	4-29
tuxTdevIndex .....	4-30
tuxTdevState.....	4-30
tuxTwhichCfgDev .....	4-31
tuxTwhichCfgDev .....	4-31

---

tuxTdomain.....	4-32
tuxTdomainKey .....	4-34
tuxTdomainMaster .....	4-35
tuxTdomainModel.....	4-36
tuxTdomainState .....	4-36
tuxTdomainID .....	4-37
tuxTdomainUID .....	4-37
tuxTdomainGID .....	4-38
tuxTdomainPerm.....	4-38
tuxTdomainMask .....	4-38
tuxTdomainMaxAccessers .....	4-38
tuxTdomainMaxConv .....	4-39
tuxTdomainMaxGTT .....	4-39
tuxTdomainMaxBufsType .....	4-39
tuxTdomainMaxBufType.....	4-40
tuxTdomainMaxDRT .....	4-40
tuxTdomainMaxGroups .....	4-40
tuxTdomainMaxMachines .....	4-40
tuxTdomainMaxQueues .....	4-41
tuxTdomainMaxRFT .....	4-41
tuxTdomainMaxRTData .....	4-41
tuxTdomainMaxServers .....	4-42
tuxTdomainMaxServices .....	4-42
tuxTdomainMaxACLgroups .....	4-42
tuxTdomainCMTRET .....	4-43
tuxTdomainLoadBalance .....	4-43
tuxTdomainNotify.....	4-43
tuxTdomainSystemAccess .....	4-44
tuxTdomainOptions .....	4-45
tuxTdomainSignal.....	4-45
tuxTdomainSecurity.....	4-46
tuxTdomainAuthsvc.....	4-46
tuxTdomainScanUnit .....	4-47
tuxTdomainBBLQuery .....	4-47
tuxTdomainBlockTime .....	4-47

---

tuxTdomainDBBLWait .....	4-48
tuxTdomainSanityScan .....	4-48
tuxTdomainCurDRT .....	4-48
tuxTdomainCurGroups.....	4-48
tuxTdomainCurMachines.....	4-49
tuxTdomainCurQueues .....	4-49
tuxTdomainCurRFT .....	4-49
tuxTdomainCurRTdata.....	4-49
tuxTdomainCurServers .....	4-49
tuxTdomainCurServices.....	4-50
tuxTdomainCursType.....	4-50
tuxTdomainCurType .....	4-50
tuxTdomainHwDRT.....	4-50
tuxTdomainHwGroups .....	4-50
tuxTdomainHwMachines .....	4-51
tuxTdomainHwQueues.....	4-51
tuxTdomainHwRFT .....	4-51
tuxTdomainHwRTdata.....	4-51
tuxTdomainHwServers.....	4-51
tuxTdomainHwServices .....	4-52
tuxTdomainMaxNetGroups .....	4-52
m3MaxObjects .....	4-52
m3MaxInterfaces.....	4-52
m3CurInterfaces .....	4-53
m3HwInterfaces .....	4-53
tuxTgroupTable .....	4-54
tuxTgroupName .....	4-54
tuxTgroupNo .....	4-55
tuxTgroupLMID.....	4-55
tuxTgroupState .....	4-56
tuxTgroupCurLMID.....	4-58
tuxTgroupCloseInfo .....	4-58
tuxTgroupOpenInfo.....	4-58
tuxTgroupTMScount.....	4-59
tuxTgroupTMSname .....	4-59



---

tuxTmachineTable.....	4-60
tuxTmachinePmid .....	4-61
tuxTmachineLmid .....	4-62
tuxTmachineTuxConfig .....	4-62
tuxTmachineTuxDir .....	4-62
tuxTmachineAppDir .....	4-63
tuxTmachineState.....	4-63
tuxTmachineUid.....	4-65
tuxTmachineGid.....	4-65
tuxTmachineEnvFile .....	4-65
tuxTmachinePerm .....	4-66
tuxTmachineUlogPfx .....	4-66
tuxTmachineType .....	4-67
tuxTmachineMaxAccessers .....	4-67
tuxTmachineMaxConv.....	4-67
tuxTmachineMaxGtt .....	4-68
tuxTmachineMaxWsClients.....	4-68
tuxTmachineMaxAclCache .....	4-68
tuxTmachineTlogDevice.....	4-69
tuxTmachineTlogName .....	4-69
tuxTmachineTlogSize .....	4-69
tuxTmachineBridge.....	4-70
tuxTmachineNaddr.....	4-70
tuxTmachineNlsaddr .....	4-70
tuxTmachineCmpLimit .....	4-71
tuxTmachineTmNetLoad .....	4-71
tuxTmachineSpinCount .....	4-72
tuxTmachineRole .....	4-72
tuxTmachineMinor.....	4-72
tuxTmachineRelease .....	4-73
tuxTmachineMaxPendingBytes .....	4-73
m3MachineMaxObjects .....	4-73
tuxTmachineActive .....	4-74
tuxTmachineCurAccessers.....	4-75
tuxTmachineCurClients .....	4-76

---

tuxTmachineCurConv .....	4-76
tuxTmachineCurGTT .....	4-76
tuxTmachineCurLoad.....	4-76
tuxTmachineCurWsClients .....	4-76
tuxTmachineHwAccessers .....	4-77
tuxTmachineHwClients.....	4-77
tuxTmachineHwConv .....	4-77
tuxTmachineHwGTT .....	4-77
tuxTmachineHwWsClients .....	4-77
tuxTmachineNumConv .....	4-78
tuxTmachineNumDequeue.....	4-78
tuxTmachineNumEnqueue.....	4-78
tuxTmachineNumPost.....	4-78
tuxTmachineNumReq .....	4-78
tuxTmachineNumSubscribe .....	4-79
tuxTmachineNumTran .....	4-79
tuxTmachineNumTranAbt .....	4-79
tuxTmachineNumTranCmt .....	4-79
tuxTmachineLicExpires .....	4-79
tuxTmachineLicMaxUsers .....	4-80
tuxTmachineLicSerial .....	4-80
tuxTmachinePageSize .....	4-80
tuxTmachineSWrelease.....	4-80
tuxTmachineHwAclCache .....	4-80
tuxTmachineAclCacheHits .....	4-81
tuxTmachineAclCacheAccess.....	4-81
tuxTmachineAclFail.....	4-81
tuxTmachineWkCompleted .....	4-81
tuxTmachineWkInitiated.....	4-82
m3MachineCurObjects.....	4-82
m3MachineHwObjects.....	4-82
tuxTmsgTable .....	4-83
tuxTmsgId .....	4-83
tuxTmsgState.....	4-84
tuxTmsgCurTime .....	4-84

---

tuxTmsgCbytes .....	4-84
tuxTmsgCtime .....	4-85
tuxTmsgLrPid .....	4-85
tuxTmsgLsPid .....	4-85
tuxTmsgQbytes .....	4-85
tuxTmsgQnum .....	4-85
tuxTmsgRtime .....	4-86
tuxTmsgStime .....	4-86
tuxTqueueTable .....	4-87
tuxTqueueRqAddr.....	4-87
tuxTqueueState.....	4-88
tuxTqueueRqId.....	4-89
tuxTqueueSrvrCnt.....	4-89
tuxTqueueTotNqueued.....	4-89
tuxTqueueTotWkQueued .....	4-90
tuxTqueueSource .....	4-90
tuxTqueueNqueued .....	4-90
tuxTqueueWkQueued .....	4-91
tuxTroutingTable .....	4-92
tuxTroutingName .....	4-93
tuxTroutingBufType .....	4-93
tuxTroutingField .....	4-93
tuxTroutingRanges .....	4-94
tuxTroutingState.....	4-95
beaRoutingType .....	4-95
m3RoutingFieldType .....	4-96
tuxInternalIdx.....	4-96
tuxTsrvrTbl.....	4-97
tuxTsrvrGrp.....	4-98
tuxTsrvrId.....	4-98
tuxTsrvrName .....	4-99
tuxTsrvrGrpNo.....	4-99
tuxTsrvrState .....	4-99
tuxTsrvrBaseSrvId .....	4-101
tuxTsrvrClOpt .....	4-101

---

tuxTsrvrEnvFile .....	4-102
tuxTsrvrGrace.....	4-102
tuxTsrvrMaxgen .....	4-103
tuxTsrvrMax .....	4-103
tuxTsrvrMin .....	4-104
tuxTsrvrRcmd.....	4-104
tuxTsrvrRestart.....	4-104
tuxTsrvrSequence.....	4-105
tuxTsrvrSystemAccess .....	4-105
tuxTsrvrConv .....	4-106
tuxTsrvrReplyQ.....	4-106
tuxTsrvrRpPerm .....	4-106
tuxTsrvrRqAddr .....	4-106
tuxTsrvrRqPerm .....	4-107
tuxTsrvrGeneration .....	4-107
tuxTsrvrPid.....	4-107
tuxTsrvrRpid .....	4-108
tuxTsrvrRqId .....	4-108
tuxTsrvrTimeRestart .....	4-108
tuxTsrvrTimeStart .....	4-109
tuxTsrvrTblExt .....	4-110
tuxTsrvrIdExt .....	4-111
tuxTsrvrGrpNoExt .....	4-111
tuxTsrvrNumConv .....	4-112
tuxTsrvrNumDeque.....	4-112
tuxTsrvrNumEnque .....	4-112
tuxTsrvrNumPost .....	4-112
tuxTsrvrNumReq.....	4-112
tuxTsrvrNumSubscribe .....	4-113
tuxTsrvrNumTran.....	4-113
tuxTsrvrTranAbt.....	4-113
tuxTsrvrTranCmt.....	4-113
tuxTsrvrTotReqC .....	4-113
tuxTsrvrTotWorkL .....	4-114
tuxTsrvrCltLmid.....	4-114

---

tuxTsrvrCltpid .....	4-114
tuxTsrvrCltply .....	4-114
tuxTsrvrCmtRet .....	4-115
tuxTsrvrCurConv .....	4-115
tuxTsrvrCurReq .....	4-115
tuxTsrvrCurService .....	4-115
tuxTsrvrCurTime .....	4-116
tuxTsrvrLastGrp .....	4-116
tuxTsrvrSvcTimeOut .....	4-116
tuxTsrvrTimeLeft .....	4-116
tuxTsrvrTranLev .....	4-117
tuxTsrvrStateExt .....	4-117
tuxTsrvrGrpExt .....	4-117
m3SrvrCurObjsExt .....	4-117
m3SrvrCurInterfaceExt .....	4-117
tuxTsvcTbl .....	4-118
tuxTsvcName .....	4-118
tuxTsvcType .....	4-119
tuxTsvcState .....	4-119
tuxTsvcAutoTran .....	4-120
tuxTsvcLoad .....	4-120
tuxTsvcPrio .....	4-120
tuxTsvcTimeOut .....	4-121
tuxTsvcTranTime .....	4-121
tuxTsvcBufType .....	4-122
tuxTsvcRoutingName .....	4-122
tuxTsvcGrp .....	4-123
tuxTsvcGrpSvcName .....	4-124
tuxTsvcGrpName .....	4-125
tuxTsvcGrpNo .....	4-125
tuxTsvcGrpState .....	4-125
tuxTsvcGrpAutoTran .....	4-126
tuxTsvcGrpLoad .....	4-126
tuxTsvcGrpPrio .....	4-126
tuxTsvcGrpSvcTimeOut .....	4-126

---

tuxTsvcGrpTranTime .....	4-127
tuxTsvcSrvrLmid .....	4-127
tuxTsvcSrvrRqAddr .....	4-127
tuxTsvcSrvrId .....	4-127
tuxTsvcrName .....	4-128
tuxTsvcSrvrNcompleted .....	4-128
tuxTsvcSrvrNqueued .....	4-128
tuxTlistenTbl .....	4-129
tuxTlistenLmid .....	4-129
tuxTlistenState .....	4-129
tuxTranTbl .....	4-130
tuxTranCoordLmid .....	4-131
tuxTpTranId .....	4-131
tuxTranXid .....	4-131
tuxTranIndx1 .....	4-131
tuxTranIndx2 .....	4-132
tuxTranIndx3 .....	4-132
tuxTranIndx4 .....	4-132
tuxTranIndx5 .....	4-132
tuxTranState .....	4-132
tuxTranTimeOut .....	4-134
tuxTranGrpCnt .....	4-134
tuxTranGrpIndex .....	4-134
tuxTranGrpNo .....	4-135
tuxTranGstate .....	4-135
tuxTulogTable .....	4-138
tuxTulogSerNo .....	4-139
tuxTulogLmid .....	4-139
tuxTulogPmid .....	4-139
tuxTulogMmDdYy .....	4-139
tuxTulogTime .....	4-139
tuxTulogLine .....	4-140
tuxTulogMsg .....	4-140
tuxTulogTpTranId .....	4-140
tuxTulogXid .....	4-140

---

tuxTulogPid.....	4-140
tuxTulogSeverity.....	4-141
tuxTulogCat .....	4-141
tuxTulogMsgNum .....	4-141
tuxTulogProcName .....	4-141
tuxTulogCtrl .....	4-142
tuxTulogLmidCtrl .....	4-143
tuxTulogPmidCtrl .....	4-143
tuxTulogMmddyCtrl .....	4-143
tuxTulogTimeCtrl .....	4-143
tuxTulogEndTimeCtrl .....	4-144
tuxTulogLineCtrl .....	4-144
tuxTulogMsgCtrl.....	4-144
tuxTulogTptranIdCtrl.....	4-144
tuxTulogXidCtrl.....	4-145
tuxTulogPidCtrl .....	4-145
tuxTulogSeverityCtrl .....	4-145
tuxTulogCatCtrl .....	4-145
tuxTulogMsgNumCtrl.....	4-146
tuxTulogProcNameCtrl .....	4-146
tuxTnetGrpTbl .....	4-147
tuxTnetGrpName .....	4-147
tuxTnetGrpNo .....	4-147
tuxTnetGrpState .....	4-148
tuxTnetGrpPrio .....	4-148
tuxTnetMapTbl.....	4-149
tuxTnetMapGrpName .....	4-149
tuxTnetMapGrpNo.....	4-150
tuxTnetMapLmid .....	4-150
tuxTnetMapState.....	4-150
tuxTnetMapNaddr.....	4-151
tuxTnetMapMinEncryptBit.....	4-151
tuxTnetMapMaxEncryptBit .....	4-151
beaEventFilters .....	4-152
beaEvtFilterTblStatus.....	4-152

---

beaEvtFilterTable .....	4-153
beaEvtFilterId .....	4-153
beaEvtAgentName .....	4-153
beaEvtExpr .....	4-154
beaEvtFilter .....	4-154
beaEvtFilterState .....	4-155

## 5. BEA Domain List

beaDomainKey .....	5-1
beaLogicalAgentName .....	5-2
beaDomainId .....	5-2
beaDomainTuxdir.....	5-2
beaDomainTuxconfig.....	5-3
beaDomainStatus.....	5-3

## 6. M3 MIB Groups

m3FactoryTable .....	6-3
m3FactorySerNo .....	6-3
m3FactoryId .....	6-3
m3FactoryIfName .....	6-4
m3FactoryState.....	6-4
m3InterfaceTable.....	6-5
m3IfSerNo .....	6-6
m3IfName.....	6-7
m3IfSrvGrp .....	6-7
m3IfState .....	6-7
m3IfAutoTran.....	6-9
m3IfLoad .....	6-10
m3IfPrio .....	6-10
m3IfTimeout.....	6-10
m3IfTranTime .....	6-11
m3IfFbRoutingName .....	6-11
m3IfLmid .....	6-11
m3IfNumServers .....	6-11
m3IfTpPolicy .....	6-12



---

m3IfTxPolicy .....	6-12
m3LclInterfaceTable .....	6-13
m3LclIfSerNo .....	6-13
m3LclIfName .....	6-13
m3LclIfSrvGrp.....	6-14
m3LclIfNcompleted .....	6-14
m3LclIfNqueued .....	6-14
m3IfQueueTable.....	6-15
m3IfQueueSerNo .....	6-16
m3IfQueueName .....	6-16
m3IfQueueSrvGrp.....	6-16
m3IfQueueRqAddr.....	6-16
m3IfQueueState .....	6-17
m3IfQueueAutoTran .....	6-18
m3IfQueueLoad .....	6-18
m3IfQueuePrio.....	6-18
m3IfQueueTimeout .....	6-19
m3IfQueueTranTime .....	6-19
m3IfQueueFbRoutingName.....	6-19
m3IfQueueLmid.....	6-19
m3IfQueueNumServers .....	6-20
m3IfQueueTpPolicy .....	6-20
m3LclIfQueueTable .....	6-21
m3LclIfQueueSerNo .....	6-21
m3LclIfQueueName.....	6-21
m3LclIfQueueSrvGrp .....	6-22
m3LclIfQueueRqAddr .....	6-22
m3LclIfQueueNcompleted.....	6-22
m3LclIfQueueNqueued.....	6-22
m3LclIfQueueCurObjs.....	6-23
m3LclIfQueueCurTrans .....	6-23

---

## 7. Access Control List MIB

tuxTAclGrpTable.....	7-2
tuxTAclGrpName.....	7-2
tuxTAclGrpId .....	7-2
tuxTAclGrpState .....	7-3
tuxTAclPermTable .....	7-4
tuxTAclPermName.....	7-4
tuxTAclPermType .....	7-4
tuxTAclPermGrpIds .....	7-5
tuxTAclPermState .....	7-5
tuxTAclPrinTbl.....	7-6
tuxTAclPrinName .....	7-6
tuxTAclCltnName .....	7-7
tuxTAclPrinId.....	7-7
tuxTAclPrinGrp.....	7-7
tuxTAclPrinPasswd .....	7-7
tuxTAclPrinState .....	7-8

## 8. Workstation MIB

tuxTwshTbl.....	8-2
tuxTwshTaClientId.....	8-3
tuxTwshTaWshClientId .....	8-3
tuxTwshTaSrvGrp .....	8-3
tuxTwshTaSrvId.....	8-4
tuxTwshTaGrpNo.....	8-4
tuxTwshTaState.....	8-4
tuxTwshTaLmid .....	8-4
tuxTwshTaPid .....	8-5
tuxTwshTaNaddr.....	8-5
tuxTwshTaHwClients.....	8-5
tuxTwshTaMultiplex .....	8-5
tuxTwshTaCurClients .....	8-5
tuxTwshTaTimeleft .....	8-6
tuxTwshTaActive .....	8-6
tuxTwshTaTotacttime .....	8-6
tuxTwshTaTotidlname .....	8-6

---

tuxTwshTaCurwork .....	8-7
tuxTwshTaFlowcnt .....	8-7
tuxTwshTaNumblockQ .....	8-7
tuxTwshTaRcvdByt .....	8-7
tuxTwshTaRcvdNum .....	8-8
tuxTwshTaSentByt .....	8-8
tuxTwshTaSentNum .....	8-8
tuxTwsITbl .....	8-9
tuxTwsITaSrvGrp .....	8-10
tuxTwsITaSrvId .....	8-10
tuxTwsITaGrpNo .....	8-11
tuxTwsITaState .....	8-11
tuxTwsITaLmid .....	8-11
tuxTwsITaPid .....	8-11
tuxTwsITaDevice .....	8-11
tuxTwsITaNaddr .....	8-12
tuxTwsITaWshName .....	8-12
tuxTwsITaMinHandlers .....	8-12
tuxTwsITaMaxHandlers .....	8-13
tuxTwsITaMultiplex .....	8-13
tuxTwsITaMaxIdleTime .....	8-13
tuxTwsITaMaxInitTime .....	8-13
tuxTwsITaClOpt .....	8-14
tuxTwsITaEnvFile .....	8-14
tuxTwsITaGrace .....	8-14
tuxTwsITaMaxGen .....	8-15
tuxTwsITaRcmd .....	8-15
tuxTwsITaRestart .....	8-15
tuxTwsITaSequence .....	8-16
tuxTwsITaCurHandlers .....	8-16
tuxTwsITaHwHandlers .....	8-16
tuxTwsITaWsProto .....	8-17
tuxTwsITaSuspended .....	8-17
tuxTwsITaViewRefresh .....	8-17
tuxTwsITaKeepAlive .....	8-18
tuxTwsITaNetTimeOut .....	8-18

---

## 9. Application Queue MIB

tuxTAppQctrl.....	9-2
tuxTAppQctrlLmid.....	9-2
tuxTAppQctrlQmConfig .....	9-3
tuxTAppQctrlSpaceName .....	9-3
tuxTAppQctrlQname.....	9-3
tuxTAppQctrlMsgLoPrio .....	9-4
tuxTAppQctrlMsgHiPrio .....	9-4
tuxTAppQctrlMsgEndTime .....	9-4
tuxTAppQctrlMsgStartTime .....	9-5
tuxTAppQTbl .....	9-6
tuxTAppQname .....	9-7
tuxTAppQspaceName .....	9-7
tuxTAppQmConfig .....	9-7
tuxTAppQlmid .....	9-8
tuxTAppQgrpNo .....	9-8
tuxTAppQstate .....	9-8
tuxTAppQorder .....	9-9
tuxTAppQcmd.....	9-9
tuxTAppQcmdHw .....	9-10
tuxTAppQcmdLw .....	9-10
tuxTAppQmaxRetries .....	9-11
tuxTAppQoutOfOrder.....	9-11
tuxTAppQretryDelay .....	9-11
tuxTAppQcurBlocks .....	9-11
tuxTAppQcurMsg .....	9-11
tuxTAppQmsgTbl.....	9-12
tuxTAppQmsgId.....	9-13
tuxTAppQmsgSerNo.....	9-13
tuxTAppQmsgGrpNo .....	9-13
tuxTAppQmsgQname .....	9-13
tuxTAppQmsgQmConfig .....	9-14
tuxTAppQmsgQspaceName.....	9-14
tuxTAppQmsgLmid .....	9-14
tuxTAppQmsgState.....	9-14

---

tuxTAppQmsgNewQname.....	9-15
tuxTAppQmsgPrior.....	9-15
tuxTAppQmsgTime .....	9-15
tuxTAppQmsgCorId .....	9-16
tuxTAppQmsgCurRetries .....	9-16
tuxTAppQmsgSize .....	9-16
tuxTQspaceTbl .....	9-17
tuxTQspaceName .....	9-18
tuxTQspaceQmConfig .....	9-18
tuxTQspaceLmid.....	9-19
tuxTQspaceGrpNo .....	9-19
tuxTQspaceState .....	9-19
tuxTQspaceBlocking.....	9-20
tuxTQspaceErrQname .....	9-21
tuxTQspaceForceInit.....	9-21
tuxTQspaceIpckey .....	9-21
tuxTQspaceMaxMsg .....	9-21
tuxTQspaceMaxPages.....	9-22
tuxTQspaceMaxProc .....	9-22
tuxTQspaceMaxQueues .....	9-22
tuxTQspaceMaxTrans .....	9-22
tuxTQspaceCurExtent .....	9-23
tuxTQspaceCurMsg .....	9-23
tuxTQspaceCurProc .....	9-23
tuxTQspaceCurQueues .....	9-23
tuxTQspaceCurTrans .....	9-24
tuxTQspaceHwMsg .....	9-24
tuxTQspaceHwProc .....	9-24
tuxTQspaceHwQueues.....	9-24
tuxTQspaceHwTrans .....	9-25
tuxTQspacePercentInit.....	9-25
tuxTQtransTbl .....	9-26
tuxTQtransXid .....	9-26
tuxTQtransIndx1 .....	9-27
tuxTQtransIndx2 .....	9-27

---

tuxTQtransIndx3 .....	9-27
tuxTQtransIndx4 .....	9-27
tuxTQtransIndx5 .....	9-28
tuxTQtransGrpNo.....	9-28
tuxTQtranSpaceName .....	9-28
tuxTQtransQmConfig.....	9-28
tuxTQtransLmid .....	9-29
tuxTQtransState.....	9-29

## 10. Event Broker MIB

tuxEventClientTbl .....	10-3
tuxEventClientIndx .....	10-3
tuxEventClientExpr .....	10-3
tuxEventClientFilter .....	10-4
tuxEventClientState.....	10-4
tuxEventClientId .....	10-4
tuxEventCmdTbl .....	10-5
tuxEventCmdIndx .....	10-5
tuxEventCmdExpr .....	10-6
tuxEventCmdFilter .....	10-6
tuxEventCmdState.....	10-6
tuxEventCmd.....	10-7
tuxEventQueTbl.....	10-8
tuxEventQueIndx .....	10-9
tuxEventQueExpr .....	10-9
tuxEventQueFilter .....	10-9
tuxEventQueState.....	10-10
tuxEventQspace.....	10-10
tuxEventQname .....	10-10
tuxEventQctlQtop.....	10-11
tuxEventQctlBeforeMsgid .....	10-11
tuxEventQctlQtimeAbs .....	10-11
tuxEventQctlQtimeRel .....	10-12
tuxEventQctlDeqTime .....	10-12
tuxEventQctlPrior.....	10-12

---

tuxEventQctlMsgId .....	10-12
tuxEventQctlCorrId .....	10-13
tuxEventQctlReplyQ .....	10-13
tuxEventQctlFailQ .....	10-13
tuxEventPersist .....	10-13
tuxEventTran .....	10-14
tuxEventSvcTbl .....	10-15
tuxEventSvcIndx .....	10-15
tuxEventSvcExpr .....	10-16
tuxEventSvcFilter .....	10-16
tuxEventSvcState .....	10-16
tuxEventSvcName .....	10-17
tuxEventSvcPersist .....	10-17
tuxEventSvcTran .....	10-17
tuxEventUlogTbl .....	10-18
tuxEventUlogIndx .....	10-18
tuxEventUlogExpr .....	10-19
tuxEventUlogFilter .....	10-19
tuxEventUlogState .....	10-19
tuxEventUserlog .....	10-20

## 11. TUXEDO Traps MIB

Specific Trap Number .....	11-1
Variable Bindings .....	11-2
tuxEventsName .....	11-2
tuxEventsSeverity .....	11-2
tuxEventsLmid .....	11-3
tuxEventsTime .....	11-3
tuxEventsUsec .....	11-3
tuxEventsDescription .....	11-3
tuxEventsClass .....	11-4
tuxEventsUlogCat .....	11-4
tuxEventsUlogMsgNum .....	11-4
tuxTdomainID .....	11-4
tuxTdomainKey .....	11-5
beaLogicalAgentName .....	11-5

---

Trap Definitions.....	11-6
DOMAIN Traps.....	11-6
resourceConfigTrap.....	11-6
MACHINE Traps .....	11-7
machineBroadcastTrap.....	11-7
machineConfigTrap.....	11-7
machineFullMaxAccessersTrap .....	11-8
machineFullMaxConvTrap .....	11-8
machineFullMaxGttTrap .....	11-9
machineFullMaxWsClientsTrap .....	11-9
machineMsgQTrap.....	11-10
machinePartitionedTrap .....	11-10
machineSlowTrap.....	11-11
machineStateTrap.....	11-11
BRIDGE Traps .....	11-12
networkConfigTrap .....	11-12
networkDroppedTrap .....	11-12
networkFailureTrap .....	11-13
networkFlowTrap .....	11-13
networkStateTrap .....	11-14
SERVER Event Traps .....	11-14
serverCleaningTrap .....	11-14
serverConfigTrap .....	11-15
serverDiedTrap.....	11-15
serverInitTrap .....	11-16
serverMaxgenTrap .....	11-16
serverRestartingTrap .....	11-17
serverStateTrap.....	11-17
serverTpExitTrap .....	11-18
CLIENT Traps.....	11-18
clientConfigTrap .....	11-18
clientDiedTrap.....	11-19
clientSecurityTrap .....	11-19
clientStateTrap .....	11-20



---

TRANSACTION Traps.....	11-20
transHeuristicAbortTrap .....	11-20
transHeuristicCommitTrap.....	11-21
EVENT Traps.....	11-21
eventDeliveryTrap .....	11-21
eventFailureTrap .....	11-22

## Index



---

# Preface

This preface includes the following topics:

- ◆ Purpose of This Manual
- ◆ How to Use This Manual
- ◆ Related Documentation
- ◆ Contact Information

## Purpose of This Manual

This document describes the BEA Manager Agent Connection for M3 and TUXEDO and gives instructions for using the Agent Connection with BEA M3 or TUXEDO applications.

## Who Should Read This Manual

This document is intended for system administrators, network administrators, and developers interested in managing M3 or TUXEDO applications using enterprise system management tools. This document is intended for reference purposes only.

---

# How This Manual Is Organized

This manual is organized as follows:

- ◆ Chapter 1, “Overview” — Describes what the Agent Connection software is and how it can facilitate M3 or TUXEDO resource management. Explains the benefits of integrating M3 or TUXEDO applications into a network management platform. Provides an overview of the Simple Network Management Protocol (SNMP), and the TUXEDO Management Information Base (TMIB) for SNMP. Compares the standard TUXEDO MIB to the TUXEDO MIB for SNMP. Suggests next steps to take to get started using the Agent Connection.
- ◆ Chapter 2, “Setting Up the Agents” — Provides instructions for setting up and configuring the agents on the managed nodes. Explains how to start and stop the SNMP agents on UNIX and Windows platforms. Describes how the TUXEDO SNMP agent functions on TUXEDO master and non-master nodes.
- ◆ Chapter 3, “Integrating Agent Connection with a Management System” — Provides instructions for setting up and configuring the Agent Connection software on the management system, including how to integrate M3 and TUXEDO event notifications into your management platform or Tivoli TME platform.
- ◆ Chapter 4, “TUXEDO Core MIB” — Details the basic groups, objects, and attributes of those objects that form a BEA TUXEDO application, and are defined in the Core Management Information Base (MIB). The Core MIB is the main information repository to control the operation and configuration of a BEA TUXEDO application via these basic objects.
- ◆ Chapter 5, “BEA Domain List” — Describes the BEA domain list. The domain list is a MIB group that represents information about the M3 or TUXEDO domain the agent is monitoring, as specified at startup.
- ◆ Chapter 6, “M3 MIB Groups” — Describes the MIB groups and member objects that are specific to M3 applications. Also, lists the M3 specific objects included as a part of the TUXEDO Core MIB.
- ◆ Chapter 7, “Access Control List MIB” — Describes the access control list (ACL) MIB groups. An ACL is a list that specifies who and what is authorized to access TUXEDO system objects.

- 
- ◆ Chapter 8, “Workstation MIB” — Describes the workstation MIB groups. The Workstation MIB specifies the information required to control access to a TUXEDO application from multiple workstations. The Workstation MIB specifies information about workstation listeners (WSL) and workstation handlers (WSH).
  - ◆ Chapter 9, “Application Queue MIB” — Describes the Application Queue MIB groups. The Application Queue MIB provides the administrative environment required to manage and control access to application queues, and defines the structure of the application queues.
  - ◆ Chapter 10, “Event Broker MIB” — Describes the Event Broker MIB. The Event Broker MIB defines the characteristics by which an application (clients or services) can subscribe to be notified of TUXEDO run-time system events. (To enable both system event and application event notification, you must first define the system event broker and the application event broker in the TUXEDO Core MIB.)
  - ◆ Chapter 11, “TUXEDO Traps MIB” — Describes the TUXEDO Event Traps MIB. The TUXEDO SNMP agent on the master node subscribes to all system events and generates a corresponding SNMP trap notification whenever certain types of events occur—mostly failures that a system operator should know about. The Event Traps MIB defines all the traps which will be generated and the objects which are passed in the variable bindings for these traps. This chapter describes the cause and recommended action for each of these events.

## How to Use This Manual

This manual is designed primarily as an online, hypertext guide. If you are reading this as a paper publication, note that to get full use from this guide you should install and access it as an online document via a Web browser that supports HTML 3.0. (Information on how to install the online documentation is available in the *BEA Manager Installation Guide*.)

---

## Opening the Manual in a Web Browser

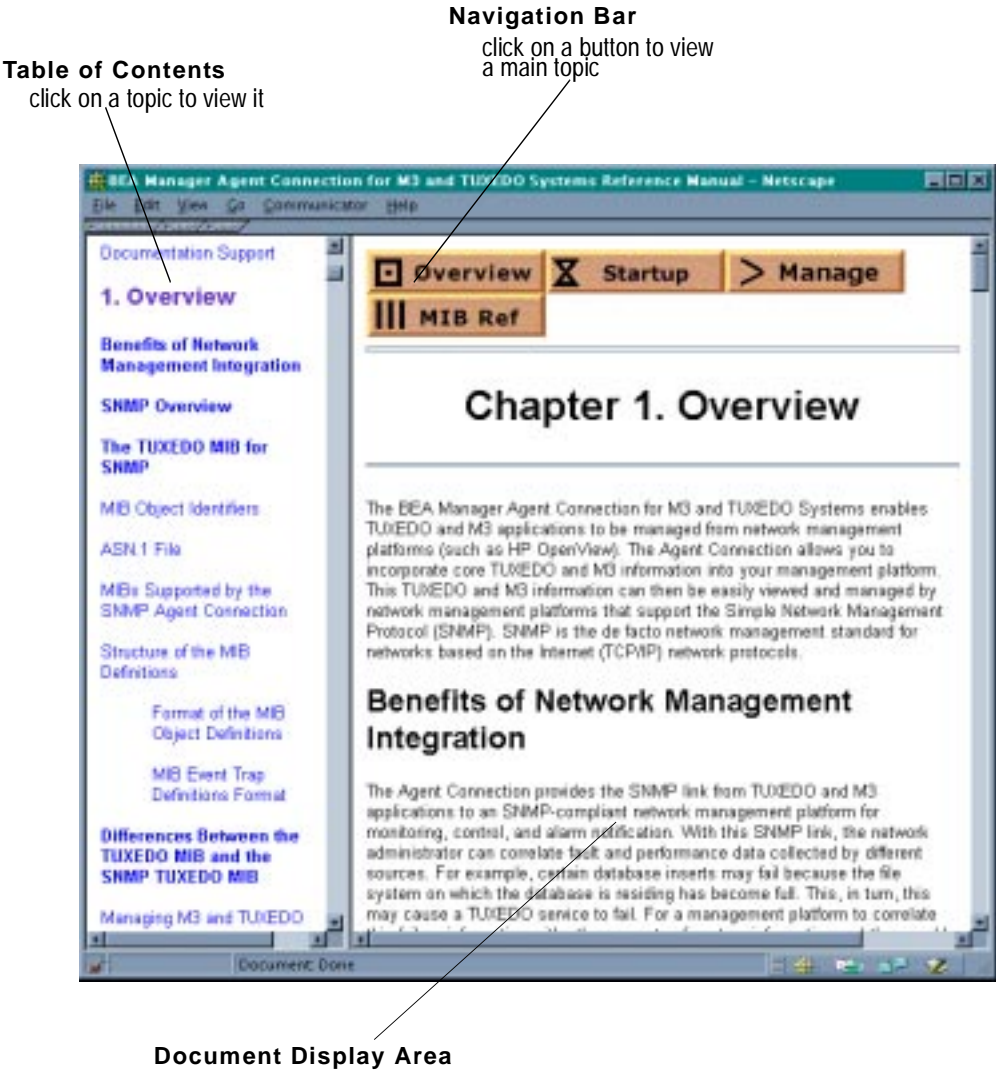
To access the online version of this document, open the following HTML file in a Web browser:

`$installation_dir/docs/mgr/20/agcon/index.htm`

Figure 1 shows the online guide with the clickable navigation bar and table of contents.

**Note:** The online documentation requires a Web browser that supports HTML 3.0. Netscape Navigator 2.02 or Microsoft Internet Explorer 3.0 or later are recommended.

**Figure 1 The Manual Displayed in a Web Browser**



---

# How to Print

You can print a hardcopy version of the BEA Manager Agent Connection for M3 and TUXEDO Systems Reference Manual, a chapter at a time, from the Web browser. Before you print, make sure that the chapter or appendix you want to print is displayed and *selected* in your browser. (To select a file, click anywhere inside the frame you want to print. If your browser offers a Print Preview feature, you can use it to verify what file you are about to print.)

## How to Print the Complete Book

A PDF version of the *Agent Connection for M3 and TUXEDO Systems Reference Manual* is available in the following location:

*YourDrive*: \docs\mgr\20\pdf\agcon.pdf

To print the documentation, open a PDF file in an Adobe Acrobat Reader and choose the file print option.

If you do not have a reader, you can download one from the Adobe Web site at <http://www.adobe.com/>.

## Documentation Conventions

Throughout this manual, the following formatting conventions are used to specify data entry words, screen and menu options, and various symbols.

Convention	Item
<b>boldface text</b>	Indicates terms defined in the glossary.
Ctrl+Tab	Indicates that you must press two or more keys sequentially.
<i>italics</i>	Indicates emphasis or book titles.



Convention	Item
monospace text	<p>Indicates code samples, commands and their options, data structures and their members, data types, directories, and file names and their extensions. Monospace text also indicates text that you must enter from the keyboard.</p> <p><i>Examples:</i></p> <pre>#include &lt;iostream.h&gt; void main ( ) the pointer psz chmod u+w * \tux\data\ap .doc tux.doc BITMAP float</pre>
<b>monospace boldface text</b>	<p>Identifies significant words in code.</p> <p><i>Example:</i></p> <pre>void <b>commit</b> ( )</pre>
<i>monospace italic text</i>	<p>Identifies variables in code.</p> <p><i>Example:</i></p> <pre>String <i>expr</i></pre>
UPPERCASE TEXT	<p>Indicates device names, environment variables, and logical operators.</p> <p><i>Examples:</i></p> <pre>LPT1 SIGNON OR</pre>
{ }	<p>Indicates a set of choices in a syntax line. The braces themselves should never be typed.</p>
[ ]	<p>Indicates optional items in a syntax line. The brackets themselves should never be typed.</p> <p><i>Example:</i></p> <pre>buildobjclient [-v] [-o name ] [-f <i>file-list</i>]... [-l <i>file-list</i>]...</pre>
	<p>Separates mutually exclusive choices in a syntax line. The symbol itself should never be typed.</p>

---

Convention	Item
...	<p>Indicates one of the following in a command line:</p> <ul style="list-style-type: none"> <li>◆ That an argument can be repeated several times in a command line</li> <li>◆ That the statement omits additional optional arguments</li> <li>◆ That you can enter additional parameters, values, or other information</li> </ul> <p>The ellipsis itself should never be typed.</p> <p><i>Example:</i></p> <pre>buildobjclient [-v] [-o name ] [-f file-list]... [-l file-list]...</pre>
. . . .	<p>Indicates the omission of items from a code example or from a syntax line.</p> <p>The vertical ellipsis itself should never be typed.</p>

---

## Related Documentation

In addition to this manual (*BEA Manager Agent Connection for M3 and TUXEDO Systems Reference Manual*), the Agent Connection documentation also includes:

- ◆ *BEA Manager Installation Guide*
- ◆ *BEA Manager Release Notes*

## Contact Information

The following sections provide information about how to obtain support for the documentation and software.

---

# Documentation Support

If you have questions or comments on the documentation, you can contact the BEA Information Engineering Group by e-mail at **docsupport@beasys.com**. (For information about how to contact Customer Support, refer to the following section.)

## Customer Support

If you have any questions about this version of the BEA Manager Agent Connection software, or if you have problems installing and running the software, contact BEA Customer Support through BEA WebSupport at [www.beasys.com](http://www.beasys.com). You can also contact Customer Support by using the contact information provided on the Customer Support Card, which is included in the product package.

When contacting Customer Support, be prepared to provide the following information:

- ◆ Your name, e-mail address, phone number, and fax number
- ◆ Your company name and company address
- ◆ Your machine type and authorization codes
- ◆ The name and version of the product you are using
- ◆ A description of the problem and the content of pertinent error messages



# 1 Overview

The BEA Manager Agent Connection for M3 and TUXEDO Systems enables TUXEDO and M3 applications to be managed from network management platforms (such as HP OpenView). The Agent Connection allows you to incorporate core TUXEDO and M3 information into your management platform. This TUXEDO and M3 information can then be easily viewed and managed by network management platforms that support the Simple Network Management Protocol (SNMP). SNMP is the de facto network management standard for networks based on the Internet (TCP/IP) network protocols.

## Benefits of Network Management Integration

The Agent Connection provides the SNMP link from TUXEDO and M3 applications to an SNMP-compliant network management platform for monitoring, control, and alarm notification. With this SNMP link, the network administrator can correlate fault and performance data collected by different sources. For example, certain database inserts may fail because the file system on which the database is residing has become full. This, in turn, may cause a TUXEDO service to fail. For a management platform to correlate this failure information with other aspects of system information and thus enable a pro-active management of the system, all pieces of management information need to be available from the same management console.

To achieve this level of correlation, you need a standardized way of communicating management information. SNMP is an open network management standard that provides a unified way of representing information about the manageable features of the heterogeneous components of large distributed systems.

Why provide an SNMP link to SNMP-compliant management platforms? The main reason is that TUXEDO or M3 applications are part of an overall organization or business middleware solution. It is not the only application that will be running on your network. Integrating TUXEDO or M3 applications with SNMP allows you to effectively manage all of your large-scale applications using the SNMP-compliant network management tool of your choice. Since most of the management platforms support SNMP today, SNMP agents for TUXEDO and M3 applications can be integrated into virtually every management framework. Examples of such management platforms include:

- ◆ HP OpenView
- ◆ IBM NetView/6000
- ◆ Tivoli
- ◆ Sun Domain/SunNet/Site Manager
- ◆ CA Unicenter

These management platforms help the network or system administrator to manage and control systems, databases, applications, and user access from a centralized management console. The tools available from the management platform enable the automation and delegation of routine and complex system tasks.

SNMP manageability of TUXEDO and M3 applications provides the following benefits:

- ◆ Enables you to move towards a single management console and thus provide integrated systems management of M3 or TUXEDO based applications.
- ◆ Enables you to hook TUXEDO or M3 applications into popular management platforms such as HP OpenView, Sun SunNet Manager, IBM NetView/6000. This makes the management of TUXEDO or M3 applications more effective by providing a whole-system perspective instead of piecemeal solutions.
- ◆ Helps preserve your investment in standard, compliant (SNMP-capable) management frameworks.
- ◆ Enables you to benefit by the experience of a large community of SNMP users.
- ◆ Opens up a way for providers of management applications to write management applications for TUXEDO or M3 customers, as SNMP is an open standard.

# SNMP Overview

The Simple Network Management Protocol (SNMP) provides a standard system for classifying system information about hardware, software, and other aspects of a distributed client/server system. The basic SNMP standard for system management is defined in the Network Working Group (NWG) RFC 1157.

SNMP network and systems management is based on the manager/agent model described in the network management standards defined by the International Organization for Standardization (ISO). In this model, a network manager exchanges monitoring and control information about network and system resources with distributed software processes called *agents*.

Any system or network resource that is manageable through this exchange of information is a *managed resource*. This could be a software resource, such as a message queue or TUXEDO application, or a hardware resource such as a router or NFS file server.

Agents function as “collection devices” that typically gather and send data about the managed resource in response to a request from a manager. In addition, agents often have the ability to issue unsolicited reports to managers when they detect certain predefined thresholds or events on a managed resource. In SNMP terminology, these unsolicited event reports are called *trap notifications*.

A manager relies upon a database of definitions and information about the properties of managed resources and the services the agents support — this comprises the Management Information Base (MIB). When new agents are added to extend the management domain of a manager, the manager must be provided with a new MIB component that defines the manageable features of the resources managed through that agent.

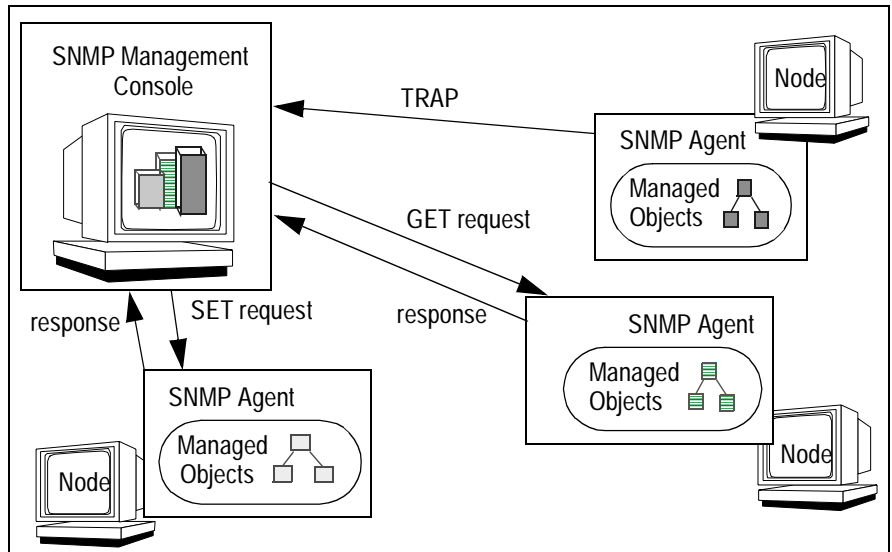
The manageable features of resources, as defined in an SNMP-compliant MIB, are called *managed objects* (also termed management variables or variables). Examples of managed objects include the state of a TUXEDO domain, the number of users currently logged on to a system, the number of physical network interfaces on a router, or a global static variable in an application. When the heterogeneous components of an enterprise’s distributed systems are defined within a common MIB on the management station, this provides a unified perspective and single access point for managing system and network resources.

The data types and the representations of resources within a MIB, as well as the structure of a particular MIB, are defined in a standard called the Structure of Management Information (SMI). This standard is described in the NWG RFC 1155.

A formal language, known as the ISO Abstract Syntax Notation One (ASN.1), is used to describe MIB data independently of any encoding technique used. SNMP uses a subset of the ASN.1 language to represent a MIB.

As illustrated in Figure 1-1, under the SNMP paradigm, there is a management platform that provides operators access to management information from a management console. At the management console, the systems administrator issues management commands to SNMP agents to collect the values of various management variables which are defined in that platform's Management Information Base. The SNMP protocol is based on asynchronous request/response commands. A management station sends a GET or GET-NEXT command to request values of MIB variables from an agent, or a SET request to modify the value of a variable. Once the data is collected, management platforms can present views or graphs of the information or take action in response to the information provided by SNMP agents.

**Figure 1-1 SNMP Management and SNMP Agent Interaction**





Typically, management platforms save the collected data to a repository for historical reporting. They also commonly include various tools and utilities to analyze the management data. The management framework enables the network administrator to automate responses to event-based operations, change access privileges, update application information, and tune application parameters.

The Agent Connection provides several key features for TUXEDO and M3 applications management:

**Monitor**

SNMP monitoring allows any supported SNMP-capable management station to monitor the state of the TUXEDO or M3 system. Also, appropriate actions may be triggered, once a variable crosses a predefined threshold.

**Control**

Using SNMP SET commands, an SNMP-capable management station can modify the value of TUXEDO or M3 control and configuration parameters.

**Alarm Notification**

The TUXEDO or M3 system generates certain system-wide events in case of exceptions (for example, a TUXEDO node going down). These events are trapped and sent as a SNMP trap notification to the management station.

## **The TUXEDO MIB for SNMP**

The TUXEDO system includes management tools and a classification system that identifies its application items in a hierarchy of information known as a TUXEDO Management Information Base (TMIB). Although the term “MIB” is also used to designate the definition of the manageable features of managed resources within the SNMP standard, the TUXEDO TMIB is not compliant with the SNMP standard MIB format. (For more information, refer to “Differences Between the TUXEDO MIB and the SNMP TUXEDO MIB.”)

The Agent Connection for TUXEDO and M3 provides the following components to incorporate TUXEDO and M3 information within an SNMP management framework:

- ◆ An SNMP translation of the TUXEDO and M3 MIB (TMIB), thus making manageable features of TUXEDO components recognizable within an SNMP management framework
- ◆ An agent, `tux_snmpd`, that “speaks” SNMP in order to respond to requests from SNMP managers and to generate SNMP trap notifications for TUXEDO system events
- ◆ `m3_snmpd`, a separate version of the SNMP agent for use with M3 applications

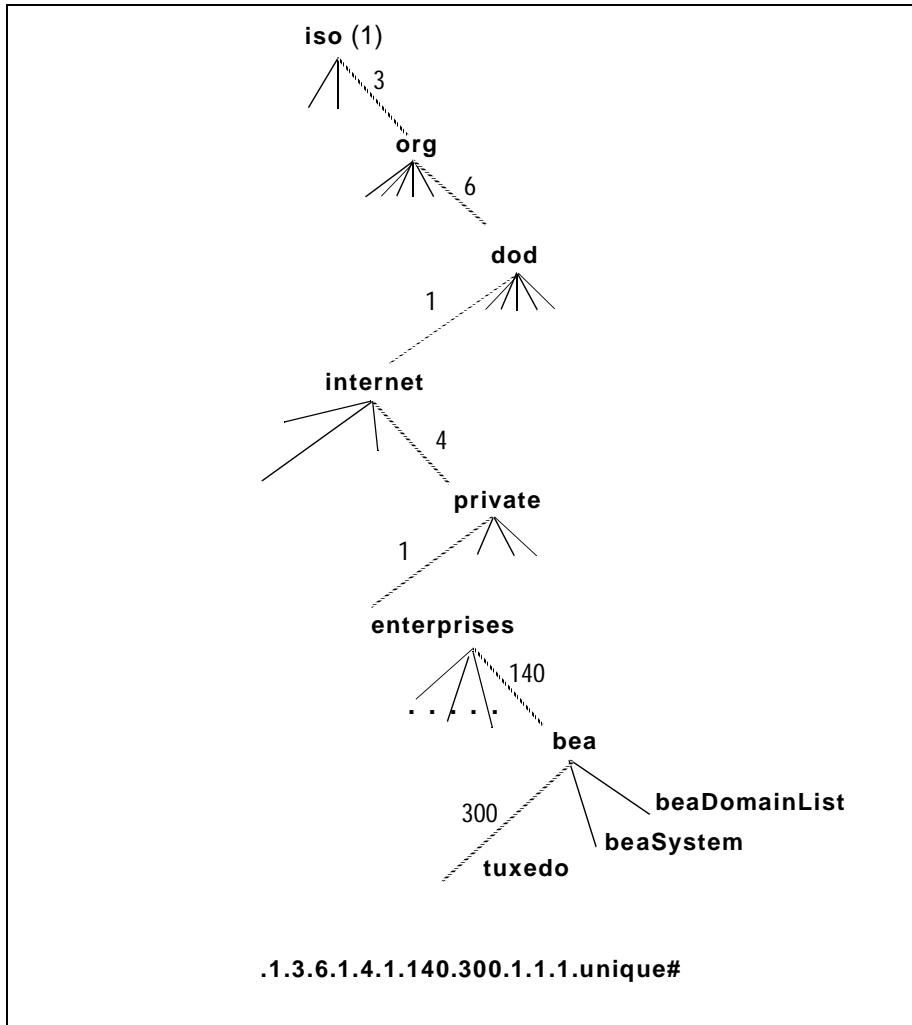
**Note:** If an M3 application is not installed, the M3 specific objects included in the BEA TUXEDO MIB (`bea.asn1`) will not return values.

## MIB Object Identifiers

A MIB structure includes the idea of a hierarchical relationship between managed objects. Inclusive of the overall SNMP standard is a tree hierarchy (MIB tree) for defining groups of managed objects. Each managed object in the MIB is assigned a unique number called an object identifier (OID). As illustrated in Figure 1-2, an object identifier consists of a left-to-right sequence of integers known as sub-identifiers. The sequence defines the location of the object within a MIB tree. By specifying a path to the object through the MIB tree (also known as the registration tree), the OID allows the object to be identified uniquely. As shown in Figure 1-2, each node in the path defined in an OID has both a number and a name associated with it. The digits below the `enterprise` OID in the tree can be any sequence of user-defined numbers chosen by an organization to represent its private MIB groups and managed objects.

The following figure shows an example of the format of the `bea` object identifiers.

Figure 1-2 MIB Object Identifier Hierarchy and Format



In this hierarchy, each BEA private MIB object that the SNMP agent software manages has a unique object identifier. The BEA TUXEDO object uses a prefix of **.1.3.6.1.4.1.140** to identify it as an object in the BEA private MIB.

For more information about the MIB, refer to chapters 4 through 11 of this manual. For a complete listing of objects in the BEA private MIB in ASN.1 notation, read the file `bea.asn1` in the SNMP Agent Connection product.

## ASN.1 File

An ASN.1 file is a standard SNMP file that defines the objects that make up an SNMP-compliant MIB. Each object in the file is defined in compliance with the SNMP standard. The Agent Connection provides the ASN.1 file `bea.asn1` for defining the TUXEDO MIB (with M3 extensions) for SNMP (and other BEA private MIBs).

**Note:** The TUXEDO and M3 MIB definitions are written in concise MIB in accordance with RFC 1212, as required by the SNMP standard.

## MIBs Supported by the SNMP Agent Connection

Within the `bea.asn1` file are definitions for TUXEDO and M3 objects that are SNMP compliant. The Agent Connection supports the following MIBs:

- ◆ **TUXEDO Core MIB.** This MIB supplies the definitions for controlling the operation and configuration of the TUXEDO middleware system. This MIB contains the main information groups for TUXEDO applications, including domains, machines, queues, servers, routing, clients, and services.
- ◆ **BEA Domain List MIB.** This MIB group represents information about the M3 or TUXEDO domains that the agent is monitoring.
- ◆ **M3 MIB.** An extension of the core MIB that incorporates definitions for management of M3 factories and CORBA interfaces.
- ◆ **Access Control List MIB.** This MIB allows you to define and control your application security options.
- ◆ **Workstation or /WS MIB.** This MIB specifies information about TUXEDO client workstations including workstation listeners and handlers.
- ◆ **Application Queue MIB.** This MIB provides the administrative control required for managing access to application queues. The objects in this MIB include items for managing queue spaces, queues, messages, and transactions.
- ◆ **Event Broker MIB.** This MIB represents event subscriptions registered with the Event Broker for receiving event notifications.
- ◆ **Event Traps MIB.** This MIB defines all the trap notifications that will be generated by the Agent Connection and the objects passed in the variable bindings for these traps.

TUXEDO managed object names within the MIBs for SNMP are usually prefixed with the letters `tux`. M3 managed object names within the SNMP MIB are usually prefixed with `m3`. For example, the TUXEDO Core MIB contains a group termed Machine. Within this group are managed objects such as the following:

`tuxTmachinePmid`

Represents a physical machine identifier

`tuxTmachineLmid`

Represents the logical machine identifier

All TUXEDO objects apply to M3 applications also.

## Structure of the MIB Definitions

Chapters 4 through 11 of this manual contain the definitions of the TUXEDO MIBs (with M3 extensions) and serve as a reference for the material in the SNMP Agent Connection `bea.asn1` file. The definitions that make up the TUXEDO MIBs for SNMP are grouped by MIB (Access Control List, Application Queues, Traps, etc.), with the M3 extensions in a separate chapter.

## MIB Object Definitions Format

Chapters 4 through 11 specify the definitions of managed objects supported by the Agent Connection. The following keywords are used to define MIB managed objects:

### Syntax

Defines the abstract data structure corresponding to that object type. The ASN.1 language is used for this purpose. However, the SMI purposely restricts the ASN.1 constructs that may be used. These restrictions are made expressly for simplicity.

### Access

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

**Note:** For tabular objects, a read-write object in some cases can only be set during creation of a new row. When this is true, this will be noted in the Description section for that MIB object.

## Description

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

**Note:** 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.

## MIB Event Trap Definitions Format

Traps are also defined in the `bea.asn1` file. These traps are defined in accordance with RFC 1215, Trap definitions. Chapter 11, “TUXEDO Traps MIB,” specifies the list of traps generated by the TUXEDO or M3 SNMP agent. 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 the traps generated by the TUXEDO or M3 SNMP agent have an enterprise field set to the TUXEDO object identifier. The TUXEDO object identifier is `.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, which are contained within every 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. This is passed in the specific trap ID field of the trap packet (PDU).

**Note:** The value of the generic trap ID field in traps is always set to 6, indicating an enterprise-specific trap.

# Differences Between the TUXEDO MIB and the SNMP TUXEDO MIB

If you are familiar with the TUXEDO MIB (TMIB), the primary difference to note when using the SNMP-based MIB is a difference in terms and a few additional MIB items in the SNMP-based MIB.

The TUXEDO MIB identifies an abstract structure for TUXEDO resources. In a TUXEDO system, a MIB is the classification of information in a TUXEDO application. However, instead of referring to *groups* and *managed objects* as is common in SNMP terminology, the TUXEDO MIB defines application resources as *classes* and *attributes*. Classes are the administrative class definitions that make up the TUXEDO MIB. Each class has a set of attributes that identify individual items in the class. Examples of TUXEDO 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 standard TUXEDO MIB (TMIB), refer to the *TUXEDO Reference Manual*.

In contrast to all this; in SNMP the features of manageable resources are called *objects* rather than attributes, and objects fall under MIB *groups* rather than classes.

# Managing M3 and TUXEDO Applications Using the SNMP MIB

The M3 and TUXEDO systems identify application items in a hierarchy of information known as TUXEDO Management Information Bases (TMIB). These MIBs contain definitions that describe the components found in the M3 or TUXEDO application. Included with the Agent Connection is an SNMP version of the TMIBs. The Agent Connection SNMP MIB also includes MIB objects that represent attributes of M3 resources.

To monitor or modify values of managed objects, using your systems management platform, you will need to know which MIB objects represent the features of TUXEDO or M3 resources that are relevant to your management goals. You will also need to know the data types, default values, and access permissions for these MIB objects. Chapters 4 through 11 of this manual contain all of the reference information necessary to manage TUXEDO or M3 objects.

## Querying Non-Existent MIB Objects

If you attempt to retrieve the value of a MIB object, and that object does not exist, either no value will be returned, or one of the following values will be returned:

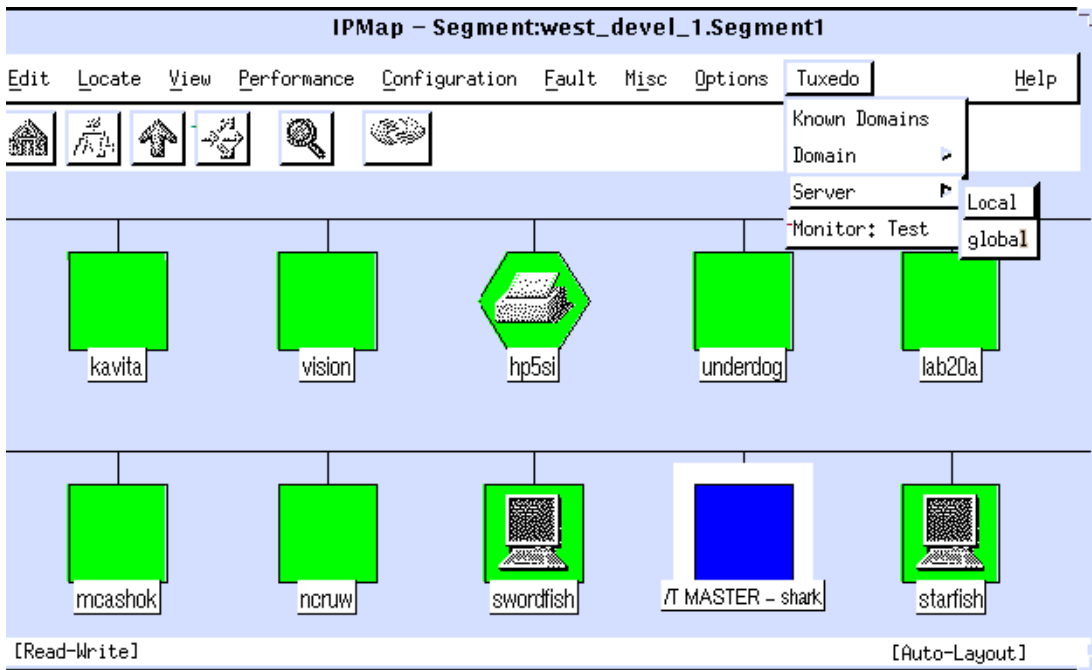
- ◆ -1 if the object is numeric
- ◆ A dash (-) if the object data type is `DisplayString`



## Managing TUXEDO Resources from a Management Platform

Figure 1-3 illustrates a management platform's graphical view of servers on a network, which includes the capability to identify and manage various TUXEDO servers.

**Figure 1-3 Managing TUXEDO Servers**

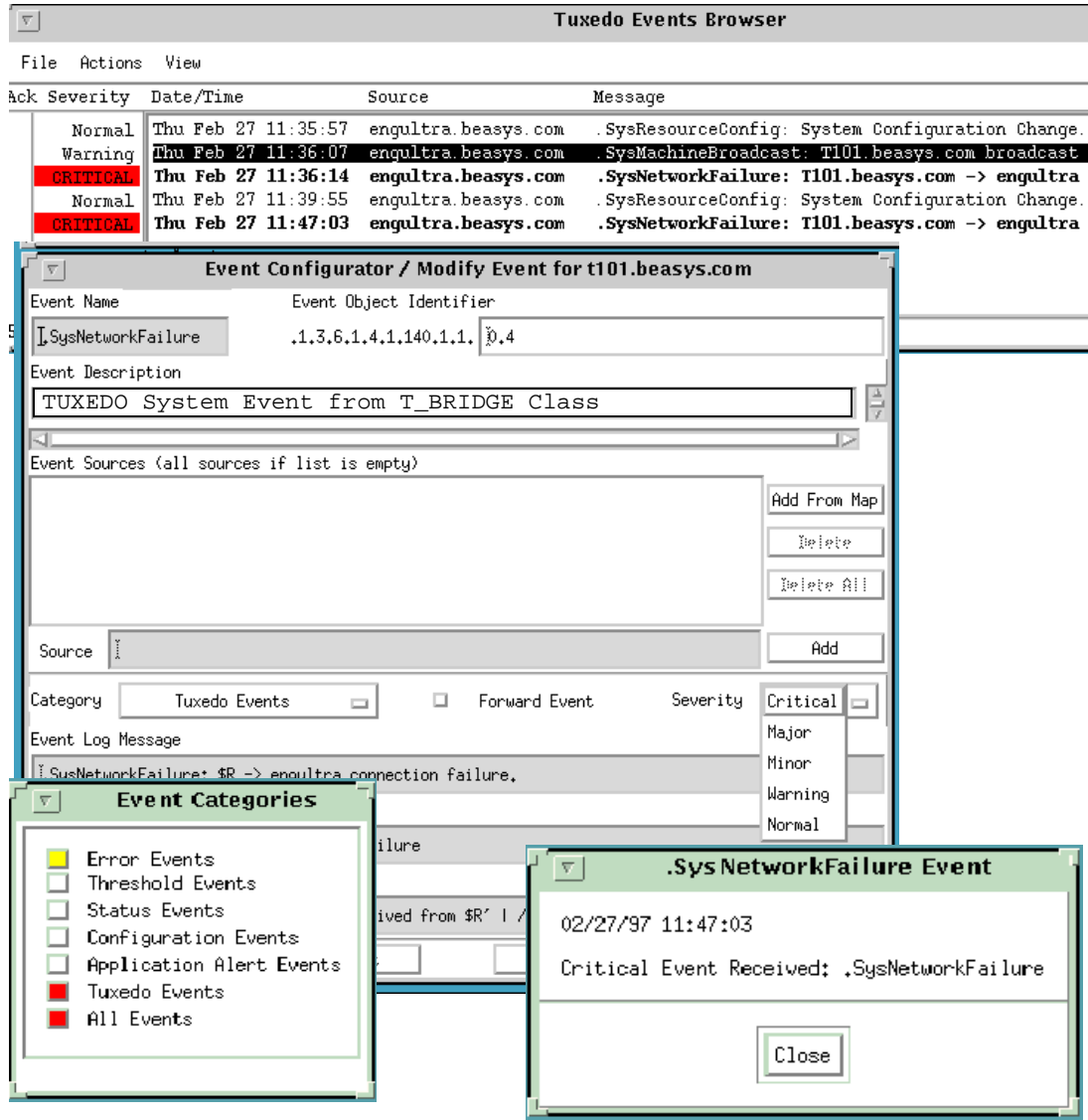


Within most management platforms, you can set up conditions for alarm generation based on your defined event criteria. The criteria typically consists of changes in the values of certain attributes of the managed resources. These attributes are represented as MIB objects. You can also define actions to take when specified events occur, such as when a particular threshold is crossed.

The TUXEDO and M3 MIB for SNMP supports a full range of TUXEDO and M3 system and application events. These system and application events are transmitted as enterprise-specific traps. Figure 1-4 provides an example of how TUXEDO events, as

well as other system events, can be selected in a management platform GUI. Chapter 11, “TUXEDO Traps MIB,” contains information that may help you to interpret TUXEDO specific traps.

**Figure 1-4 Specifying Event and Event Notifications**



Each management platform has its own view of the SNMP-based TUXEDO MIB information. This presentation can differ from platform to platform. However, if your management platform accepts SNMP-compliant MIBs, you can manage all the objects provided in the TUXEDO and M3 MIB for SNMP.

## Getting Started

Integrating the Agent Connection into your management system requires set-up tasks on both the managed node and on the management system. Set-up tasks for the agents on the managed node are described in Chapter 2, “Setting Up the Agents.” Integration with the management system is described in Chapter 3, “Integrating Agent Connection with a Management System.”



# 2 Setting Up the Agents

The Agent Connection facilitates TUXEDO and M3 system management from any SNMP-compliant management platform. Integrating the Agent Connection into your management system requires set-up tasks on both the managed node and on the management system. This chapter describes the procedure for setting up the Agent Connection on the managed node. Integration into the management system is described in Chapter 3, “Integrating Agent Connection with a Management System.”

## Setting Up the Agent Connection

Preparing the Agent Connection for TUXEDO or M3 system management requires the following steps on Windows NT systems:

1. Make sure TUXEDO or M3 is installed.
2. Install the Agent Connection on the managed nodes.

Agent Connection can be installed on a managed node from the BEA Manager CD. The BEA Manager CD contains an installation script for installing the agents on UNIX systems and an installation program for installing the agents on Windows NT machines. For detailed information about how to install the Agent Connection, refer to the *BEA Manager Installation Guide*.

Some attributes of TUXEDO resources are accessible globally (that is, no matter which TUXEDO node they are on) while others are accessible only by an SNMP agent local to the same machine. If you want to access managed objects that are only accessible locally, you must install TUXEDO SNMP agents on each machine where these resources reside. See “Local and Global TUXEDO Information” for more information.

Both the M3 version of the agent (`m3_snmpd`) and the TUXEDO version (`tux_snmpd`) are installed when you install Agent Connection.

### 3. Set up access to M3 or TUXEDO shared binaries.

#### ◆ On Windows NT systems:

If the Agent Connection was not installed in the same directory as the TUXEDO or M3 application, you need to make sure that the `bin` directory of the appropriate TUXEDO or M3 installation is prior to any other TUXEDO or M3 installations in the `PATH` system environment variable. This is to ensure that the Agent Connection has access to the correct TUXEDO or M3 DLLs.

#### ◆ On UNIX systems:

`LD_LIBRARY_PATH` (`SHLIB_PATH` on HP-UX or `LIBPATH` on AIX)

Defines the search path for shared libraries. This variable should include `$TUXDIR/lib`.

### 4. Install the BEA Manager configuration file.

#### ◆ On Windows NT systems:

Copy the BEA Manager configuration file (`beamgr.conf`):

```
md c:\etc
copy installation-directory\etc\beamgr.conf c:\etc
```

#### ◆ On UNIX systems:

Log in as root and copy the BEA Manager configuration file `beamgr.conf` from `installation_directory/etc` to the `/etc` directory.

```
%su
Password:
# cp installation_directory/etc/beamgr.conf /etc
```

### 5. Set your `PATH` to include the location of the BEA Manager executables (on UNIX systems).

All users of the installed BEA Manager products will need to update their `PATH` environment variable to include the location of the BEA Manager executable files. The following is an example in C shell:

```
% set path = ( $PATH installation_directory/bin )
```

### 6. Set your master agent timeout if you are running the agent as a subagent with TUXEDO 6.3 or later, or with M3.

If you are using TUXEDO 6.3 or later, or M3, you need to configure the timeout of your SMUX master, if any (e.g., `snmp_integrator`), and of your SNMP

manager, to at least 30 seconds. For `snmp_integrator`, this can be done by adding a `INTEGRATOR_TIMEOUT` entry to the BEA Manager configuration file (`beamgr.conf`) as follows:

```
INTEGRATOR_TIMEOUT 30
```

7. Ensure match between TCP/IP host name and computer name on Windows NT systems

When Agent Connection is installed on a Windows NT system, make sure that the host name specified in the TCP/IP Properties window, under DNS (Control Panel→Network→Protocols→TCP/IP→Properties→DNS) is the same as specified in the Computer Name field on the Network window, under Identification (Control Panel→Network→Identification). The name should be in all uppercase in both places.

8. Specify the destination for traps (if desired).

The default destination for SNMP trap notifications is `localhost`. If you want traps to be sent to some other destination, use your favorite text editor to modify the BEA Manager configuration file (`beamgr.conf`) `TRAP_HOST` entry to specify the host name of the target destination machine for SNMP trap notifications, and the port number and community name to use in sending traps.

Typically the destination will be the host machine where the SNMP management system is located. Some management systems use distributed trap daemons that “collect” SNMP trap notifications for forwarding to management stations. In that case, the machine with the trap daemon would be the destination.

For more information refer to the “Configuration Files” chapter in the *Agent Integrator Reference Manual*.

9. Identify the domain to be managed.

The identity of the TUXEDO application to be managed can be specified in two ways. Agent Connection uses the following sources in the indicated order of precedence:

- a. The `TAGENT` entry in the BEA Manager configuration file. This entry is of the form:

```
TAGENT logical_agent_name tuxdir tuxconfig_path
```

For more information refer to the “Configuration Files” chapter in the *Agent Integrator Reference Manual*.

- b. `TUXCONFIG` and `TUXDIR` environment variables

10. Ensure that the TUXEDO Event Broker is configured.

The TUXEDO Agent Connection will not receive TUXEDO event notifications unless the TUXEDO Event Broker servers (TMSYSEVT and TMUSREVT) are running. To enable forwarding of TUXEDO events as SNMP traps, ensure that the TUXEDO Event Broker servers are running. Information on the TUXEDO Event Broker can be found in the “Programmed Administration” chapter of the *BEA TUXEDO Administrator’s Guide* and in Section 5 of the *BEA TUXEDO Reference Manual*.

11. Integrate the Agent Connection with your SNMP management system.

The steps for integrating Agent Connection into your management system are described in the next chapter, Chapter 3, “Integrating Agent Connection with a Management System.”

12. Start the TUXEDO or M3 SNMP agents.

Now you can start the SNMP agents on the managed nodes where your TUXEDO or M3 resources are present. Refer to “Starting and Stopping the SNMP Agents.”

# Advanced Configuration

There are additional steps that you may want to take to customize Agent Connection to your needs, such as monitoring multiple TUXEDO domains concurrently or using nondefault ports for communication with the system manager. The following lists a number of optional configuration steps:

1. Define logical agent names if you want to monitor multiple TUXEDO domains concurrently.

To monitor multiple TUXEDO domains at the same time, add a TMAGENT entry to the BEA Manager configuration file for each agent. The TMAGENT entry is of the following form:

```
TMAGENT logical_agent_name tuxdir tuxconfig_path
```

Monitoring of multiple domains is done by running a separate TUXEDO or M3 agent for each domain being monitored. These agents must be run as subagents under the Agent Integrator.



When multiple agents are running on the same node, then SNMP manager SET or GET requests to an agent must be addressed using a community of the form:

*community@logical\_agent\_name*

*logical\_agent\_name* identifies the agent to which the SNMP request is forwarded. For example:

*public@simpapp\_agent*

If only one agent is running on a node, *logical\_agent\_name* is optional in specifying the community in GET or SET requests.

2. Define TUXEDO event filters to use (if desired).

TUXEDO event filters can define a subset of TUXEDO events to be received by the agent for each domain being monitored. You can use TMEVENT\_FILTER entries in the BEA Manager configuration file to define a subset of TUXEDO event notifications that are to be forwarded as SNMP trap notifications. For more information refer to the “Configuration Files” chapter in the *Agent Integrator Reference Manual*. MIB objects corresponding to TUXEDO event filters are described in Chapter 4, “TUXEDO Core MIB.”

3. Specify non-default SNMP communities and SMUX password (if desired).

By default, BEA Manager agents (such as the Agent Integrator, or *tux\_snmpd* or *m3\_snmpd* when running as an SNMP agent) use *public* as the read-only community and *iview* as the write-read community when communicating with SNMP managers. If you want to specify different community names to be used by BEA Manager SNMP agents, this is specified in the BEA Manager passwords file. The passwords file can also be used to specify a password to be used by Agent Integrator for authenticating connection requests from SMUX subagents. To set up the passwords file, do the following:

- a. Copy the BEA Manager passwords file (*beamgr\_snmpd.conf*) to *c:\etc*. For example, on Windows NT issue the following command:

```
copy installation-directory\etc\beamgr_snmpd.conf c:\etc
```

- b. Now you can modify the SNMP communities in this file. The keywords used in this file are:

- ◆ SMUX\_PASSWD
- ◆ COMMUNITY\_RO
- ◆ COMMUNITY\_RW
- ◆ DISABLE\_SET

- c. If you want to set the agent to be read-only, specify a `DISABLE_SET` entry in the passwords file as follows:

`DISABLE_SET YES`

If there is no `DISABLE_SET` entry in the passwords file, the agent has both SET and GET capability.

For more information refer to the “Configuration Files” chapter in the *Agent Integrator Reference Manual*.

4. Specify a SMUX password (if desired) when using Agent Connection as a subagent under a SMUX master agent (such as Agent Integrator).

The environment variable `BEA_SMUX_PASSWD` specifies the password that the SNMP agents (`tux_snmpd` or `m3_snmpd`) use when registering with a SMUX master agent (such as Agent Integrator). This environment variable is required only if the SMUX master agent expects a password. If this environment variable is not set, no password is specified by `tux_snmpd` or `m3_snmpd` when registering.

5. How to use nonstandard ports.

By default, BEA Manager agents assume the following port numbers as specified by SNMP and SMUX standards:

<code>snmp</code>	<code>161/udp</code>
<code>snmp-trap</code>	<code>162/udp</code>
<code>smux</code>	<code>199/tcp</code>

The default port assignments may be sufficient for your needs. If necessary, you can define these services on other ports, or use the appropriate command-line options when starting BEA Manager agents to assign them to nondefault ports.

- ◆ On Windows NT systems:

To modify or define the services, add the appropriate lines in the `NT-root-directory\system32\drivers\etc\services` file. You may wish to consult your system administrator.

## ◆ On UNIX systems:

To modify or define the services, do the following:

- a. Determine if the NIS is running. You can use the `ypwhich` command to determine if an NIS server or map master is available. For example:

```
% ypwhich
zort.kremvax.com
```

- b. If an NIS server is available, you can use the `ypcat` command to determine if the services are available.

```
% ypcat services | grep snmp
snmp-trap      162/udp      snmptrap
snmp           161/udp
```

- c. If an NIS server is not available and services are provided on the local host, you can examine the `/etc/services` file instead.

```
% cat /etc/services | grep snmp
snmp-trap      162/udp      snmptrap
snmp           161/udp
```

Refer to your UNIX system documentation, or consult your UNIX system administrator, for instructions specific to your UNIX platform to establish the SNMP services if necessary.

# Starting and Stopping the SNMP Agents

To enable management access to the TUXEDO or M3 resources represented in the BEA MIBs, you need to start the TUXEDO or M3 SNMP agents.

To monitor multiple TUXEDO or M3 domains, you can run multiple SNMP agents on the same node. Each agent can monitor only one domain. To monitor multiple domains, you must have the BEA Manager Agent Integrator running and the agents must be started as subagents.

On startup, a TUXEDO or M3 SNMP agent checks for a `TMAGENT` entry in the BEA Manager configuration file that matches its logical agent name. A `TMAGENT` entry provides a path to the TUXEDO or M3 domain to be monitored. If no matching `TMAGENT` entry is found, the agent connects to the TUXEDO domain specified in the `TUXCONFIG` and `TUXDIR` environment variables. The agent exits if the `TUXCONFIG` or `TUXDIR` environment variable is not defined and no appropriate `TMAGENT` entry is found in the BEA Manager configuration file. For more information refer to the “Configuration Files” chapter in the *Agent Integrator Reference Manual*.

## Starting the SNMP Agents on UNIX Platforms

To start the SNMP agents on a UNIX system, type the TUXEDO or M3 SNMP startup command at the command-line prompt.

For the TUXEDO SNMP agent, the syntax of the startup command is:

```
tux_snmpd [-l logical_agent_name] [-d] [-n] [-s] [-p snmp_port]  
[-r smux_port] [-m hostname] [-h]
```

For the M3 SNMP agent, the syntax of the startup command is:

```
m3_snmpd [-l logical_agent_name] [-d] [-n] [-s] [-p snmp_port]  
[-r smux_port] [-m hostname] [-h]
```

## Startup Options

The command options have the following interpretation:

`-l logical_agent_name`

Separate logical agent names must be assigned if you want to run multiple instances of the agent on the same node. If the `-l` option is not specified, the name of the executable is used as the logical agent name.

*logical\_agent\_name* is a string that associates an agent with a TUXEDO domain as defined by a TMAGENT entry in the BEA Manager configuration file (*beamgr.conf*). The logical agent name can be a maximum of 32 characters in length. The format of the TMAGENT entry is as follows:

```
TMAGENT logical_agent_name tuxdir tuxconfig
```

This entry assigns the agent started with *logical\_agent\_name* to the indicated TUXEDO domain. Refer to the “Configuration Files” chapter in the *Agent Integrator Reference Manual*.

`-d`

This option dumps to standard output the SNMP or SMUX packets received and sent by the agent.

`-n`

If the agent/subagent is run with this option, it does not become a daemon. This option is normally used if *tux\_snmpd* (or *m3\_snmpd*) is to be started by *init* using the *inittab* table with the *respawn* option.

`-s`

Specifies *tux\_snmpd* or *m3\_snmpd* to run as an SNMP agent. If this option is not specified, *tux\_snmpd* or *m3\_snmpd* runs as a SMUX subagent.

`-p snmp_port`

*snmp\_port* designates the UDP port on which *tux\_snmpd* or *m3\_snmpd* listens for incoming SNMP packets. This option allows running the *tux\_snmpd* or *m3\_snmpd* on a port other than the standard SNMP port 161. This option is meaningful only when *tux\_snmpd* or *m3\_snmpd* is running as an SNMP agent.

`-r smux_port`

This option specifies the TCP port to use to connect to a SMUX master agent. (The default is port 199.) This option is meaningful only when *tux\_snmpd* or *m3\_snmpd* is running as a SMUX subagent.

`-m hostname`

*hostname* is the name of the machine where the SMUX master agent (such as the Agent Integrator) is running. This option is used only when you want `tux_snmpd` or `m3_snmpd` to register with a SMUX master agent on a remote machine.

`-h`

Causes the agent to print the syntax for the `tux_snmpd` command.

### Description of `tux_snmpd` and `m3_snmpd` Commands

The `tux_snmpd` binary is the TUXEDO SNMP agent which supports the TUXEDO MIB. For a description of the supported MIB groups and objects, please refer to chapters 4 through 11 in this manual.

The `m3_snmpd` binary is the M3 SNMP agent which supports the TUXEDO MIB with M3 extensions. For a description of the supported M3-specific MIB groups and objects, refer to Chapter 6, “M3 MIB Groups.”

This agent is capable of running as an SNMP agent or as an SMUX subagent.

When the `tux_snmpd` or `m3_snmpd` starts up as an SNMP agent, it generates a `coldStart` trap. The destination host, port, and community used when sending traps are as specified in the `TRAP_HOST` entry in the BEA Manager configuration file (`beamgr.conf`).

SNMP read-write and read-only communities supported by the `tux_snmpd` or `m3_snmpd` can be specified in the BEA Manager passwords file (`beamgr_snmpd.conf`). By default, the read-only community is `public` and the read-write community is `iview`. Community is meaningful only when the agent is running as an SNMP agent.

When running as an SMUX subagent, `tux_snmpd` or `m3_snmpd` specifies a password to the SMUX master agent at the time of registration if the environment variable `BEA_SMUX_PASSWD` has been defined. In that case, the value of `BEA_SMUX_PASSWD` is used as the password by `tux_snmpd` or `m3_snmpd`. If `BEA_SMUX_PASSWD` has not been defined, `tux_snmpd` or `m3_snmpd` does not specify a password to the master agent when registering.

`tux_snmpd` and `m3_snmpd` support the MIB-II `snmp` group when running as an SNMP agent.

Like other TUXEDO clients, `tux_snmpd` needs access to the TUXEDO shared libraries. For M3 environments, `m3_snmpd` needs access to the M3 shared libraries. For UNIX platforms, the user should make sure that the environment variable `LD_LIBRARY_PATH` is defined appropriately. (For HP-UX users, the variable name is `SHLIB_PATH`. For AIX users, the variable name is `LIBPATH`.)

## Starting the SNMP Agents on Windows NT Systems

To start SNMP agents on a Windows NT system, do the following:

1. Install additional Windows NT services, if you want to run multiple agents on a single node.

The installation program for Windows NT installs the SNMP agent as a single Windows NT service. If you want to run multiple instances of the agent to monitor multiple M3 or TUXEDO domains, you will need to install additional Windows NT services for the additional agents. Run the following commands for each additional TUXEDO SNMP agent:

```
instsrv logical_agent_name  
install_directory\bin\tux_snmpd.exe
```

Alternatively, run the following commands for each additional M3 SNMP agent:

```
instsrv logical_agent_name  
install_directory\bin\m3_snmpd.exe
```

Separate logical agent names must be assigned if you want to run multiple instances of the agent on the same node. To use multiple agents to monitor multiple TUXEDO or M3 domains, *logical\_agent\_name* is a string that associates an agent with a TUXEDO domain as defined by a TMAGENT entry in the BEA Manager configuration file (`beamgr.conf`). The format of the TMAGENT entry is as follows:

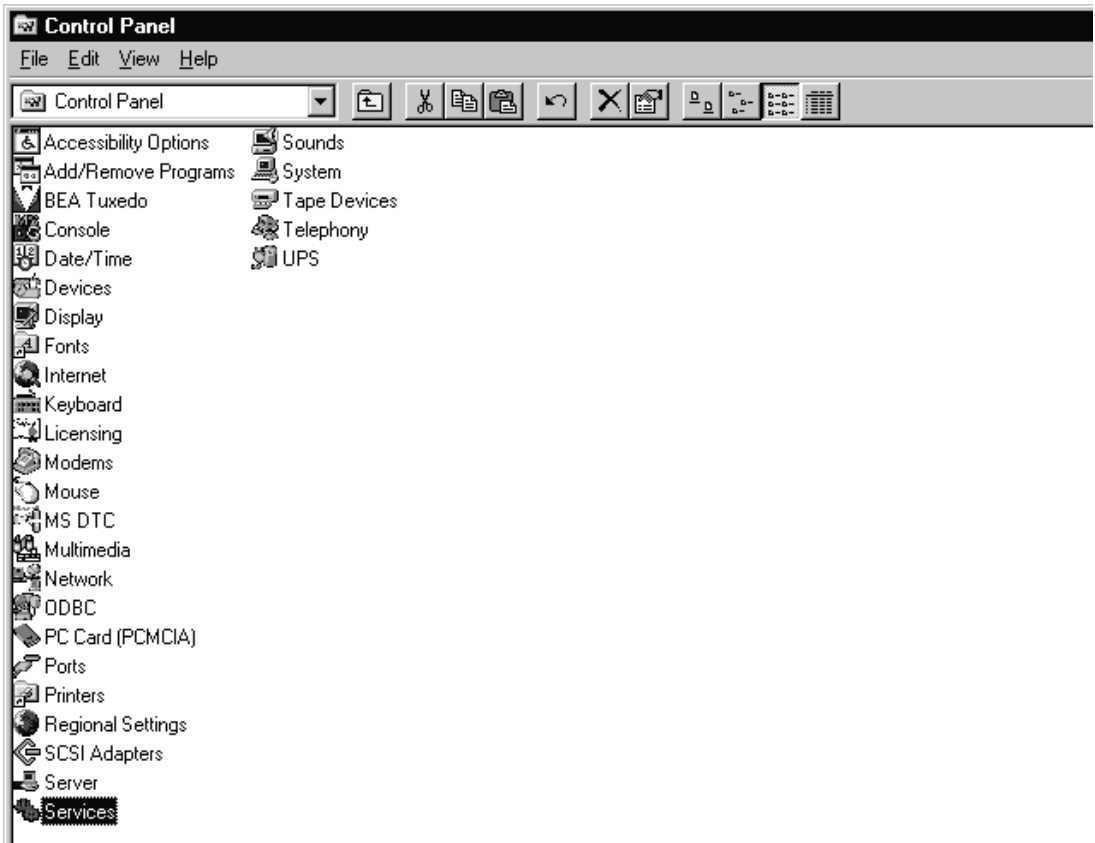
```
TMAGENT logical_agent_name tuxdir tuxconfig
```

This entry assigns the agent started with *logical\_agent\_name* to the indicated TUXEDO or M3 domain. Refer to the “Configuration Files” chapter in the *Agent Integrator Reference Manual*.

2. Start the TUXEDO or M3 SNMP agent from the Services control panel.

On the Windows taskbar, choose Start→Settings→Control Panel. In the Control Panel window, double-click on the Services applet (as shown in Figure 2-1).

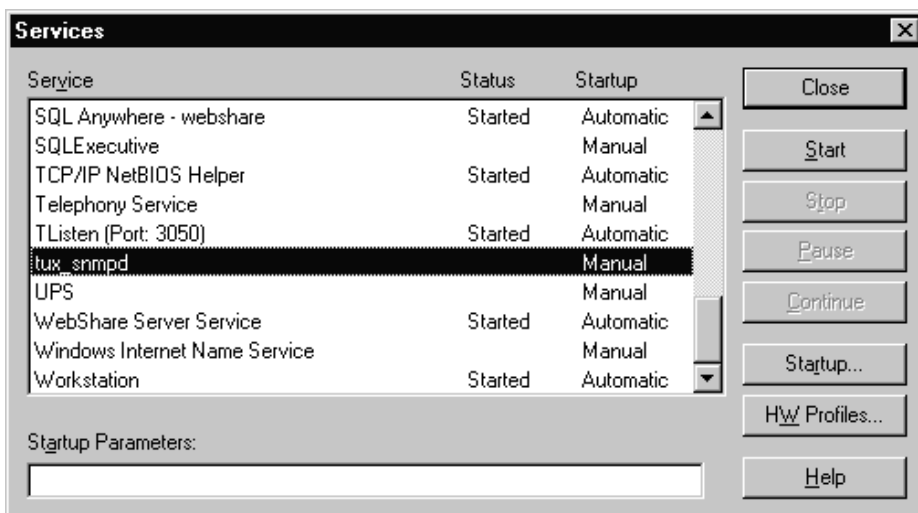
**Figure 2-1** Selecting Services from the Control Panel





In the list of Services, locate and select the installed service (`tux_snmpd` or `m3_snmpd`) and click **Start** to start it (as shown in Figure 2-2. There may be a short delay as the service is initiated.

**Figure 2-2 Starting a Service from the Services Applet**



## Startup Options

-d

If this option is specified, the SNMP or SMUX packets received and sent by the agent are dumped to the Windows NT Event Log.

-s

Specifies `tux_snmpd` or `m3_snmpd` to run as an SNMP agent. If this option is not specified, `tux_snmpd` or `m3_snmpd` runs as a SMUX subagent. If a SMUX master agent (e.g., `snmp_integrator`) is not running, the user must provide `-s` as a start-up parameter before selecting the **Start** button.

-p *snmp\_port*

*snmp\_port* designates the UDP port on which `tux_snmpd` or `m3_snmpd` listens for incoming SNMP packets. This option allows running the `tux_snmpd` or `m3_snmpd` on a port other than the standard SNMP port 161. This option is meaningful only when `tux_snmpd` or `m3_snmpd` is running as an SNMP agent.

`-r smux_port`

This option specifies the TCP port to use to connect to a SMUX master agent. (The default is port 199.) This option is meaningful only when `tux_snmpd` or `m3_snmpd` is running as a SMUX subagent.

`-m hostname`

*hostname* is the name of the machine where the SMUX master agent (such as the Agent Integrator) is running. This option is used only when you want `tux_snmpd` or `m3_snmpd` to register with a SMUX master agent on a remote machine.

## Stopping the Agents

The following command is used to stop one or more BEA Manager agents:

```
stop_agent logical_agent_name | all [logical_agent_name]
```

For example,

```
stop_agent tux_snmpd
```

If you specify `all`, all BEA Manager agents (including any agents built using the BEA Manager Agent Development Kit) will be stopped. The name of the executable is the default logical agent name.

## TUXEDO Master and Non-Master Nodes

The TUXEDO SNMP agent may be installed on both TUXEDO master and non-master nodes. If the TUXEDO application is down on the non-master node, SNMP GET requests addressed to the SNMP agent on the non-master node may not have the latest information. This would be the case, for example, if the requested information has been updated on a master node after the application on the non-master node went down.

Also, SET requests to a non-master node are not allowed if the TUXEDO application is down on the local node.

## Local and Global TUXEDO Information

Some MIB groups in the TUXEDO MIB return values for all TUXEDO nodes whereas other MIB groups return data only for the local node. Thus, if you want to manage objects whose values are local to a particular machine, you must install a copy of the TUXEDO SNMP agent on that machine.

The following MIB groups or tables return only values local to the managed node:

- ◆ `tuxTwshTbl` — This table represents runtime attributes of workstation handler (WSH) client processes.
- ◆ `tuxTmachineActive` group — This group represents runtime statistics on a machine where some component of the TUXEDO application is active.
- ◆ `tuxTmsgTable` — This table represents runtime attributes of the TUXEDO System/T UNIX system message tables.
- ◆ `tuxTqueueTable` — This table represents runtime attributes of queues in an application.
- ◆ `tuxTAppQTbl` — This table represents attributes of application queues.
- ◆ `tuxTAppQmsgTbl` — This table represents attributes of messages stored in application queues.
- ◆ `tuxTQspaceTbl` — This table represents attributes of application queue spaces.
- ◆ `tuxTQtransTbl` — This table represents runtime attributes of transactions associated with application queue spaces.
- ◆ `tuxTBridgeTbl` — This table represents status and statistics pertaining to connections between machines making up an application.
- ◆ `tuxTclientTbl` — This table represents runtime attributes of active clients within an application.
- ◆ `tuxTconnTable` — This table represents runtime attributes of active conversations within an application.
- ◆ `tuxTdeviceTbl` — This table represents configuration and runtime attributes of raw disk slices or UNIX system files being used to store TUXEDO System/T device lists.

- ◆ `tuxTsrvrTblExt` — This table, an extension of `tuxTsrvrTbl`, represents attributes of servers within an application.
- ◆ `tuxTsvcGrp` — This group represents configuration attributes of services within an application.
- ◆ `tuxTranTbl` — This table represents runtime attributes of active transactions within the application.
- ◆ `m3LclIfQueueTable` — This table represents the local runtime attributes of an interface for a particular server queue.
- ◆ `m3LclInterfaceTable` — This table represents configuration and runtime attributes of CORBA interfaces for the local host on which the Agent Connection is running.

## Updating MIB Objects

Some objects in the BEA SNMP MIB for TUXEDO systems can be set (updated) only under certain states of the TUXEDO system. If you get an error while trying to set read-write objects in this MIB, please refer to the `ULOG` file. For details refer to `TM_MIB(5)` in the *TUXEDO Reference Manual*.

## Disabling SET Access

Access of a TUXEDO or M3 SNMP agent to managed resources can be made read-only, regardless of the community (password) used in SNMP requests, by disabling SET access. To do so, add the following line to the BEA Manager passwords file (`beamgr_snmpd.conf`):

```
DISABLE_SET YES
```

The default is for SET access to be enabled. This will take effect only if the change is made prior to starting the agent, or prior to re-initializing the agent using the `reinit_agent` command.

# 3 Integrating Agent Connection with a Management System

This chapter describes the steps to integrate the Agent Connection into your management system. Using the Agent Connection with your management system requires the following steps:

1. Load the SNMP MIB for TUXEDO and M3 into the management system.

The MIB defines the data types and access permissions for the various managed objects that can be accessed through the Agent Connection. It also defines the event notifications that can be generated by the Agent Connection. The MIB thus provides the management system with information it requires to manage TUXEDO and M3 resources.

By default, this file is installed in the *installation\_directory/etc* directory. This MIB must be imported into the management database of your management platform. Some management platforms refer to this process as loading a MIB. Refer to the *BEA Manager Release Notes* for a list of management platforms tested with the TUXEDO Agent Connection.

2. Decide what kind of information you need in order to meet your system management goals.

For example, are there particular attributes of the resources you are managing that you want to monitor? Do you want to be notified when certain TUXEDO system events occur?

3. Configure the management system response to incoming TUXEDO system events.

This is described in the next section, “Integrating M3 and TUXEDO Event Notifications.”

4. Configure polling or data collection rules on the manager for performance and fault management.

Periodic collection of values of pertinent objects is valuable for analysis of trends. This is valuable for capacity-planning and load-balancing. Polling can also be used to generate alarms which is useful for fault management.

5. Define Agent Integrator polling rules (optional)

If you are using the Agent Connection as a subagent with the Agent Integrator, you may want to offload some threshold checking to the Agent Integrator. The Agent Integrator generates enterprise-specific traps when the user-defined threshold is crossed. Offloading checking of selective thresholds to the Agent Integrator reduces the network bandwidth consumed by the management station’s polling activities.

6. Configure and start the agents.

The procedure for configuring and starting the SNMP agents is described in Chapter 2, “Setting Up the Agents.”

# Integrating M3 and TUXEDO Event Notifications

To integrate the M3 and TUXEDO system event traps with your management platform, perform the following actions:

1. Make sure the TUXEDO Event Broker server (TMSYSEVT) is running for the domain being managed.

The Agent Connection will not receive event notifications unless the Event Broker server (TMSYSEVT) is running. Information on the TUXEDO Event Broker can be found in Section 5 of the *BEA TUXEDO Reference Manual*.

2. Modify the TRAP\_HOST entry in the BEA Manager configuration file (`beamgr.conf`) to specify the location of the management machine that is to be the destination for traps generated by the agent.
3. Load the BEA Manager SNMP MIB file (`bea.asn1`) into your management platform (if you have not already done so).

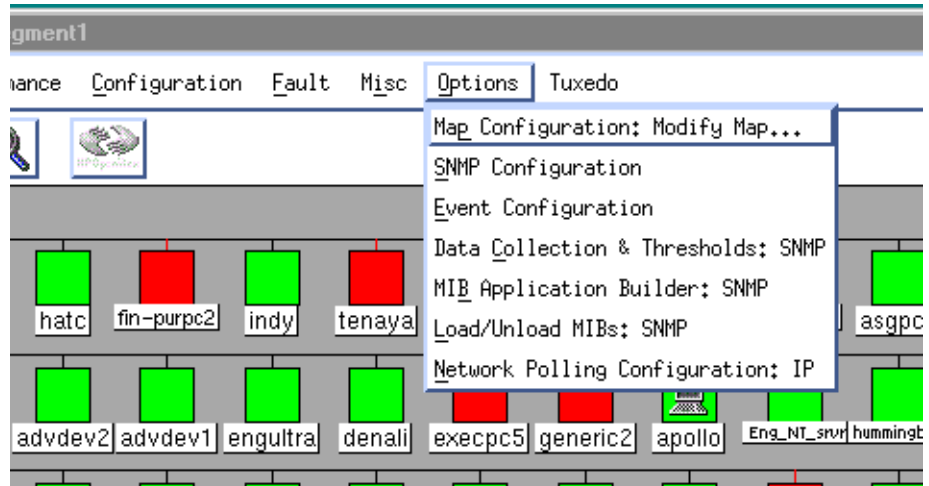
For example, on HP OpenView Network Node Manager, do the following:

- a. Select Options→Load/Unload MIBs: SNMP (Figure 3-1).
- b. Select Load.
- c. Specify the path to the BEA Manager configuration file (`bea.asn1`). By default, this file is installed in:

`install_directory\etc\bea.asn1`

- d. Select OK.

**Figure 3-1 Selecting Load/Unload MIBs in OpenView**



4. Configure the management system to take the appropriate action in response to incoming TUXEDO SNMP traps.

You may want to change the way in which TUXEDO SNMP traps are displayed on your management console, or the actions that the management system takes in response to specified events. For example, you might choose to ignore some routine informational notifications. For example, to view the event configuration on HP OpenView, do the following:

- a. Select Options→Event Configuration.
- b. Select the enterprise tuxedo.
- c. Select an event type

In Figure 3-2, networkFlowTrap is the selected event type.



Figure 3-2 OpenView Event Configuration

The screenshot displays the 'Event Configuration for t101.beasys.com' window. It features a menu bar (File, Edit, View, Help) and a title bar. The main area is divided into two panes. The top pane, titled 'Enterprise Identification', shows a list of enterprises with columns for 'Enterprise Name' and 'Enterprise ID'. The bottom pane, titled 'Event Configurator / Modify Event for t101.beasys.com', is further divided into sections for 'Event Name', 'Event Object Identifier', 'Event Description', 'Event Sources', and 'Event Log Message'.

Enterprise Name	Enterprise ID
rmon	.1.3.6.1.2.1.16
ENTERPRISES	.1.3.6.1.4.1
OpenView	.1.3.6.1.4.1.11.2.17.1
BEA	.1.3.6.1.4.1.140.1.1
tuxedo	.1.3.6.1.4.1.140.300
snmpTraps	.1.3.6.1.6.3.1.1.5

Event Name	Event Object Identifier
networkFlowTrap	.1.3.6.1.4.1.140.300.0.15

**Event Description**

Variables:

- 1: tuxEventsName  
Syntax="Display String"  
Descr=" A string that uniquely identifies this event. All system-generated events begin with '\,Sys'."
- 2: tuxEventsSeverity  
Syntax="Integer"  
Descr=" Indicates the severity of the system event."
- 3: tuxEventsLmid  
Syntax="Display String (SIZE(1,,30))"

**Event Sources (all sources if list is empty)**

Source: I

Category: Log only ☐ Forward Event

Severity: Normal

**Event Log Message**

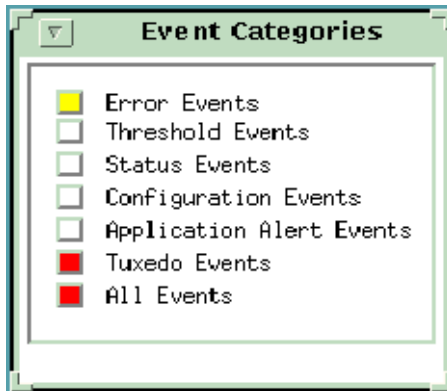
NO FORMAT DEFINED

**Popup Notification (Optional)**

**Command for Automatic Action (Optional)**

- d. Customize the management system response to incoming events of that type. Select Edit→Modify Event in OpenView. This invokes the Event Configuration window (Figure 3-2). You can modify the event configuration to ignore an event, or generate a pop-up notification or run a program or script when the event is received. You might also want to create a separate category for TUXEDO events, as shown in Figure 3-3.

**Figure 3-3 OpenView Event Categories Window**



## Retrieving or Modifying Object Values when Managing Multiple Domains

Monitoring of multiple domains is done by running a separate TUXEDO or M3 agent for each domain being monitored. These agents must be run as subagents under the Agent Integrator.

When more than one M3 or TUXEDO SMP agent is running on a node, then SNMP manager GET or SET requests to an agent must be addressed using a community of the form:

```
community@logical_agent_name
```

For example:

```
public@payrollagent
```

In this example `payrollagent` is a logical agent name that identifies the agent to which the request is to be forwarded by the Agent Connection.

# Integrating Events Generated by Agent Integrator Polling

Agent Integrator can be used to poll TUXEDO or M3 objects, or other managed resources. To integrate the Agent Integrator threshold-checking activity with the management system, do the following:

1. Set up the Agent Integrator polling rules.

A polling rule is defined by a `RULE_ACTION` entry in the BEA Manager configuration file (`beamgr.conf`). This is described in the *Agent Integrator Reference Manual*.

2. Configure the management system to recognize the events generated by Agent Integrator.

For example, in HP OpenView, you can add a new event type by doing the following:

- a. Select Options→Event Configuration
- b. Select `beaSystemDescr`
- c. Select Edit→Add Event.

In the window that is invoked you would use the following as the event number:

`.1.3.6.1.4.1.140.1.1.0.specific_trap_number`

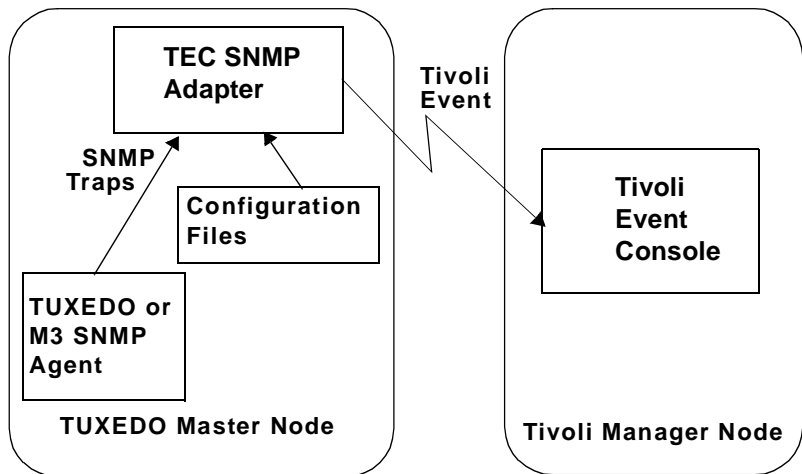
3. Configure the management system to respond appropriately to incoming Agent Integrator events.

This involves essentially the same process as described in Step 4 in the previous section.

# Integrating the Agent Connection with the Tivoli TME Platform

The Agent Connection for TUXEDO and M3 has the ability to translate TUXEDO system event notifications into SNMP trap notifications. These traps can be sent to the Tivoli Event Console (TEC) using the TEC SNMP Adapter. The TEC SNMP Adapter must be installed on the TUXEDO master node along with the Agent Connection, as illustrated in Figure 3-4.

**Figure 3-4 Integrating Agent Connection with the Tivoli Event Console**



To integrate the Agent Connection with the Tivoli Event Console, do the following:

1. Install the TEC SNMP Adapter.

The Tivoli Event Console SNMP Adapter needs to be installed — preferably on the TUXEDO master node.

2. Configure the TEC SNMP Adapter.

The following is a suggested mapping of TUXEDO to Tivoli events. You may need to modify this mapping to meet your own requirements.

- a. To provide a class definition for TUXEDO events, copy the following lines from the file `bea2tiv.baroc`, provided with Agent Connection, to the file `tecad_snmp.baroc` of the TEC SNMP Adapter.

```
TEC_CLASS :
    Tux_Event ISA Specific_SNMP_Trap
    DEFINES {
        severity: default = WARNING;
        class: STRING;
        ulogcat: STRING;
        ulogmsgnum: INTEGER;
    };
END
```

- b. To provide a class description for TUXEDO events, copy the following lines from the file `bea2tiv.cds`, provided with Agent Connection, to the file `tecad_snmp.cds` of the TEC SNMP Adapter:

```
CLASS Tux_Event
    SELECT
        1: ATTR(=, "tuxEventsName" ) ;
        2: ATTR(=, "tuxEventsLmid" ) ;
        3: ATTR(=, "tuxEventsTime" ) ;
        4: ATTR(=, "tuxEventsDescription" ) ;
        5: ATTR(=, "tuxEventsClass" ) ;
        6: ATTR(=, "tuxEventsUlogCat" ) ;
        7: ATTR(=, "tuxEventsUlogMsgNum" ) ;
    MAP
        source = "TUXEDO";
        enterprise = "tuxedo";
        sub_source = $V1;
        hostname = $V2;
        date = $V3;
        msg = $V4;
        class = $V5;
        ulogcat = $V6;
        ulogmsgnum = $V7;
END
```

- c. To provide the OID definitions needed for TUXEDO events, copy the following lines from the file `bea2tiv.oid`, provided with Agent Connection, to the file `tecad_snmp.oid` of the TEC SNMP Adapter:

```
"tuxedo"                "1.3.6.1.4.1.140"
"tuxEventsName"         "1.3.6.1.4.1.140.300.2.6.1"
"tuxEventsSeverity"     "1.3.6.1.4.1.140.300.2.6.2"
"tuxEventsLmid"         "1.3.6.1.4.1.140.300.2.6.3"
"tuxEventsTime"         "1.3.6.1.4.1.140.300.2.6.4"
"tuxEventsUsec"         "1.3.6.1.4.1.140.300.2.6.5"
```

```
"tuxEventsDescription"    "1.3.6.1.4.1.140.300.2.6.6"  
"tuxEventsClass"         "1.3.6.1.4.1.140.300.2.6.7"  
"tuxEventsUlogCat"       "1.3.6.1.4.1.140.300.2.6.8"  
"tuxEventsUlogMsgNum"    "1.3.6.1.4.1.140.300.2.6.9"
```

3. Set up trap destination for BEA Manager.

Ensure that the TRAP\_HOST entry in the BEA Manager configuration file (`beamgr.conf`) points to the host where the Tivoli Event Console SNMP Adapter is running.

4. Start the TUXEDO or M3 SNMP agents.

Refer to Chapter 2, “Setting Up the Agents,” for more information.

5. Set up the Tivoli Event Server for TUXEDO events.

Set up the rules on the Tivoli management system that are to be applied to incoming TUXEDO system events. This is essentially the same process as described in Step 4 under “Integrating M3 and TUXEDO Event Notifications.”

6. Set up the Tivoli Event Console for display of TUXEDO events.

You may need to configure the Tivoli management system to select which TUXEDO events to display and how they should be displayed.

## Distributed Monitoring of TUXEDO or M3 Applications

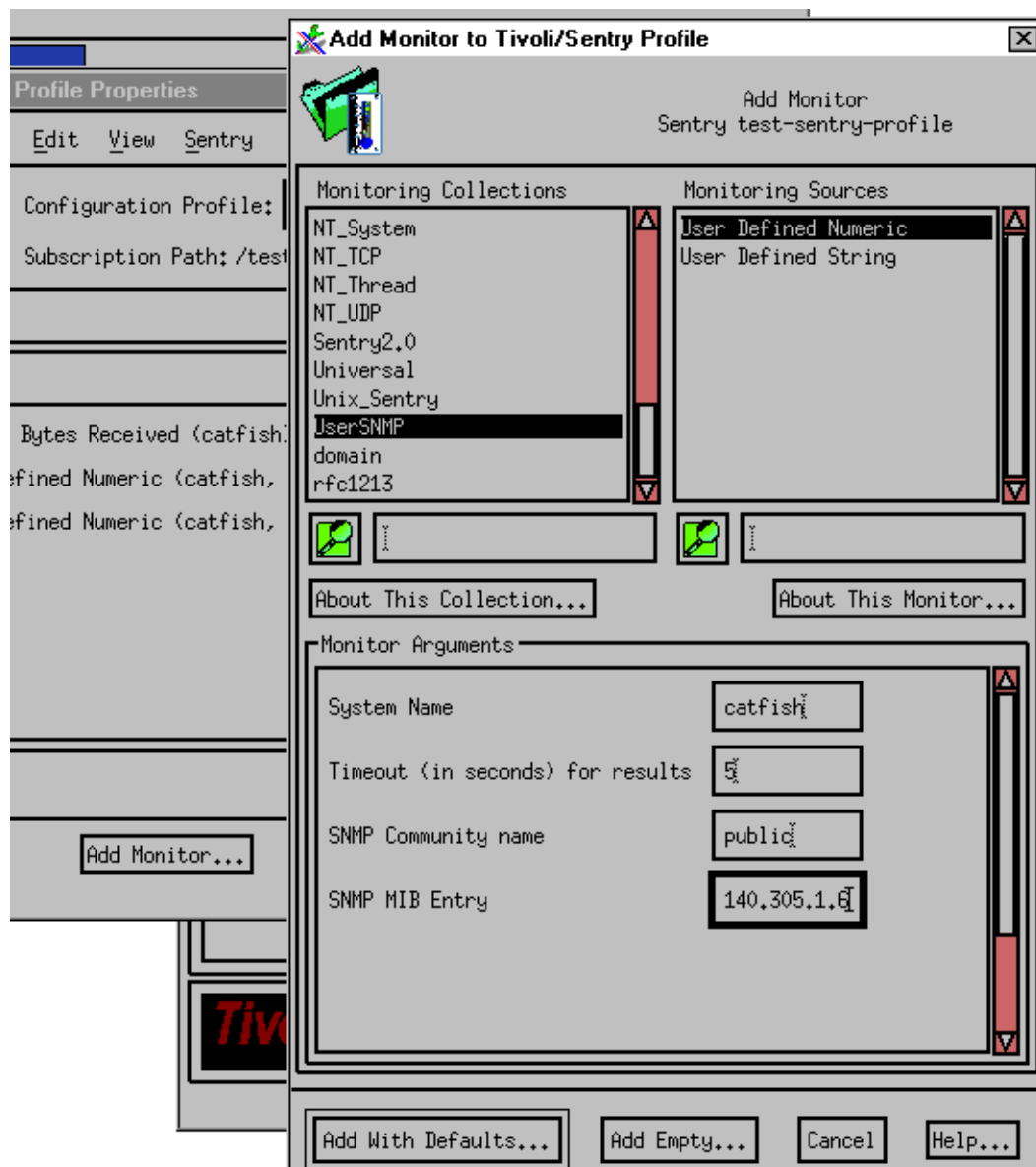
Tivoli has the ability to distribute polling or data collecting to distributed Tivoli agents. The following steps are an example of how to implement distributed monitoring of TUXEDO or M3 MIB objects on the Tivoli platform:

1. Create a new monitor that specifies the object you want to monitor.

The Profiles Property window lists the monitors which you have configured to do threshold-checking. Select Add Monitor from the Profile Properties window.

2. Select the `UserSNMP` monitoring collection, as shown in Figure 3-5.

Figure 3-5 Adding a New Monitor to a Tivoli Profile



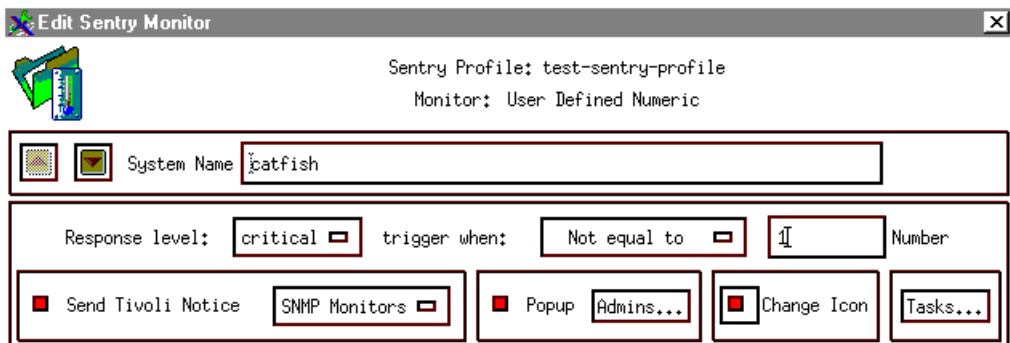
You can define the frequency of polling, the SNMP community, and the object that you wish to monitor. The object must be specified using the absolute object identifier (OID). In the example we have used .1.3.6.1.4.1.140.305.1.6.0, which is the OID for beaDomainStatus. In this case we are setting up a monitor to check whether a TUXEDO domain is active.

In the case of columnar objects, you need to know which instance to monitor. For information on how to specify an instance using an object identifier, refer to the “Polling” chapter in the *Agent Integrator Reference Manual*.

#### 3. Define the polling threshold and the desired response.

Select Add Empty to invoke the Monitor window. This allows you to specify the polling threshold, and other properties of the monitor, such as the severity of the Tivoli event that is generated when the threshold is crossed. In Figure 3-6, we specify a threshold of Not equal to 1 because we want to know when the domain is not active.

**Figure 3-6 Defining a Polling Threshold**



#### 4. Distribute the monitor to the Tivoli agents.

Once you have defined a monitor that specifies the polling interval and condition being checked, you can deploy this monitoring profile to the Tivoli agents for distributed monitoring. Consult the Tivoli documentation for further information.



# 4 TUXEDO Core MIB

The TUXEDO system Core MIB defines the set of groups through which the fundamental aspects of an application may be configured and managed. This includes management of machines, servers, networking, and load balancing.

The TUXEDO Core MIB defines the basic objects that form a TUXEDO application. The Core MIB is the main information repository to control the operation and configuration of the application. When an application is active, the Core MIB contains groups related to the runtime 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
tuxTBridgeTbl	Network connection
tuxTclientTbl	Client
tuxTconnTable	Conversation
tuxTdeviceTbl	Device
tuxTwhichCfgDev	Control MIB for tuxTdeviceTbl
tuxTdomain	Domain information
tuxTgroupTable	Server group
tuxTmachineTable	Machine configuration attributes
tuxTmachineActive	Runtime machine characteristics
tuxTmsgTable	Message queue
tuxTqueueTable	Server queue

Group Name	Description
tuxTroutingTable	Routing criteria
tuxTsrvrTbl	Server configuration attributes
tuxTsrvrTblExt	Server runtime characteristics
tuxTsvcTbl	Service
tuxTsvcGrp	Service-group configuration attributes
tuxTlistenTbl	/T listeners
tuxTranTbl	Transaction
tuxTulogTable	Userlog
tuxTulogCtrl	Control filter MIB for tuxTulogTable
tuxTnetMapTbl	Maps logical machine IDs to network groups
tuxTnetGrpTbl	Application attributes of network groups

# tuxTBridgETbl

This group represents runtime attributes pertaining to connectivity between logical machines making up an application. These attribute values represent connection status and statistics. Objects in this table are only accessible through a TUXEDO SNMP agent installed on the local 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.

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

### tuxTBridgeLmid

Syntax	<i>DisplayString</i> (SIZE(1..61))
Access	read-only
Description	<i>DisplayString</i> is of the format: <i>LMID1</i> [ , <i>LMID2</i> ]
	<p><i>LMID1</i></p> <p>Is the logical machine identifier for network connection and is in the range from one to sixty-one characters.</p> <p><i>LMID2</i></p> <p>Is the destination logical machine identifier for network connection and is in the range from one to sixty-one characters.</p>

### tuxTBridgeState

Syntax	INTEGER {active(1), inactive(2), suspended(3), pending(4)}
Access	read-write
Description	<p>The values for GET and SET operations are as follows:</p> <p>GET: {active(1)   inactive(2)   suspended(3)   pending(4)}</p> <p>A GET operation retrieves runtime information for the selected tuxTBridgeTbl instance(s). A tuxTBridgeLmid attribute value with only one logical machine identifier matches all active connections from <i>LMID1</i> to other machines in the application. In this case, each retrieved record contains an expanded tuxTBridgeLmid attribute 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 will not be returned.</p> <p>active(1)</p> <p>The connection is established and active.</p> <p>inactive(2)</p> <p>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 attribute and the source logical machine is reachable.</p>

suspended(3)

An established connection was terminated due to an error condition, and reconnection has been suspended for at least the amount of time indicated in the `tuxTBridgeSuspTime` attribute 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. This state is only supported on TUXEDO 6.4 or later.

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

A SET operation updates runtime information for the selected `tuxTBridgeTb1` object. The following states indicate the meaning of a `tuxTBridgeState` set in a SET request. States not listed cannot be set.

active(1)

**TUXEDO 6.3 and earlier:** Activate the `tuxTBridgeTb1` object by establishing a connection between the indicated logical machines. This operation fails if only one logical machine is specified, if either of the two machines is not active, or if the source logical machine is not reachable. State change allowed in the `inactive(2)` and `suspended(3)` states. Successful return leaves the object in the `active(1)` state. **TUXEDO 6.4 and later:** Activate the `tuxTBridgeTb1` 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 may 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 `tuxTBridgeTb1` 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 may 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. This state is supported only on 6.4 and later versions of TUXEDO systems.

### tuxTBridgeCurTime

Syntax	INTEGER
Access	read-only
Description	Current time, in seconds, since 00:00:00 UTC, January 1, 1970, as returned by the <code>time(2)</code> system call on <code>tuxTBridgeLmid</code> . This attribute can be used to compute elapsed time from the following attribute 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 <code>tuxTBridgeState</code> of <code>inactive(2)</code> and may 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. This is negotiated between the machines when the link is established. The number specifies the encryption key length (in bits). This object is only supported on TUXEDO 6.4 and later.

### **tuxTBridgeNetworkGrpNo**

Syntax	Integer
Access	read-only
Description	Logical network group number. When both the source and destination <code>tuxTBridgeLmid</code> machine identifiers are in the same network group, <code>tuxTBridgeTbl</code> will present all instances of related fields per network group. This object is supported only on TUXEDO 6.4 and later.



**tuxTBridgeNetworkGrpName**

Syntax	DisplayString
Access	read-only
Description	Logical network group name. This object is only supported on TUXEDO 6.4 or later.

# tuxTclientTbl

This group represents runtime attributes of active clients within an application. These attribute 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.

Variable Name	Object ID
tuxTclientState	.1.3.6.1.4.1.140.300.17.1.1.1
tuxTclientBirthTime	.1.3.6.1.4.1.140.300.17.1.1.2
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

Variable Name	Object ID
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
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

### 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 will retrieve runtime 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 will not be returned.

active(1)

tuxTclientTbl instance active. This is not an indication of whether the client is idle or busy. A non-0 value retrieved for either the tuxTclientCurConv attribute or the tuxTclientCurReq attribute 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 will exist 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 will update runtime information for the selected `tuxTclientTbl` object. The following states indicate the meaning of a `tuxTclientState` set in a SET request. States not listed may not 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 will be permitted to make these calls until they abort or commit the current transaction, at which time they will become suspended. Invocations of these routines will result in a TPESYSTEM error return and a system log message being generated indicating the situation. State change allowed only when in the `active(1)` state. Successful return leaves the object in the `suspended(2)` state.

`dead(3)`

Abortively deactivate the `tuxTclientTbl` instance. State change allowed only when 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)`) may not be set to a state of `dead(3)`. The system may not be able to kill the client due to platform or signaling restrictions. In this case, a native client will be abortively terminated at its next access to ATMI, and a workstation client's connection to a WSH will be 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.

### tuxTclientMachinelId

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.

### tuxTclientClntName

Syntax	<i>DisplayString</i> (SIZE(0..30))
Access	read-only
Description	Client name associated with client at <code>tpinit(3)</code> time via the <code>clntname</code> element of the <code>TPINIT</code> structure.

## tuxTclientIdleTime

Syntax	INTEGER
Access	read-only
Description	Approximate amount of time, in seconds, since this client last interacted with the system via an ATMI call. This value is accurate to within <code>tuxTdomainScanUnit</code> (see the <code>tuxTdomain</code> group) seconds. When specified as a key field, a positive value indicates that all clients with idle times of 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 may be specified on a <code>GET</code> operation for the purpose of retrieving client information for the calling process. If the calling process is not a client, then an error will be returned.

## tuxTclientSrvGrp

Syntax	<i>DisplayString</i> (SIZE(0..30))
Access	read-only
Description	Server group with which the client is associated. This information is set via the <code>grpname</code> element of the <code>TPINIT</code> structure at <code>tpinit(3)</code> time.

## tuxTclientUserName

Syntax	<i>DisplayString</i> (SIZE(0..30))
Access	read-only
Description	User name associated with client at <code>tpinit(3)</code> time via the <code>username</code> element of the <code>TPINIT</code> structure.

### tuxTclientWsc

Syntax	INTEGER { yes(1), no(2) }
Access	read-only
Description	If this attribute is set to yes(1), then 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 attribute is set to yes(1), then the indicated client is a workstation handler process.

### tuxTclientWshClientId

Syntax	<i>DisplayString</i> (SIZE(1..78))
Access	read-only
Description	Client identifier for the associated workstation handler (WSH) if this client is a workstation client ( <code>tuxTclientWsc == yes(1)</code> ); otherwise, this attribute will be returned as a 0-length string.

### tuxTclientRelease

Syntax	INTEGER
Access	read-only
Description	The TUXEDO System/T major protocol release number for the machine where the client is running. This may be different from the <code>tuxTmachineSWrelease</code> for the same machine. Note that for /WS clients ( <code>tuxTclientWsc == yes(1)</code> ), this value may be different than 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/T /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 attribute when associated with non-/WS clients ( <code>tuxTclientWsc == no(2)</code> ).

## tuxTclientNumConv

Syntax	INTEGER
Access	read-only
Description	Number of conversations initiated by this client via <code>tpconnect(3)</code> .

## tuxTclientNumDeque

Syntax	INTEGER
Access	read-only
Description	Number of dequeue operations initiated by this client via <code>tpdequeue(3)</code> .

## tuxTclientNumEnque

Syntax	INTEGER
Access	read-only
Description	Number of enqueue operations initiated by this client via <code>tpenqueue(3)</code> .

## tuxTclientNumPost

Syntax	INTEGER
Access	read-only
Description	Number of postings initiated by this client via <code>tppost(3)</code> .

### **tuxTclientNumReq**

Syntax	INTEGER
Access	read-only
Description	Number of requests made by this client via <code>tpcall(3)</code> or <code>tpacall(3)</code> .

### **tuxTclientNumSubscribe**

Syntax	INTEGER
Access	read-only
Description	Number of subscriptions made by this client via <code>tpsubscribe(3)</code> .

### **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 System/T ATMI function <code>tpscmt(3)</code> for details on this characteristic.

## tuxTclientCurConv

Syntax	INTEGER
Access	read-only
Description	Number of conversations initiated by this client via <code>tpconnect(3)</code> that are still active.

## tuxTclientCurReq

Syntax	INTEGER
Access	read-only
Description	Number of requests initiated by this client via <code>tpcall(3)</code> or <code>tpacall(3)</code> 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 <code>time(2)</code> system call on the local host. This attribute can be used to compute elapsed time from the <code>tuxTclientTimeStart</code> attribute 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	<i>DisplayString</i> (SIZE(1..78))
Access	read-only
Description	<p>For workstation clients, this attribute indicates the network address of the client. Network addresses with unprintable characters will be converted to the “0x...” network address format as described in the <code>tuxTmachineNaddr</code> attribute. Non-workstation clients will have a 0-length string associated with them for this attribute value.</p> <p><b>Note:</b> The ability of the system to provide this information is determined by the transport provider in use. In some cases, workstation clients may not have addresses associated with them if the provider does not make this information available.</p>

### tuxTclientNotify

Syntax	INTEGER { dipin(1), signal(2), ignore(3) }
Access	read-only
Description	Setting of the notification characteristic for this client. See the <code>tuxTdomain</code> group description of this attribute for more details.

### tuxTclientNumUnSol

Syntax	INTEGER
Access	read-only
Description	Number of unsolicited messages queued for this client awaiting processing.

## tuxTclientRpid

Syntax     INTEGER

Access     read-only

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

**Note:**     This is a UNIX system specific attribute that may 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 will timeout. This timeout may 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	<i>DisplayString</i> (SIZE(1..78))
Access	read-only
Description	Client Identifier.

# tuxTconnTable

This represents runtime attributes 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 attributes i.e., values for these objects correspond to the local host only, where the TUXEDO agent is running. So 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`.

Variable Name	Object ID
tuxTconnSerNo	.1.3.6.1.4.1.140.300.18.1.1.1
tuxTconnState	.1.3.6.1.4.1.140.300.18.1.1.2
tuxTconnSvcName	.1.3.6.1.4.1.140.300.18.1.1.3
tuxTconnClientId	.1.3.6.1.4.1.140.300.18.1.1.4
tuxTconnOgrpNo	.1.3.6.1.4.1.140.300.18.1.1.5
tuxTconnOlmid	.1.3.6.1.4.1.140.300.18.1.1.6
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	<p>The values for GET and SET operations are as follows:</p> <p>GET: active(1)</p> <p>A GET operation will retrieve runtime 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 will not be returned.</p> <p>active(1)</p> <p>The object returned reflects one or both sides of an active conversation within the application.</p> <p>SET:</p> <p>SET operations are not permitted on this class.</p>

### tuxTconnSvcName

Syntax	<i>DisplayString</i> (SIZE(1..15))
Access	read-only
Description	Service name of the conversational service invoked by the originator and processed by the subordinate.



---

## tuxTconnClientId

Syntax	<i>DisplayString</i> (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 attribute.

## tuxTconnOImid

Syntax	<i>DisplayString</i> (SIZE(1..30))
Access	read-only
Description	Logical machine identifier indicating where the originator is running or is accessing the application (in the case of /WS clients).

## tuxTconnOpid

Syntax	INTEGER
Access	read-only
Description	Process identifier for the originator of the conversation.

## tuxTconnOsndcnt

Syntax	INTEGER
Access	read-only
Description	Number of <code>tpsend(3)</code> calls done 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 attribute.

### **tuxTconnSlmid**

Syntax	<i>DisplayString</i> (SIZE(1..30))
Access	read-only
Description	Logical machine identifier indicating where the subordinate is running or is accessing the application (in the case of /WS clients).

### **tuxTconnSpid**

Syntax	INTEGER
Access	read-only
Description	Process identifier for the subordinate in the conversation.

### **tuxTconnSsndcnt**

Syntax	INTEGER
Access	read-only
Description	Number of <code>tpsend(3)</code> calls done by the subordinate.

**tuxTconnSrvId**

Syntax	INTEGER (1..30001)
Access	read-only
Description	Server identifier for the subordinate in the conversation.

# tuxTdeviceTbl

This represents configuration and runtime attributes of raw disk slices or UNIX system files being used to store TUXEDO System/T device lists. This class 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`.

Variable Name	Object ID
tuxTdevLmid	.1.3.6.1.4.1.140.300.19.1.1.1
tuxTdevCfgDev	.1.3.6.1.4.1.140.300.19.1.1.2
tuxTdeviceName	.1.3.6.1.4.1.140.300.19.1.1.3
tuxTdevOffset	.1.3.6.1.4.1.140.300.19.1.1.4
tuxTdevSize	.1.3.6.1.4.1.140.300.19.1.1.5
tuxTdevIndex	.1.3.6.1.4.1.140.300.19.1.1.6
tuxTdevState	.1.3.6.1.4.1.140.300.19.1.1.7

## tuxTdevLmid

Syntax	<i>DisplayString</i> (SIZE(1..30))
Access	read-write
Description	Logical machine identifier where the device is located. Note that this attribute may be used as a key field in both unbooted and booted applications as long as they are already configured (that is, at least one <code>tuxTmachineTable</code> instance exists). It is required as a key field on SET operations when accessing a booted application. If specified when accessing the <code>tuxTdeviceTbl</code> table in an unconfigured application, this attribute 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/T 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/T 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 attribute may 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 <code>tuxTdevice</code> within the device list addressed by <code>tuxTdevCfgDev</code> . This attribute value is used for identification purposes only in getting and setting attribute values relating to particular devices within a TUXEDO System/T filesystem.

### tuxTdevState

Syntax	INTEGER { valid(1), invalid(2), re-init(3) }
Access	read-write
Description	<p>The values for GET and SET operations are as follows:</p> <p>GET: {valid(1)}</p> <p>A GET operation will retrieve runtime information for the selected <code>tuxTdeviceTbl</code> instance(s). The following state indicates the meaning of a <code>tuxTdevState</code> returned in response to a GET request. States not listed will not be returned.</p> <p>valid(1)</p> <p>The TUXEDO System/T filesystem indicated by <code>tuxTdevCfgDev</code> exists and contains a valid device list. <code>tuxTdevice</code> is a valid device within that filesystem with the device index <code>tuxTdevIndex</code>.</p> <p>SET: {invalid(2) re-init(3)}</p> <p>A SET operation will update information for the selected <code>tuxTdeviceTbl</code> instance or add the indicated object. The following states indicate the meaning of a <code>tuxTdevState</code> set in a SET request. States not listed may not be set.</p> <p>invalid(2)</p> <p>Delete <code>tuxTdeviceTbl</code> instance for application. State change allowed only when in the <code>valid(1)</code> state. Successful return leaves the object in the <code>invalid(2)</code> state. Note that <code>tuxTdevIndex 0</code> is special and must be deleted last.</p> <p>re-init(3)</p> <p>To re-initialize a valid device.</p>

# tuxTwhichCfgDev

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

Variable Name	Object ID
tuxTwhichCfgDev	.1.3.6.1.4.1.140.300.19.2

## tuxTwhichCfgDev

- Syntax

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

read-write
- Description

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

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

# tuxTdomain

The following objects of `tuxTdomain` represent global application attributes for the domain to which the TUXEDO SNMP agent is currently connected. These object values serve to identify, customize, size, secure, and tune a TUXEDO System/T 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.

Variable Name	Object ID
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



Variable Name	Object ID
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
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

Variable Name	Object ID
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
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
m3MaxObjects	.1.3.6.1.4.1.140.300.3.63
m3MaxInterfaces	.1.3.6.1.4.1.140.300.3.68
m3CurInterfaces	.1.3.6.1.4.1.140.300.3.73
m3HwInterfaces	.1.3.6.1.4.1.140.300.3.78

tuxTdomainKey

Syntax    INTEGER ( 32769..262143)  
Access    read-write

**Description**     Numeric key for the well-known address in a TUXEDO System/T 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) may set only the master logical machine identifier. Modifications to this attribute 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* attribute 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 may 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 will verify that the two sites connecting share a common view of where the master machine is. Otherwise, the connection will be refused and an appropriate log message generated.

The master and backup machines in an active application must always have a TUXEDO System/T 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` attribute must preserve this relationship.

### tuxTdomainModel

Syntax	INTEGER { single-machine(1), multi-machine(2) }
Access	read-write
Description	Configuration type. <code>single-machine(1)</code> specifies a single machine configuration; only one <code>tuxTmachineTable</code> object may be specified. <code>multi-machine(2)</code> specifies a multi-machine or network configuration; it must be specified if a networked application is being defined.

### tuxTdomainState

Syntax	INTEGER { active(1), inactive(2), forcible-inactive(3) }
Access	read-write
Description	<p>The values for GET and SET operations are as follows:</p> <p>GET: {active(1) inactive(2)}</p> <p>A GET operation will retrieve configuration and runtime information for the <code>tuxTdomain</code> group. The following states indicate the meaning of a <code>tuxTdomainState</code> returned in response to a GET request. States not listed will not be returned.</p> <p><code>active(1)</code>  <code>tuxTdomain</code> group defined and the master machine is active.</p> <p><code>inactive(2)</code>  <code>tuxTdomain</code> group defined and application is inactive.</p> <p>SET: active(1) inactive(2) forcible-inactive(3)</p> <p>A SET operation will update configuration and run-time information for the <code>tuxTdomain</code> group. The following states indicate the meaning of a <code>tuxTdomainState</code> set in a SET request. States not listed may not be set.</p>

`active(1)`

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

`inactive(2)`

Deactivate administrative processes (DBBL, BBL, etc.) on the master machine. A state change is allowed only when 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, etc.) on the master machine. Attached clients will be ignored for the purpose of determining if shutdown should be allowed. State change is allowed only when in the `active(1)` state. Successful return leaves the object in the `inactive(2)` state. You will 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 attribute setting for newly configured objects in the `tuxTmachineTable` group.

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

### tuxTdomainGID

Syntax `INTEGER`

Access `read-write`

Description Default attribute setting for newly configured objects in the `tuxTmachineTable` group.

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

### tuxTdomainPerm

Syntax `DisplayString(SIZE(1..9))`

Access `read-write`

Description Default attribute setting for newly configured objects in the `tuxTmachineTable` group.

**Note:** Changes to this attribute 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 this attribute value will no longer be allowed for all class/attribute 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 attribute setting for newly configured objects in the <code>tuxTmachineTable</code> group.
<b>Note:</b>	Changes to this attribute do not affect active or already configured <code>tuxTmachineTable</code> instances.

**tuxTdomainMaxConv**

Syntax	INTEGER (0..32767)
Access	read-write
Description	Default attribute setting for newly configured objects in the <code>tuxTmachineTable</code> group.
<b>Note:</b>	Changes to this attribute do not affect active or already configured <code>tuxTmachineTable</code> instances.

**tuxTdomainMaxGTT**

Syntax	INTEGER (0..32767)
Access	read-write
Description	Default attribute setting for newly configured objects in the <code>tuxTmachineTable</code> group.
<b>Note:</b>	Changes to this attribute do not affect active or already configured <code>tuxTmachineTable</code> 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 <code>tuxTroutingTable</code> group object is required. Additional entries should be allocated to allow for runtime 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.  <b>Note:</b> TUXEDO System/T Release 4.2.2 and earlier sites have a fixed setting of 100 for this attribute. Interoperability with these sites requires that no more than 100 server group entries be in use at any time. Release 4.2.2 and earlier sites will not be allowed to join an application that has more than 100 defined server groups. Additionally, applications already including Release 4.2.2 or earlier sites will not be allowed to add server groups beyond 100.

### tuxTdomainMaxMachines

Syntax	INTEGER ( 256 . . 8190 )
Access	read-write



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

**Note:**     TUXEDO System/T Release 4.2.2 has a fixed setting of 256 for this attribute. Releases prior to Release 4.2.2 have a fixed setting of 50 for this attribute. Interoperability with Release 4.2.2 and earlier sites requires that no more than the lowest fixed setting number of machine table entries be in use at any time. Release 4.2.2 sites will not be allowed to join an application that has more than 256 defined machines. Pre-Release 4.2.2 sites will not be allowed to join an application that has more than 50 defined machines. Additionally, applications already including active Release 4.2.2 or earlier sites will not be allowed to add machines beyond the lowest applicable limit.

## **tuxTdomainMaxQueues**

**Syntax**     INTEGER (1..8191)

**Access**     read-write

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

**Note:**     Release 4.2.2 and earlier sites may join an active application only if the setting for `tuxTdomainMaxQueues` is equal to the setting for `tuxTdomainMaxServers`.

## **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` class object. Additional entries should be allocated to allow for runtime 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 arrays specified within `tuxTroutingRanges` values are stored in the string pool. Additional space should be allocated to allow for runtime 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 attribute for system supplied administrative servers. Administration of each System/T site adds approximately one server. Additionally, if TMSs are specified for any server groups (see `tuxTgroupTMSname`), then they will be 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 attribute for system supplied servers offering services for administrative purposes. Administration of each System/T site adds approximately five services. Other administrative components such as /WS, /Q, and /DM may 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 ACL permissions checking. 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 System/T 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 System/T ATMI function <code>tpscmt(3)</code> for details on the setting of this characteristic.

**Note:** Runtime modifications to this attribute do not affect active clients and servers.

## tuxTdomainLoadBalance

Syntax	INTEGER { yes(1), no(2) }
Access	read-write
Description	Load balancing is/will be on yes(1) or off no(2).

**Note:** Runtime modifications to this attribute do not affect active clients and servers.

## tuxTdomainNotify

Syntax	INTEGER { dipin(1), signal(2), ignore(3) }
Access	read-write
Description	Default notification detection method to be 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 <code>tpinit(3)</code> 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 via the <code>tpsetunsol(3)</code> function.

The value `dipin(1)` specifies that dip-in-based notification detection should be used. This means that the system will only detect 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 will not interrupt blocking system calls. `dipin(1)` is the default notification detection method.

The value `signal(2)` specifies that signal-based notification detection should be used. This 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 selecting this method of notification.

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

**Note:** Runtime modifications to this attribute 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 using the `signal(2)` method.

## tuxTdomainSystemAccess

Syntax	INTEGER { <code>fastpath(1)</code> , <code>protected(2)</code> , <code>fastpath-no-override(3)</code> , <code>protected-no-override(4)</code> }
Access	read-write
Description	Default mode used by System/T libraries within application processes to gain access to System/T's internal tables. <code>fastpath(1)</code> specifies that System/T's internal tables are accessible by System/T libraries via unprotected shared memory for fast access. <code>protected(2)</code> specifies that System/T's internal tables are accessible by System/T libraries via protected shared memory for safety against corruption by application code. <code>fastpath-no-override(3)</code> or <code>protected-no-override(4)</code> can be specified to indicate that the mode selected cannot be overridden by an application process using flags available for use with <code>tpinit(3)</code> .
<b>Note:</b>	Updates to this attribute value in a running application affect only newly started clients and newly configured <code>tuxTsrvrTbl</code> 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).  <b>Note:</b> Only the accstats(3) may 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

*app\_password* 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.

A string value NONE for this attribute indicates that security is (or will be) turned off. The value APP\_PW/*app\_password* indicates that application password security is to be enforced. Clients must provide the application password during initialization. The value USER\_AUTH is similar to APP\_PW, but indicates also that per-user authentication is done during client initialization. The value ACL is similar to USER\_AUTH, but also indicates that access control checks will be 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. The value MANDATORY\_ACL is similar to ACL, but permission is denied if an associated ACL is not found for the name.

### tuxTdomainAuthsvc

Syntax	<i>DisplayString</i> (SIZE (1..15))
Access	read-write
Description	Application authentication service invoked by the system for each client joining the system. This attribute is ignored if the tuxTdomainSecurity attribute 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 <code>tuxTdomainBBLQuery</code> , <code>tuxTdomainBlockTime</code> , <code>tuxTdomainDBBLWait</code> , and <code>tuxTdomainSanityScan</code> objects are multipliers of this value. Passing a value of 0 for this attribute on a SET operation will cause the attribute to be reset to its default value.

## tuxTdomainBBLQuery

Syntax	INTEGER (0..32767)
Access	read-write
Description	Multiplier of the <code>tuxTdomainScanUnit</code> object indicating time between DBBL status checks on registered BBLs. The DBBL checks to ensure that all BBLs have reported in within the <code>tuxTdomainBBLQuery</code> 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 attribute on a SET operation will cause the attribute to be reset to its default value. This attribute value should be set to at least twice the value set for <code>tuxTdomainSanityScan</code> .

## tuxTdomainBlockTime

Syntax	INTEGER (0..32767)
Access	read-write
Description	Multiplier of the <code>tuxTdomainScanUnit</code> object indicating the minimum amount of time a blocking ATMI call will block before timing out. Passing a value of 0 for this attribute on a SET operation will cause the attribute to be reset to its default value.

### **tuxTdomainDBBLWait**

Syntax	INTEGER (0..32767)
Access	read-write
Description	Multiplier of the <code>tuxTdomainScanUnit</code> object indicating maximum amount of time a DBBL should wait for replies from its BBLs before timing out. Passing a value of 0 for this attribute on a SET operation will cause the attribute to be reset to its default value.

### **tuxTdomainSanityScan**

Syntax	INTEGER (0..32767)
Access	read-write
Description	Multiplier of the <code>tuxTdomainScanUnit</code> object indicating time 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 ( <code>multi-machine(2)</code> mode only). Passing a value of 0 for this attribute on a SET operation will cause the attribute to be reset to its default value.

### **tuxTdomainCurDRT**

Syntax	INTEGER (0..32767)
Access	read-only
Description	Current number of in use bulletin board routing table entries.

### **tuxTdomainCurGroups**

Syntax	INTEGER (0..32767)
Access	read-only
Description	Current number of in use bulletin board server group table entries.



## **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 in use bulletin board queue table entries.

## **tuxTdomainCurRFT**

Syntax	INTEGER (0..32767)
Access	read-only
Description	Current number of in use bulletin board routing criteria range table entries.

## **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 in use bulletin board server table entries.

### **tuxTdomainCurServices**

Syntax	INTEGER (0..32767)
Access	read-only
Description	Current number of in use bulletin board service table entries.

### **tuxTdomainCursType**

Syntax	INTEGER (0..32767)
Access	read-only
Description	Current number of in use bulletin board subtype table entries.

### **tuxTdomainCurType**

Syntax	INTEGER (0..32767)
Access	read-only
Description	Current number of in use bulletin board type table entries.

### **tuxTdomainHwDRT**

Syntax	INTEGER (0..32767)
Access	read-only
Description	High water number of in use bulletin board routing table entries.

### **tuxTdomainHwGroups**

Syntax	INTEGER (0..32767)
Access	read-only
Description	High water number of in use bulletin board server group table entries.

## **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 in use bulletin board queue table entries.

## **tuxTdomainHwRFT**

Syntax	INTEGER (0..32767)
Access	read-only
Description	High water number of in use bulletin board routing criteria range table entries.

## **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 in use bulletin board server table entries.

### tuxTdomainHwServices

Syntax	INTEGER ( 0 . . 32767 )
Access	read-only
Description	High water number of in use bulletin board service table entries.

### tuxTdomainMaxNetGroups

Syntax	INTEGER ( 1 . . 8191 )
Access	read-write
Description	The maximum number of groups that can be configured. This object is only supported on TUXEDO 6.4 or later.

### m3MaxObjects

Syntax	INTEGER
Access	read-write
Description	The default maximum number of active objects to be accommodated in the Active Object Map tables in the M3 bulletin board.

**Note:** This object is supported for M3 applications only.

### m3MaxInterfaces

Syntax	INTEGER ( 1 . . 32765 )
Access	read-write
Description	Specifies the maximum number of interfaces to 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 m3MaxInterfaces.

**Note:** This object is supported for M3 applications only.

### m3CurInterfaces

Syntax	INTEGER
Access	read-only
Description	The current number of interface entries used in the bulletin board interface tables.
<b>Note:</b> This object is supported for M3 applications only.	

### m3HwInterfaces

Syntax	INTEGER
Access	read-only
Description	The high water mark for the number of interface entries used in the bulletin board interface tables.
<b>Note:</b> This object is supported for M3 applications only.	

# tuxTgroupTable

The `tuxTgroupTable` group represents application attributes pertaining to a particular server group. These attribute 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`.

Variable 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

## 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` class and `tuxTgroupLMID` values in the `tuxTmachineTable` class. 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.
<b>Note:</b>	This object can be set only during row creation.

## tuxTgroupLMID

Syntax	<i>DisplayString</i> (SIZE (1..61))
Access	read-write
Description	<i>DisplayString</i> is in the format: <i>LMID1</i> [ , <i>LMID2</i> ]  <i>LMID1</i> Is the primary machine logical machine identifier for this server group and is in the range from one to sixty-one characters.  <i>LMID2</i> 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 <code>tuxTdomainOptions</code> attribute). A single LMID specified on a GET operation will match either the primary or secondary LMID. Note that the location of an active group is available in the <code>tuxTgroupCurLMID</code> object. Logical machine identifiers specified with the <code>tuxTgroupLMID</code> object must be already configured.  <b>Note:</b> Modifications to this attribute for an active object may 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	<p>The values for GET and SET operations are as follows:</p> <p>GET: {active(1) inactive(2) migrating(3)}</p> <p>A GET operation will retrieve configuration and runtime 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 will not be returned.</p> <p>active(1)</p> <p>tuxTgroupTable object defined and active (TMS and/or application servers). Server groups with non-0 length values for the tuxTgroupTMSname attribute 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.</p> <p>inactive(2)</p> <p>tuxTgroupTable object defined and inactive.</p> <p>migrating(3)</p> <p>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.</p> <p>SET: {active(1) inactive(2) migrating(3) invalid(4) re-active(5) suspend-services(6) resume-services(7)}</p> <p>A SET operation will update 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 may not be set.</p> <p>active(1)</p> <p>Activate the tuxTgroupTable object. State change allowed only when in the inactive(2) or migrating(3) state. If the group is currently in the inactive(2) state, then TMS and application servers are started on the primary logical machine if the primary logical machine is active; otherwise, the TMS and application servers are started on the secondary logical machine if it is active. If neither machine is active, then the request fails. If the group is</p>



currently in the `migrating(3)` state, then the active secondary logical machine (identified as the alternate to `tuxTgroupCurLMID` in the `tuxTgroupLMID` list) is used to start TMS and application servers if it is active. 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 allowed only when 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 allowed only when in the `active(1)` state. Successful return leaves the object in the `migrating(3)` state.

`invalid(4)`

Delete `tuxTgroupTable` object for application. State change allowed only when 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 in addition to being allowed in 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. This state is only available in TUXEDO 6.4 and later.

`resume-services(7)`

Unsuspend and resume all application services in the group that are marked suspended. This operation is allowed only when the group is in the `active(1)` state. The operation leaves the group in the `active(1)` state. Note that this operation fails in an application environment that includes any machine where TUXEDO 6.3 or earlier applications are active.

### tuxTgroupCurLMID

Syntax	<i>DisplayString</i> (SIZE (1..30))
Access	read-only
Description	Current logical machine on which the server group is running. This attribute will not be returned for server groups that are not active.

### tuxTgroupCloseInfo

Syntax	<i>DisplayString</i> (SIZE (0..256))
Access	read-write
Description	<p>If a non-0 length value other than TMS is specified for the <code>tuxTgroupTMSname</code> object, then this object value indicates the resource manager dependent information needed when terminating access to the resource manager. Otherwise, this attribute value is ignored.</p> <p>The format for this object value is dependent on the requirements of the vendor providing the underlying resource manager. The information required by the vendor must be prefixed with <code>rm_name:</code>, which is the published name of the vendor's transaction (XA) interface followed immediately by a colon (:).</p> <p>A 0-length string value for this attribute means that the resource manager for this group (if specified) does not require any application specific information to close access to the resource.</p> <p><b>Note:</b> Runtime modifications to this attribute will not affect active servers in the group.</p>

### tuxTgroupOpenInfo

Syntax	<i>DisplayString</i> (SIZE (0..256))
Access	read-write
Description	<p>If a non-0 length value other than TMS is specified for the <code>tuxTgroupTMSname</code> object, then this object value indicates the resource manager dependent information needed when initiating access to the resource manager. Otherwise, this object value is ignored.</p>

The format for this object 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 attribute means that the resource manager for this group (if specified) does not require any application specific information to open access to the resource.

**Note:** Runtime modifications to this attribute will 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 <code>tuxTgroupTMSname</code> object, then this object value indicates the number of transaction manager servers to start for the associated group. Otherwise, this object value is ignored.

## tuxTgroupTMSname

Syntax	<i>DisplayString</i> (SIZE (0..78))
Access	read-write
Description	Transaction manager server <code>a.out</code> associated with this group. This parameter must be specified for any group entry whose servers will participate in distributed transactions (transactions across multiple resource managers and possibly machines that are started with <code>tpbegin(3)</code> and ended with <code>tpcommit(3)/tpabort(3)</code> ).

The value `TMS` is reserved to indicate use of the null XA interface. If a non-empty value other than `TMS` is specified, then 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 will be restartable an unlimited number of times.

# tuxTmachineTable

The `tuxTmachineTable` group represents application attributes pertaining to a particular machine. These attribute 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, a SET request should be issued 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.

Variable 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

Variable Name	Object ID
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
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
m3MaxMachineObjects	.1.3.6.1.4.1.140.300.5.1.1.35

## tuxTmachinePmid

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

Access read-write

Description Physical machine identifier. This identifier should match the UNIX system nodename as 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..78))  
Access read-write  
Description Absolute pathname of the file or device where the binary TUXEDO System/T configuration file is found on this machine. The administrator need only maintain one such file, namely the one identified by the `tuxTmachineTuxConfig` attribute 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 this attribute value is used in the environment.

### tuxTmachineTuxDir

Syntax *DisplayString* (SIZE (2..78))  
Access read-write  
Description Absolute pathname of the directory where the TUXEDO System/T software is found on this machine. See `tuxTmachineEnvFile` below for a discussion of how this attribute value is used in the environment.

## tuxTmachineAppDir

Syntax	<code>DisplayString (SIZE (2..78))</code>
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 starting application servers. See <code>tuxTmachineEnvFile</code> for a discussion of how this attribute value is used in the environment.

## tuxTmachineState

Syntax	<code>INTEGER { active(1), inactive(2), partitioned(3), invalid(4), re-activate(5), cleaning(7) }</code>
Access	read-write
Description	<p>The values for GET and SET operations are as follows:</p> <p>GET: {active(1) inactive(2) partitioned(3)}</p> <p>A GET operation will retrieve configuration and runtime information for the selected <code>tuxTmachineTable</code> instance(s). The following states indicate the meaning of a <code>tuxTmachineState</code> returned in response to a GET request. States not listed will not be returned</p> <p>active(1)     <code>tuxTmachineTable</code> instance defined and active (administrative servers, that is, DBBL, BBL, and BRIDGE).</p> <p>inactive(2)     <code>tuxTmachineTable</code> instance defined and inactive.</p> <p>partitioned(3)     <code>tuxTmachineTable</code> instance defined, listed in accessible bulletin boards as active, but currently unreachable.</p> <p>SET: {active(1) inactive(2) invalid(4) re-activate(5) cleaning(7)}</p> <p>A SET operation will update configuration and run-time information for the selected <code>tuxTmachineTable</code> instance. The following states indicate the meaning of a <code>tuxTmachineState</code> set in a SET request. States not listed may not be set.</p>

### `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 allowed only when 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 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 allowed only when 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 allowed only when 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 will be detected at this time. If the machine has been partitioned from the application master site, then global bulletin board entries for that machine will be 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)` allowed only for non-master machines. The master site administrative processes are deactivated via the `tuxTdomain` class.



## tuxTmachineUid

Syntax	INTEGER
Access	read-write
Description	UNIX system user identifier for the TUXEDO System/T application administrator on this machine. Administrative commands such as <code>tmboot(1)</code> , <code>tmshutdown(1)</code> , and <code>tmadmin(1)</code> must run as the indicated user on this machine. Application and administrative servers on this machine will be started as this user.

**Note:** This is a UNIX system-specific attribute that may not 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 System/T application administrator on this machine. Administrative commands such as <code>tmboot(1)</code> , <code>tmshutdown(1)</code> , and <code>tmadmin(1)</code> must run as part of the indicated group on this machine. Application and administrative servers on this machine will be started as part of this group.

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

## tuxTmachineEnvFile

Syntax	<i>DisplayString</i> (SIZE (2..78))
Access	read-write
Description	Environment file for clients and servers running on this machine.

### tuxTmachinePerm

Syntax	<i>DisplayString</i> (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.
<b>Note:</b>	Modifications to this attribute for an active object will not affect running servers or clients.
<b>Note:</b>	This is a UNIX system-specific attribute that may not be returned if the platform on which the application is being run is not UNIX-based.

### tuxTmachineUlogPfx

Syntax	<i>DisplayString</i> (SIZE (0..78))
Access	read-write
Description	Absolute pathname prefix of the path for the <code>userlog(3)</code> file on this machine. The <code>userlog(3)</code> file name is formed by appending the string <code>.mmddyy</code> to the <code>tuxTmachineUlogPfx</code> attribute value. <code>.mmddyy</code> represents the month, day, and year that the messages were generated. All application and system <code>userlog(3)</code> messages generated by clients and servers running on this machine are directed to this file.
<b>Note:</b>	Modifications to this attribute for an active object will not affect running servers or clients.

## tuxTmachineType

Syntax	<i>DisplayString</i> (SIZE (1..15))
Access	read-write
Description	Machine type. Used to group machines into classes of like data representations. Data encoding is not performed when communicating between machines of identical types. This attribute can be given any string value; values are used only for comparison. Distinct <code>tuxTmachineType</code> attributes should be set when the application spans a heterogeneous network of machines or when compilers generate dissimilar structure representations. The default value for this attribute, a 0-length string, matches any other machine with a 0-length string as its <code>tuxTmachineType</code> attribute 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 <code>tmadmin</code> 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 may 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 <code>tuxTmachineMaxAccessers</code> attribute. The appropriate setting of this parameter helps to conserve IPC resources since workstation client access to the system is multiplexed through a System/T supplied surrogate, the workstation handler. It is an error to set this number greater than <code>tuxTmachineMaxAccessers</code> .

### **tuxTmachineMaxAclCache**

Syntax	INTEGER ( 10 .. 32000 )
Access	read-write
Description	Number of entries in the cache used for ACL entries when <code>tuxTdomainSecurity</code> is set to <code>acl(4)</code> or <code>mandatory-acl(5)</code> . 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	<i>DisplayString</i> (SIZE (0..64))
Access	read-write
Description	The device (raw slice) or UNIX system file containing the TUXEDO System/T filesystem that holds the DTP transaction log for this machine. The DTP transaction log is stored as a TUXEDO System/T VTOC table on the device. This device or file may be the same as that specified for the <code>tuxTmachineTuxConfig</code> attribute for this machine.

## tuxTmachineTlogName

Syntax	<i>DisplayString</i> (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 <code>tuxTmachineTlogDevice</code> , they must have unique names. <code>tuxTmachineTlogName</code> must be different from the name of any other table on the <code>tuxTmachineTlogDevice</code> 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 <code>tuxTmachineTlogSize</code> attribute value is subject to limits based on available space in the TUXEDO System/T filesystem identified by the <code>tuxTmachineTlogDevice</code> attribute.

### tuxTmachineBridge

Syntax	<i>DisplayString</i> (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. This is a required value for participation in a networked application via a TLI-based TUXEDO System/T binary. This attribute is not needed for sockets-based TUXEDO System/T binaries.

### tuxTmachineNaddr

Syntax	<i>DisplayString</i> (SIZE (0..78))
Access	read-write
Description	<p>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 attribute must be set if the logical machine is to participate in a networked application, that is, if the LAN option is set in the <code>tuxTdomainOptions</code> attribute value.</p> <p>If string has the form <code>0xhex-digits</code> or <code>\\xhex-digits</code>, 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.</p>

### tuxTmachineNlsaddr

Syntax	<i>DisplayString</i> (SIZE (0..78))
Access	read-write
Description	<p>Network address used by the <code>tlisten(1)</code> process servicing the network on the node identified by this logical machine. This network address is of the same format as that specified for the <code>tuxTmachineNaddr</code> attribute.</p> <p>This attribute must be set if the logical machine is to participate in a networked application, that is, if the LAN option is set in the <code>tuxTdomainOptions</code> attribute value.</p>

---

## tuxTmachineCmpLimit

Syntax	<i>DisplayString</i>
Access	read-write
Description	Threshold message size at which compression will occur for remote traffic and optionally local traffic. Remote and local may be either non-negative numeric values or the string MAXLONG, which is dynamically translated to the maximum long setting for the machine. Setting only the remote value will default local to MAXLONG.

**Note:** This attribute value is not part of the `tuxTmachineTable` object for active sites running TUXEDO System/T Release 4.2.2 or earlier. However, site release identification is not determined until runtime, so this attribute may be set and accessed for any inactive object. When a TUXEDO System/T Release 4.2.2 or earlier site is activated, the configured value is not used.

## tuxTmachineTmNetLoad

Syntax	INTEGER (0..32767)
Access	read-write
Description	Service load added to any remote service evaluated during load balancing on this machine.

**Note:** This attribute value is not part of the `tuxTmachineTable` object for active sites running TUXEDO System/T Release 4.2.2 or earlier. However, site release identification is not determined until runtime, so this attribute may be set and accessed for any inactive object. When a TUXEDO System/T Release 4.2.2 or earlier site is activated, the configured value is not used.

### tuxTmachineSpinCount

Syntax	INTEGER
Access	read-write
Description	<p>Spincount used on this machine for pre-ticket user level semaphore access. Default values are built into the TUXEDO System/T binaries on each machine. These defaults may be overridden at runtime for tuning purposes using this attribute. The spincount may be reset to the default built-in value for the site by resetting this attribute value to 0.</p> <p><b>Note:</b> This attribute value is not part of the <code>tuxTmachineTable</code> object for active sites running TUXEDO System/T Release 4.2.2 or earlier. However, site release identification is not determined until runtime, so this attribute may be set and accessed for any inactive object. When a TUXEDO System/T Release 4.2.2 or earlier site is activated, the configured value is not used.</p>

### tuxTmachineRole

Syntax	INTEGER { master(1), backup(2), other(3) }
Access	read-only
Description	<p>The role of this machine in the application. <code>master(1)</code> indicates that this machine is the master machine, <code>backup(2)</code> indicates that it is the backup master machine, and <code>other(3)</code> indicates that the machine is neither the master nor backup master machine.</p>

### tuxTmachineMinor

Syntax	INTEGER
Access	read-only
Description	<p>The TUXEDO System/T minor protocol release number for this machine.</p>



---

## tuxTmachineRelease

Syntax	INTEGER
Access	read-only
Description	The TUXEDO System/T major protocol release number for this machine. This may 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 for this is 100000. This object is supported on TUXEDO 6.4 and later only.

## m3MachineMaxObjects

Syntax	INTEGER
Access	read-write
Description	The maximum number of objects to be accommodated in the Active Object Map tables in the bulletin board.  <b>Note:</b> This object is supported for M3 applications only.

# tuxTmachineActive

This group represents run-time statistics on the local machine if it is active (i.e., 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.

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

Variable Name	Object ID
tuxTmachineNumTranAbt	.1.3.6.1.4.1.140.300.5.2.19
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
m3MachineCurObjects	.1.3.6.1.4.1.140.300.5.2.36
m3MachineHwObjects	.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 currently accessing 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` attribute is `no(2)` or the `tuxTdomainModel` attribute is `multi-machine(2)`, then an FML32 NULL value is returned (0).

### 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 <code>tpconnect(3)</code> operations performed from this machine.

### **tuxTmachineNumDequeue**

Syntax	INTEGER
Access	read-only
Description	Number of <code>tpdequeue(3)</code> operations performed from this machine.

### **tuxTmachineNumEnqueue**

Syntax	INTEGER
Access	read-only
Description	Number of <code>tpenqueue(3)</code> operations performed from this machine.

### **tuxTmachineNumPost**

Syntax	INTEGER
Access	read-only
Description	Number of <code>tppost(3)</code> operations performed from this machine.

### **tuxTmachineNumReq**

Syntax	INTEGER
Access	read-only
Description	Number of <code>tpacall(3)</code> or <code>tpcall(3)</code> operations performed from this machine.

## tuxTmachineNumSubscribe

Syntax	INTEGER
Access	read-only
Description	Number of <code>tpsubscribe(3)</code> operations performed from this machine.

## tuxTmachineNumTran

Syntax	INTEGER
Access	read-only
Description	Number of transactions initiated ( <code>tpbegin(3)</code> ) from this machine.

## tuxTmachineNumTranAbt

Syntax	INTEGER
Access	read-only
Description	Number of transactions aborted ( <code>tpabort(3)</code> ) from this machine.

## tuxTmachineNumTranCmt

Syntax	INTEGER
Access	read-only
Description	Number of transactions committed ( <code>tpcommit(3)</code> ) from this machine.

## tuxTmachineLicExpires

Syntax	<i>DisplayString</i> (SIZE(0..78))
Access	read-only
Description	Expiration date for the binary on that machine or a 0-length string if binary is not a TUXEDO System/T master binary.

### **tuxTmachineLicMaxUsers**

Syntax	INTEGER (0..32767)
Access	read-only
Description	Licensed maximum number of users on that machine or -1 if binary is not a TUXEDO System/T master binary.

### **tuxTmachineLicSerial**

Syntax	<i>DisplayString</i> (SIZE(0..78))
Access	read-only
Description	Serial number for binary on that machine or a 0-length string if binary is not a TUXEDO System/T master binary.

### **tuxTmachinePageSize**

Syntax	INTEGER
Access	read-only
Description	Disk pagesize used on this machine.

### **tuxTmachineSWrelease**

Syntax	<i>DisplayString</i> (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/T 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.

## **tuxTmachineAclFail**

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 attribute may 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 attribute may wraparound, that is, exceed the maximum value for a long, and start back at 0 again.

### m3MachineCurObjects

Syntax	INTEGER
Access	read-only
Description	The number of entries in use in the bulletin board object table for this machine.

**Note:** This object is supported for M3 applications only.

### m3MachineHwObjects

Syntax	INTEGER
Access	read-only
Description	The high water mark of entries used in the bulletin board object table for this machine.

**Note:** This object is supported for M3 applications only.

# tuxTmsgTable

The `tuxTmsgTable` class represents runtime attributes of the TUXEDO System/T 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.

Variable Name	Object ID
tuxTmsgId	.1.3.6.1.4.1.140.300.6.1.1.1
tuxTmsgState	.1.3.6.1.4.1.140.300.6.1.1.2
tuxTmsgCurTime	.1.3.6.1.4.1.140.300.6.1.1.3
tuxTmsgCbytes	.1.3.6.1.4.1.140.300.6.1.1.4
tuxTmsgCtime	.1.3.6.1.4.1.140.300.6.1.1.5
tuxTmsgLrPid	.1.3.6.1.4.1.140.300.6.1.1.6
tuxTmsgLsPid	.1.3.6.1.4.1.140.300.6.1.1.7
tuxTmsgQbytes	.1.3.6.1.4.1.140.300.6.1.1.8
tuxTmsgQnum	.1.3.6.1.4.1.140.300.6.1.1.9
tuxTmsgRtime	.1.3.6.1.4.1.140.300.6.1.1.10
tuxTmsgStime	.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 is a UNIX system-specific attribute that may not 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	<p>The values for GET and SET operations are as follows:</p> <p>GET: active(1)</p> <p>A GET operation will retrieve runtime 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 will not be returned.</p> <p>active(1)</p> <p>tuxTmsgTable object active. This corresponds exactly to the related tuxTmachineTable object being active.</p> <p>SET:</p> <p>SET operations are not permitted on this class.</p>

### 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 <code>msgctl(2)</code> operation that changed a member of the <code>msgqid_ds</code> 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 represents runtime attributes of queues in an application. These attribute values identify and characterize allocated TUXEDO System/T 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

Variable 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	<code>DisplayString (SIZE(1..30))</code>
Access	read-only
Description	Symbolic address of the request queue. Servers with the same <code>tuxTsrvrRqAddr</code> attribute value are grouped into a Multiple Server Single Queue (MSSQ) set. Attribute values returned with a <code>tuxTqueueTable</code> 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 will retrieve runtime information for the selected `tuxTqueueTable` instance(s). The `tuxTqueueTable` group does not address configuration information directly. Configuration related attributes 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 will not be 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 will update runtime 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.



## tuxTqueueRqId

Syntax	INTEGER
Access	read-only
Description	UNIX system message queue identifier.

**Note:** This is a UNIX system specific attribute that may not 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	<p>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.</p> <p><b>Note:</b> If the <code>tuxTdomainLoadBalance</code> attribute is <code>no(2)</code> or the <code>tuxTdomainModel</code> attribute is <code>multi-machine(2)</code>, then <code>tuxTqueueTotNqueued</code> is not returned. In the same configuration, updates to this attribute are ignored. Consequently, when this attribute is returned <code>tuxTqueueSource</code> has the same value as the local host.</p>

### 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.
<b>Note:</b>	If the <code>tuxTdomainLoadBalance</code> attribute is <code>no(2)</code> or the <code>tuxTdomainModel</code> attribute is <code>multi-machine(2)</code> , then <code>tuxTqueueTotWkQueued</code> is not returned. In the same configuration, updates to this attribute are ignored. Consequently, when this attribute is returned <code>tuxTqueueSource</code> has the same value as the local host.

### tuxTqueueSource

Syntax	<i>DisplayString</i> (SIZE(1..30))
Access	read-only
Description	Logical machine from which local attribute values are retrieved.

### tuxTqueueNqueued

Syntax	INTEGER
Access	read-only
Description	Number of requests currently enqueued to this queue from the <code>tuxTqueueSource</code> logical machine. This value is incremented at enqueue time and decremented when the server dequeues the request.
<b>Note:</b>	If the <code>tuxTdomainLoadBalance</code> attribute is <code>no(2)</code> or the <code>tuxTdomainModel</code> attribute is <code>multi-machine(2)</code> , then <code>tuxTqueueNqueued</code> is not returned. Consequently, when this attribute is returned <code>tuxTqueueSource</code> has the same value as the local host.

**tuxTqueueWkQueued**

Syntax	INTEGER
Access	read-only
Description	Workload currently enqueued to this queue from the <code>tuxTqueueSource</code> logical machine. If the <code>tuxTdomainModel</code> attribute is set to <code>single-machine(1)</code> and the <code>tuxTdomainLoadBalance</code> attribute is set to <code>yes(1)</code> , then this attribute reflects the application-wide workload enqueued to this queue. However, if <code>tuxTdomainModel</code> is set to <code>multi-machine(2)</code> and <code>tuxTdomainLoadBalance</code> is set to <code>yes(1)</code> , this attribute reflects the workload enqueued to this queue from the <code>tuxTqueueSource</code> logical machine during a recent timespan. This attribute is used for load balancing purposes. So as to not discriminate against newly started servers, this attribute value is zeroed out on each machine periodically by the BBL.

# tuxTroutingTable

The `tuxTroutingTable` group represents configuration objects of routing specifications for an application. These 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 M3 applications. `m3RoutingFieldType` is valid only for factory-based routing. `tuxTroutingBufType` is valid only for service-based routing.

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

`m3RoutingFieldType` is valid only for factory-based routing. This is supported only for M3 applications.

`tuxTroutingBufType` is valid only for service-based routing (either TUXEDO or M3 applications).

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 `beaRoutingType`.

Variable Name	Object ID
<code>tuxTroutingName</code>	<code>.1.3.6.1.4.1.140.300.8.1.1.1</code>
<code>tuxTroutingBufType</code>	<code>.1.3.6.1.4.1.140.300.8.1.1.2</code>
<code>tuxTroutingField</code>	<code>.1.3.6.1.4.1.140.300.8.1.1.3</code>
<code>tuxTroutingRanges</code>	<code>.1.3.6.1.4.1.140.300.8.1.1.4</code>
<code>tuxTroutingState</code>	<code>.1.3.6.1.4.1.140.300.8.1.1.5</code>
<code>beaRoutingType</code>	<code>.1.3.6.1.4.1.140.300.8.1.1.6</code>
<code>m3RoutingFieldType</code>	<code>.1.3.6.1.4.1.140.300.8.1.1.7</code>
<code>tuxInternalIdx</code>	<code>.1.3.6.1.4.1.140.300.8.1.1.8</code>

## tuxTroutingName

Syntax	<i>DisplayString</i> (SIZE(1..15))
Access	read-write
Description	Routing criterion name.

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

## tuxTroutingBufType

Syntax	<i>DisplayString</i> (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 be 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.

## tuxTroutingField

Syntax	<i>DisplayString</i> (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 M3 documentation for more details.

### tuxTroutingRanges

Syntax	<i>DisplayString</i> (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.

**Note:** Attribute values greater than 256 bytes in length will disable interoperability with TUXEDO System/T Release 4.2.2 and earlier.

## tuxTroutingState

Syntax	INTEGER { valid(1), unknown(2), invalid(3) }
Access	read-write
Description	<p>The values for GET and SET operations are as follows:</p> <p>GET: valid(1)</p> <p>A GET operation will retrieve configuration information for the selected <code>tuxTroutingTable</code> instance(s). The following state indicates the meaning of a <code>tuxTroutingState</code> returned in response to a GET request. States not listed will not be returned.</p> <p>valid(1)</p> <p><code>tuxTroutingTable</code> instance is defined. Note that this is the only valid state for this class. Routing criteria are never active; rather, they are associated through the configuration with service names and are acted upon at runtime to provide data dependent routing.</p> <p>SET: invalid(3)</p> <p>A SET operation will update configuration information for the selected <code>tuxTroutingTable</code> instance. The following state indicates the meaning of a <code>tuxTroutingState</code> set in a SET request. States not listed may not be set.</p> <p>invalid(3)</p> <p>Delete <code>tuxTroutingTable</code> instance for application. State change allowed only when in the <code>valid(1)</code> state. Successful return leaves the object in the <code>invalid(2)</code> state.</p>

## beaRoutingType

Syntax	INTEGER { service(1), factory(2) }
Access	read-write
Description	<p><code>factory(2)</code> specifies that the routing criterion applies to factory-based routing for a CORBA interface. <code>service(1)</code> specifies that routing criteria apply to data-dependent routing for a BEA TUXEDO service.</p> <p><b>Note:</b> The routing type affects the validity and possible values for other attributes defined for this table.</p> <p><b>Note:</b> This object can be set during row creation only.</p>

### m3RoutingFieldType

Syntax	INTEGER { short(1), long(2), float(3), double(4), char(5), string(6) }
Access	read-write
Description	This object specifies the type of <code>tuxTroutingField</code> on which this routing criterion is defined. This is valid only for factory-based routing.

**Note:** This object is supported only for M3.

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

### tuxInternalIdx

Syntax	<i>DisplayString</i> (SIZE(1..30))
Access	read-write
Description	This object is used as an index of this table instead of <code>tuxTroutingBufType</code> (for service-based routing) or <code>tuxTroutingField</code> (for factory-based routing) to reduce the size of the index. Its value, for service based routing ( <code>beaRoutingType = service(1)</code> ) is equal to the first 30 characters in <code>tuxTroutingBufType</code> .

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

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



# tuxTsrvrTbl

This group represents configuration and runtime attributes of servers within an application. These attribute values identify and characterize configured servers as well as provide runtime tracking of statistics and resources associated with each server object. The index into this table is provided by the attributes `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`.

Variable Name	Object ID
<code>tuxTsrvrGrp</code>	<code>.1.3.6.1.4.1.140.300.20.1.1.1</code>
<code>tuxTsrvrId</code>	<code>.1.3.6.1.4.1.140.300.20.1.1.2</code>
<code>tuxTsrvrName</code>	<code>.1.3.6.1.4.1.140.300.20.1.1.3</code>
<code>tuxTsrvrGrpNo</code>	<code>.1.3.6.1.4.1.140.300.20.1.1.4</code>
<code>tuxTsrvrState</code>	<code>.1.3.6.1.4.1.140.300.20.1.1.5</code>
<code>tuxTsrvrBaseSrvId</code>	<code>.1.3.6.1.4.1.140.300.20.1.1.6</code>
<code>tuxTsrvrClOpt</code>	<code>.1.3.6.1.4.1.140.300.20.1.1.7</code>
<code>tuxTsrvrEnvFile</code>	<code>.1.3.6.1.4.1.140.300.20.1.1.8</code>
<code>tuxTsrvrGrace</code>	<code>.1.3.6.1.4.1.140.300.20.1.1.9</code>
<code>tuxTsrvrMaxgen</code>	<code>.1.3.6.1.4.1.140.300.20.1.1.10</code>
<code>tuxTsrvrMax</code>	<code>.1.3.6.1.4.1.140.300.20.1.1.11</code>
<code>tuxTsrvrMin</code>	<code>.1.3.6.1.4.1.140.300.20.1.1.12</code>
<code>tuxTsrvrRcmd</code>	<code>.1.3.6.1.4.1.140.300.20.1.1.13</code>
<code>tuxTsrvrRestart</code>	<code>.1.3.6.1.4.1.140.300.20.1.1.14</code>
<code>tuxTsrvrSequence</code>	<code>.1.3.6.1.4.1.140.300.20.1.1.15</code>
<code>tuxTsrvrSystemAccess</code>	<code>.1.3.6.1.4.1.140.300.20.1.1.16</code>

Variable Name	Object ID
tuxTsrvrConv	.1.3.6.1.4.1.140.300.20.1.1.17
tuxTsrvrReplyQ	.1.3.6.1.4.1.140.300.20.1.1.18
tuxTsrvrRpPerm	.1.3.6.1.4.1.140.300.20.1.1.19
tuxTsrvrRqAddr	.1.3.6.1.4.1.140.300.20.1.1.20
tuxTsrvrRqPerm	.1.3.6.1.4.1.140.300.20.1.1.21
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

**tuxTsrvrGrp**

Syntax	<i>DisplayString</i> (SIZE(1..30))
Access	read-write
Description	Logical name of the server group. Server group names cannot contain an asterisk (*), comma, or colon.
<b>Note:</b> 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.
<b>Note:</b> This object can be set only during row creation.	

## tuxTsrvrName

Syntax	<i>DisplayString</i> (SIZE(1..78))
Access	read-write
Description	Name of the server executable file. The server identified by <code>tuxTsrvrName</code> will run on the machine(s) identified by the <code>tuxTgroupLMID</code> object for this server's server group. If a relative pathname is given, then the search for the executable file is done first in <code>tuxTmachineAppDir</code> , then in <code>tuxTmachineTuxDir/bin</code> , then in <code>/bin</code> and <code>/usr/bin</code> , and then in <code>&lt;path&gt;</code> , where <code>&lt;path&gt;</code> is the value of the first <code>PATH=</code> line appearing in the machine environment file, if one exists. Note that the attribute value returned for an active server will always be 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	<p>The values for GET and SET operations are as follows:</p> <p>GET: <code>active(1)   inactive(2)   migrating(3)   cleaning(4)   restarting(5)   suspended(6)   partitioned(7)   dead(8)</code></p> <p>A GET operation will retrieve configuration and runtime information for the selected <code>tuxTsrvrTbl</code> instance(s). The following states indicate the meaning of a <code>tuxTsrvrState</code> returned in response to a GET request. States not listed will not be returned.</p> <p><code>active(1)</code></p> <p><code>tuxTsrvrTbl</code> instance defined and active. This is not an indication of whether the server is idle or busy. An active server with a non-0 length <code>tuxTsrvrCurService</code> attribute should be interpreted as a busy server, that is, one that is processing a service request.</p>

`inactive(2)`  
`tuxTsrvrTbl` instance defined and inactive.

`migrating(3)`  
`tuxTsrvrTbl` instance 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` attribute that does not match the `tuxTgroupCurLMID` object.

`cleaning(4)`  
`tuxTsrvrTbl` instance defined and currently being cleaned up after by the system due to an abnormal death. Note that restartable servers may enter this state if they exceed `tuxTsrvrMaxgen` starts/restarts within their `tuxTsrvrGrace` period.

`restarting(5)`  
`tuxTsrvrTbl` instance defined and currently being restarted by the system due to an abnormal death.

`suspended(6)`  
`tuxTsrvrTbl` instance defined and currently suspended pending shutdown.

`partitioned(7)`  
`tuxTsrvrTbl` instance defined and active; however, the machine where the server is running is currently partitioned from the `tuxTdomainMaster` site.

`dead(8)`  
`tuxTsrvrTbl` instance defined, identified as active in the bulletin board, but currently not running due to an abnormal death. This state will exist 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 will update 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 may not be set.

`active(1)`  
 Activate the `tuxTsrvrTbl` instance. State change allowed only when 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 `tuxTsrvrTb1` instance. State change allowed only when in the `active(1)` state. Successful return leaves the object in the `inactive(2)` state.

`dead(8)`

Deactivate the `tuxTsrvrTb1` 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 will cause the server to initiate orderly shutdown and the server will become 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 may be used and will be treated by the system as an abnormal termination. State change allowed only when 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 `tuxTsrvrTb1` instance for application. State change allowed only when 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` attribute value of 1, this attribute will always be the same as `tuxTsrvrId`. However, for servers with a `tuxTsrvrMax` value of greater than 1, this attribute indicates the base server identifier for the set of servers configured identically.

## tuxTsrvrCLOpt

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:** Runtime modifications to this attribute will not affect a running server.

### tuxTsrvrEnvFile

Syntax	<i>DisplayString</i> (SIZE(0..78))
Access	read-write
Description	Server specific environment file. See <code>tuxTmachineEnvFile</code> for a complete discussion of how this file is used to modify the environment.

**Note:** Runtime modifications to this attribute will not affect a running server.

### tuxTsrvrGrace

Syntax	INTEGER
Access	read-write
Description	The period of time, in seconds, over which the <code>tuxTsrvrMaxgen</code> limit applies. This attribute is meaningful only for restartable servers, that is, if the <code>tuxTsrvrRestart</code> attribute is set to <code>yes(1)</code> . When a restarting server would exceed the <code>tuxTsrvrMaxgen</code> limit but the <code>tuxTsrvrGrace</code> period has expired, the system resets the current generation ( <code>tuxTsrvrGeneration</code> ) to 1 and resets the initial boot time ( <code>tuxTsrvrTimeStart</code> ) to the current time. A value of 0 for this attribute 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 attribute. If they do not, then the first server activated will establish the runtime value associated with all servers on the queue.

**Note:** Runtime modifications to this attribute will 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 attribute.

## 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 attribute. If they do not, then the first server activated will establish the runtime value associated with all servers on the queue.

**Note:** Runtime modifications to this attribute will 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 attribute.

## 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 may be started individually (by starting a particular server id) or through automatic spawning (conversational servers only). Runtime modifications to this attribute will 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 <code>tmboot(1)</code> . If a <code>tuxTsrvrRqAddr</code> is specified and <code>tuxTsrvrMin</code> is greater than 1, then the servers will form an MSSQ set. The server identifiers for the servers will be <code>tuxTsrvrId</code> up to <code>tuxTsrvrId + tuxTsrvrMax - 1</code> . All occurrences of the server will have the same sequence number, as well as any other server parameters.

**Note:** Runtime modifications to this attribute will not affect a running server.

### tuxTsrvrRcmd

Syntax	<i>DisplayString</i> (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.

Note that servers sharing a request queue (that is, equal values for `tuxTsrvrRqAddr`) should have equal values for this attribute. If they do not, then the first server activated will establish the runtime value associated with all servers on the queue.

**Note:** Runtime modifications to this attribute will 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 attribute.

### tuxTsrvrRestart

Syntax	INTEGER { yes(1), no(2) }
Access	read-write
Description	Restartable <code>yes(1)</code> or non-restartable <code>no(2)</code> server. If server migration is specified for this server group ( <code>tuxTdomainOptions = migrate(2)</code> and <code>tuxTgroupLMID</code> with alternate site), then this attribute must be set to <code>yes(1)</code> .



Note that servers sharing a request queue (that is, equal values for `tuxTsrvrRqAddr`) should have equal values for this attribute. If they do not, then the first server activated will establish the runtime value associated with all servers on the queue.

**Note:** Runtime modifications to this attribute will 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 attribute.

## 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` attribute specified or with an invalid value will 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. Runtime modifications to this attribute affect only `tmboot(1)` and `tmshutdown(1)` and will affect the order in which running servers may be shutdown by a subsequent invocation of `tmshutdown(1)`.

## tuxTsrvrSystemAccess

Syntax    `INTEGER { fastpath(1), protected(2) }`

Access    read-write

Description    Mode used by System/T libraries within this server process to gain access to System/T's internal tables. See `tuxTdomainSystemAccess` for a complete discussion of this attribute.

**Note:** Runtime modifications to this attribute will 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	Allocate a separate reply queue for the server (tuxTsrvrReplyQ == yes(1)). MSSQ servers that expect to receive replies should set this attribute to yes(1).

### tuxTsrvrRpPerm

Syntax	<i>DisplayString</i> (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)), then this attribute is ignored. This is a string representation of octal numbers starting with a leading 0 0001 through 0777.  <b>Note:</b> This is a UNIX system specific attribute that may not be returned if the platform on which the application is being run is not UNIX-based.

### tuxTsrvrRqAddr

Syntax	<i>DisplayString</i> (SIZE(0..30))
Access	read-write
Description	Symbolic address of the request queue for the server. Specifying the same tuxTsrvrRqAddr attribute value for more than one server is the way multiple server, single queue (MSSQ) sets are defined. Servers with the same tuxTsrvrRqAddr attribute 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. This is a string representation of octal numbers starting with a leading 0 0001 through 0777.

**Note:** This is a UNIX system specific attribute that may not 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 will be restarted with the generation reset to 1.

## tuxTsrvrPid

Syntax INTEGER

Access read-only

Description UNIX system process identifier for the server. Note that this may not be a unique attribute since servers may be located on different machines allowing for duplication of process identifiers.

**Note:** This is a UNIX system specific attribute that may not 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 ( <code>tuxTsrvrReplyQ == no(2)</code> ), then this attribute value will be the same as <code>tuxTsrvrRqId</code> .

**Note:** This is a UNIX system specific attribute that may not 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 ( <code>tuxTsrvrReplyQ == no(2)</code> ), then this attribute value will be the same as <code>tuxTsrvrRpid</code> .

**Note:** This is a UNIX system specific attribute that may not 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 <code>time(2)</code> 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 <code>time(2)</code> system call on local host, when the server was first started. Restarts of the server do not reset this value; however, if <code>tuxTsrvrMaxgen</code> is exceeded and <code>tuxTsrvrGrace</code> is expired, this attribute will be reset to the time of the restart.

# tuxTsrvrTblExt

An extension of the tuxTsrvrTbl. Objects in this table are only accessible through a TUXEDO SNMP agent installed on the local machine

Variable Name	Object ID
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

Variable Name	Object ID
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
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
m3SrvrCurObjsExt	.1.3.6.1.4.1.140.300.20.2.1.32
m3SrvrCurInterfaceExt	.1.3.6.1.4.1.140.300.20.2.1.37

## 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 via <code>tpconnect(3)</code> .

### **tuxTsrvrNumDeque**

Syntax	INTEGER
Access	read-only
Description	Number of dequeue operations initiated by this server via <code>tpdequeue(3)</code> .

### **tuxTsrvrNumEnque**

Syntax	INTEGER
Access	read-only
Description	Number of enqueue operations initiated by this server via <code>tpenqueue(3)</code> .

### **tuxTsrvrNumPost**

Syntax	INTEGER
Access	read-only
Description	Number of postings initiated by this server via <code>tppost(3)</code> .

### **tuxTsrvrNumReq**

Syntax	INTEGER
Access	read-only
Description	Number of requests made by this server via <code>tpcall(3)</code> or <code>tpacall(3)</code> .



## **tuxTsrvrNumSubscribe**

Syntax     `INTEGER`

Access     read-only

Description     Number of subscriptions made by this server via `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)`), this attribute value indicates the number of completed incoming conversations. This is a runtime attribute 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 ( <code>tuxTsrvrConv == yes(1)</code> ), this attribute value indicates the workload of completed incoming conversations. This is a runtime attribute that is kept across server restart but is lost at server shutdown.

### tuxTsrvrClgLmid

Syntax	<i>DisplayString</i> (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 that the server is currently working on.

### tuxTsrvrClgLpid

Syntax	INTEGER
Access	read-only
Description	UNIX system process identifier for the initiating client or server.  <b>Note:</b> This is a UNIX system specific attribute that may not be returned if the platform on which the application is being run is not UNIX-based.

### tuxTsrvrClgLreply

Syntax	INTEGER { <code>yes(1)</code> , <code>no(2)</code> , <code>null(3)</code> }
Access	read-only
Description	The initiating client or server is expecting a reply <code>yes(1)</code> or is not expecting a reply <code>no(2)</code> .

## tuxTsrvrCmtRet

Syntax	INTEGER { complete(1), logged(2) }
Access	read-only
Description	Setting of the TP_COMMIT_CONTROL characteristic for this server. See the description of the System/T ATMI function <code>tpscmt(3)</code> for details on this characteristic.

## tuxTsrvrCurConv

Syntax	INTEGER
Access	read-only
Description	Number of conversations initiated by this server via <code>tpconnect(3)</code> that are still active.

## tuxTsrvrCurReq

Syntax	INTEGER
Access	read-only
Description	Number of requests initiated by this server via <code>tpcall(3)</code> or <code>tpacall(3)</code> that are still active.

## tuxTsrvrCurService

Syntax	<i>DisplayString</i> (SIZE(1..15))
Access	read-only
Description	Service name that the server is currently working on, if any.

### **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 attribute can be used to compute elapsed time from the <code>tuxTsrvrTimeStart</code> and <code>tuxTsrvrTimeRestart</code> object values.

### **tuxTsrvrLastGrp**

Syntax	INTEGER (1..30000)
Access	read-only
Description	Server group number ( <code>tuxTgroupNo</code> ) of the last service request made or conversation initiated from this server outward.

### **tuxTsrvrSvcTimeOut**

Syntax	INTEGER
Access	read-only
Description	Time left, in seconds, for this server to process the current service request, if any. A value of 0 for an active service indicates that no timeout processing is being done. See <code>tuxTsvcTimeOut</code> 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 will timeout. This timeout may 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.

## **m3SrvrCurObjsExt**

Syntax	INTEGER
Access	read-only
Description	The number of entries in use in the bulletin board object table for this server.

## **m3SrvrCurInterfaceExt**

Syntax	DisplayString (SIZE(1..128))
Access	read-only
Description	The interface name of the interface currently active in this server.

# tuxTsvcTbl

This represents configuration attributes of services within an application. These attribute values identify and characterize configured services. A `tuxTsvcTbl` object provides activation time configuration attributes 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.

Variable Name	Object ID
tuxTsvcName	.1.3.6.1.4.1.140.300.10.1.1.1
tuxTsvcType	.1.3.6.1.4.1.140.300.10.1.1.2
tuxTsvcState	.1.3.6.1.4.1.140.300.10.1.1.3
tuxTsvcAutoTran	.1.3.6.1.4.1.140.300.10.1.1.4
tuxTsvcLoad	.1.3.6.1.4.1.140.300.10.1.1.5
tuxTsvcPrio	.1.3.6.1.4.1.140.300.10.1.1.6
tuxTsvcTimeOut	.1.3.6.1.4.1.140.300.10.1.1.7
tuxTsvcTranTime	.1.3.6.1.4.1.140.300.10.1.1.8
tuxTsvcBufType	.1.3.6.1.4.1.140.300.10.1.1.9
tuxTsvcRoutingName	.1.3.6.1.4.1.140.300.10.1.1.10

## tuxTsvcName

- Syntax

*DisplayString* (SIZE(1..15))
- 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. <code>app(1)</code> indicates an application defined service name. <code>callable(2)</code> indicates a system provided callable service. <code>system(3)</code> indicates a system provided and system callable service. <code>system(3)</code> 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	<p>The values for GET and SET operations are as follows:</p> <p>GET: {active(1) inactive(2)}</p> <p>A GET operation will retrieve configuration information for the selected <code>tuxTsvcTbl</code> instance(s). The following states indicate the meaning of a <code>tuxTsvcState</code> returned in response to a GET request. States not listed will not be returned.</p> <p>active(1)</p> <p>tuxTsvcTbl instance is defined and at least one <code>tuxTsvcGrp</code> object with a matching <code>tuxTsvcName</code> value is active.</p> <p>inactive(2)</p> <p>tuxTsvcTbl instance is defined and no <code>tuxTsvcGrp</code> object with a matching <code>tuxTsvcName</code> value is active.</p> <p>SET: invalid(3)</p> <p>A SET operation will update configuration information for the selected <code>tuxTsvcTbl</code> instance. The following states indicate the meaning of a <code>tuxTsvcState</code> set in a SET request. States not listed may not be set.</p> <p>invalid(3)</p> <p>Delete <code>tuxTsvcTbl</code> instance for application. State change allowed only when in the <code>inactive(2)</code> state. Successful return leaves the object in the <code>invalid(3)</code> state.</p>

### tuxTsvcAutoTran

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.
<b>Note:</b>	Runtime updates to this attribute 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).
<b>Note:</b>	Runtime updates to this attribute 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 will be serviced first.
<b>Note:</b>	Runtime updates to this attribute 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 will be abortively terminated (`kill -9`) if they exceed the specified time limit in processing the request. A value of 0 for this attribute indicates that the service should not be abortively terminated.

**Note:**     Runtime updates to this attribute are not reflected in active `tuxTsvcGrp` objects.

**Note:**     This attribute value is not enforced on TUXEDO System/T Release 4.2.2 sites or earlier.

## 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` attribute value for the service is `yes(1)`.

**Note:**     Runtime updates to this attribute are not reflected in active `tuxTsvcGrp` objects.

## tuxTsvcBufType

Syntax	<i>DisplayString</i> (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/T 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 will accept 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:** This attribute value represents the buffer types that must be supported by each and every instance of an application service with this service name. Since this attribute value is processed at service activation time, updates to this attribute are allowed only when there are no active *tuxTsvcGrp* objects with matching service names.

## tuxTsvcRoutingName

Syntax	<i>DisplayString</i> (SIZE(0..15))
Access	read-write
Description	This <i>tuxTsvcTbl</i> object has the indicated routing criteria name. Active updates to this attribute will be reflected in all associated <i>tuxTsvcGrp</i> objects.

# tuxTsvcGrp

The `tuxTsvcGrp` group represents configuration and runtime attributes of services/groups within an application. These attribute values identify and characterize configured services/groups as well as provide runtime tracking of statistics and resources associated with each object.

Both `tuxTsvcTbl` and `tuxTsvcGrp` define activation time attribute settings 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 attribute values to be used at service startup time.

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

Variable Name	Object ID
tuxTsvcGrpSvcName	.1.3.6.1.4.1.140.300.10.2.1.1
tuxTsvcGrpName	.1.3.6.1.4.1.140.300.10.2.1.2
tuxTsvcGrpNo	.1.3.6.1.4.1.140.300.10.2.1.3
tuxTsvcGrpState	.1.3.6.1.4.1.140.300.10.2.1.4
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	<i>DisplayString</i> (SIZE(1..15))
Access	read-only
Description	Service name.

## tuxTsvcGrpName

Syntax	<i>DisplayString</i> (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 will retrieve 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 will not be returned.

active(1)

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

inactive(2)

tuxTsvcGrp instance defined and inactive.

SET: invalid(3)

It removes the corresponding tuxTsvcGrp instance. When a tuxTsvcGrp instance is deleted it will also remove the associated tuxTsvcSrvr instances which correspond to server instances which are a part of this 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 will be 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 will be abortively terminated (kill -9) if they exceed the specified time limit in processing the request. A value of 0 for this attribute 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 <code>tuxTsvcGrp</code> instance. Transactions are started automatically when a request not in transaction mode is received and the <code>tuxTsvcGrpAutoTran</code> attribute value for the service is <code>yes(1)</code> .

## tuxTsvcSrvrLmid

Syntax	<i>DisplayString</i> (SIZE(1..30))
Access	read-only
Description	Current logical machine on which an active server offering this service is running.

## tuxTsvcSrvrRqAddr

Syntax	<i>DisplayString</i> (SIZE(1..30))
Access	read-only
Description	Symbolic address of the request queue for an active server offering this service. See <code>tuxTsrvrRqAddr</code> for more information on this attribute.

## tuxTsvcSrvrId

Syntax	INTEGER (1..30000)
Access	read-write
Description	Server ID of which the service is a part of. The user can also set the value of this object to activate (advertise) one or more <code>tuxTsvcGrp</code> instances. The value provided to set this object is used to activate another instance of <code>tuxTsvcGrp</code> .

### tuxTsvcrName

Syntax	<i>DisplayString</i> (SIZE(1..15))
Access	read-write
Description	Function name within the associated server assigned to process requests for this service. When this object is specified, the <code>tuxTsvcGrp</code> instance is activated (advertised). The user needs to specify the server ID of the corresponding server instance ( <code>tuxTsvcSrvrId</code> ) 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).
<b>Note:</b>	This attribute is returned only when <code>tuxTdomainLoadBalance</code> is equal to <code>yes(1)</code> .

### tuxTsvcSrvrNqueued

Syntax	INTEGER (0..32767)
Access	read-only
Description	Number of requests currently enqueued to this service. This attribute is incremented at enqueue time and decremented when the server dequeues the request.
<b>Note:</b>	This attribute is returned only when the <code>tuxTdomainModel</code> is set to <code>single-machine(1)</code> and the <code>tuxTdomainLoadBalance</code> attribute is set to <code>yes(1)</code> .



# tuxTlistenTbl

This group represents runtime attributes of /T listener processes for a distributed application.

Variable Name	Object ID
tuxTlistenLmid	.1.3.6.1.4.1.140.300.21.1.1.1
tuxTlistenState	.1.3.6.1.4.1.140.300.21.1.1.2

## 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 will retrieve runtime 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 will not be returned.

active(1)  
tuxTlistenTbl instance active.

inactive(2)  
tuxTlistenTbl instance not active.

# tuxTranTbl

This table represents runtime attributes 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.

Variable 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
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	<i>DisplayString</i> (SIZE(1..30))
Access	read-only
Description	Logical machine identifier of the server group responsible for coordinating the transaction.

## tuxTpTranId

Syntax	<i>DisplayString</i> (SIZE(2..78))
Access	read-only
Description	Transaction identifier as returned from <code>tpsuspend(3)</code> mapped to a string representation. The data in this field should not be interpreted directly by the user except for equality comparison.

## tuxTranXid

Syntax	<i>DisplayString</i> (SIZE(2..78))
Access	read-only
Description	Transaction identifier as returned from <code>tx_info(3)</code> mapped to a string representation. The data in this field should not be interpreted directly by the user except for equality comparison.

## tuxTranIndx1

Syntax	INTEGER
Access	read-only
Description	This number is purely for unique indexing of this table.

**tuxTranIndx2**

Syntax	INTEGER
Access	read-only
Description	This number is purely for unique indexing of this table.

**tuxTranIndx3**

Syntax	INTEGER
Access	read-only
Description	This number is purely for unique indexing of this table.

**tuxTranIndx4**

Syntax	INTEGER
Access	read-only
Description	This number is purely for unique indexing of this table.

**tuxTranIndx5**

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 will retrieve runtime information for the selected tuxTranTbl instance(s). The following states indicate the meaning of a tuxTranState object. States not listed will not be returned. Note that distinct objects pertaining to the same global transaction (equivalent transaction identifiers) may 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 will eventually be propagated to the coordinator site and result in the rollback of the transaction, but this change may not 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 will be returned from the initiator's site only.

SET: aborted(3)

A SET operation will update runtime information for the selected `tuxTranTbl` instance. The following state indicates the meaning of a `tuxTranState` set in a SET request. States not listed may not be set.

aborted(3)

Abort the `tuxTranTbl` instance for the application. State change allowed only when in the `active(1)`, `abort-only(2)`, or `com-called(4)` states. May not 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 will timeout on the retrieval site. Note that this attribute 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 attribute 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 will retrieve runtime 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 will not be returned. Note that distinct objects pertaining to the same global transaction (equivalent transaction identifiers) may 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 will be propagated to the group's site and result in the rollback of the group's work in the transaction but may not 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 may or may not agree with the final resolution of the transaction.

`habort(6)`

The group has been heuristically rolled back. This may or may 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 M3 servers that have called `xa_end` (TMSUSPEND) during the course of transactional work and that commit processing is beginning. This state will exist until either (1) All servers that called `xa_end` (TMSUSPEND) have caused a call to `xa_end` (TMSUCCESS), at which point the group state will become ready, or (2) One of the target servers does a rollback of the transaction at which point the group state will become either `post-abort(9)` or `aborted(2)`.

**Note:** This state is supported for M3 applications only.

`post-abort(9)`

Indicates that an M3 server called `xa_end` (TPFAIL) and that the TMS has not yet called `xa_rollback()`. In this case, other M3 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 for M3 applications only.

`post-commit(10)`

This state is not implemented yet.

**Note:** This state is supported for M3 applications only.



SET: `hcommit(5) | habort(6)`

A SET operation will update runtime 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 may not 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 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 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 represents runtime attributes of userlog 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.

Variable Name	Object ID
tuxTulogSerNo	.1.3.6.1.4.1.140.300.9.1.1.1
tuxTulogLmid	.1.3.6.1.4.1.140.300.9.1.1.2
tuxTulogPmid	.1.3.6.1.4.1.140.300.9.1.1.3
tuxTulogMmDdYy	.1.3.6.1.4.1.140.300.9.1.1.4
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

## tuxTulogSerNo

Syntax	INTEGER
Access	read-only
Description	A running serial number for the rows in tuxTulogTable.

## tuxTulogLmid

Syntax	<i>DisplayString</i> (SIZE(1..30))
Access	read-only
Description	Retrieval machine logical machine identifier.

## tuxTulogPmid

Syntax	<i>DisplayString</i> (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	<i>DisplayString</i> (SIZE(1..256))
Access	read-only
Description	The entire text of the userlog message as it appears in the userlog file.

### tuxTulogTpTranId

Syntax	<i>DisplayString</i> (SIZE(1..78))
Access	read-only
Description	Transaction identifier as returned from <code>tpsuspend(3)</code> . The data in this field should not be interpreted directly by the user except for equality comparison. Messages not associated with transactions will retrieve a 0-length string as the value for this attribute.

### tuxTulogXid

Syntax	<i>DisplayString</i> (SIZE(1..78))
Access	read-only
Description	Transaction identifier as returned from <code>tx_info(3)</code> . The data in this field should not be interpreted directly by the user except for equality comparison. Messages not associated with transactions will retrieve a 0-length string as the value for this attribute.

### tuxTulogPid

Syntax	INTEGER
Access	read-only
Description	Process identifier of the client or server that generated the userlog message.

## tuxTulogSeverity

Syntax	<i>DisplayString</i> (SIZE(1..30))
Access	read-only
Description	Severity of message, if any.

## tuxTulogCat

Syntax	<i>DisplayString</i> (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	<i>DisplayString</i> (SIZE(1..30))
Access	read-only
Description	Process name of the client or server that generated the userlog message.

# tuxTulogCtrl

The values of objects in this group control the ulog messages returned by the tuxTulogTable.

Variable Name	Object ID
tuxTulogLmidCtrl	.1.3.6.1.4.1.140.300.9.2.1
tuxTulogPmidCtrl	.1.3.6.1.4.1.140.300.9.2.2
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	<i>DisplayString</i> (SIZE(1..30))
Access	read-write
Description	Logical machine ID to qualify machine from where the userlog file is read for <code>tuxTulogTable</code> . By default, the ULOG files from the local host are returned, per the ULOGPFX. To revert to the default setting, set this object to <code>null</code> .

## tuxTulogPmidCtrl

Syntax	<i>DisplayString</i> (SIZE(1..30))
Access	read-write
Description	Physical machine name to qualify the source machine for userlog messages to be listed in <code>tuxTulogTable</code> . By default, messages from all hosts within ULOG files qualified by <code>tuxTulogLmidCtrl</code> are returned. To revert to the default setting, set this object to <code>null</code> .

## tuxTulogMmddyyCtrl

Syntax	INTEGER
Access	read-write
Description	Date value to qualify userlog messages to be listed in <code>tuxTulogTable</code> . 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 for the time range for which the userlog messages are to be listed in <code>tuxTulogTable</code> . 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 for the time range for which the userlog messages are to be listed in <code>tuxTulogTable</code> . 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 to be listed in <code>tuxTulogTable</code> . By default, all messages are returned. To revert to the default setting, set this object to 0.

### tuxTulogMsgCtrl

Syntax	<i>DisplayString</i> (SIZE(1..30))
Access	read-write
Description	Regular expression to qualify userlog messages to be listed in <code>tuxTulogTable</code> 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	<i>DisplayString</i> (SIZE(1..78))
Access	read-write
Description	Value of <code>tuxTpTranId</code> to qualify messages to be displayed in the in <code>tuxTulogTable</code> . By default, all messages are returned. To revert to the default setting, set it to null.



## tuxTulogXidCtrl

Syntax	<i>DisplayString</i> (SIZE(1..30))
Access	read-write
Description	Value of <code>tuxTranXid</code> to qualify messages to be displayed in the in <code>tuxTulogTable</code> . By default, all messages are returned. To revert to the default setting, set it to <code>null</code> .

## tuxTulogPidCtrl

Syntax	INTEGER
Access	read-write
Description	Value of process Id of the source to qualify messages to be displayed in the <code>tuxTulogTable</code> . By default, messages with any pid are listed. To revert to the default setting, set this object to 0.

## tuxTulogSeverityCtrl

Syntax	<i>DisplayString</i> (SIZE(1..30))
Access	read-write
Description	Regular expression to qualify userlog messages to be listed in <code>tuxTulogTable</code> 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 <code>null</code> .

## tuxTulogCatCtrl

Syntax	<i>DisplayString</i> (SIZE(1..30))
Access	read-write
Description	Regular expression to qualify userlog messages to be listed in <code>tuxTulogTable</code> 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 <code>null</code> .

### **tuxTulogMsgNumCtrl**

Syntax	INTEGER
Access	read-write
Description	Message number in catalog to qualify userlog messages to be listed in <code>tuxTulogTable</code> . By default, all message numbers are returned. To revert to the default setting, set this object to 0.

### **tuxTulogProcNameCtrl**

Syntax	<i>DisplayString</i> (SIZE(1..30))
Access	read-write
Description	Regular expression to qualify userlog messages to be listed in <code>tuxTulogTable</code> 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 <code>null</code> .

# tuxTnetGrpTbl

This table represents application attributes 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.

**Note:** This table is supported only on TUXEDO 6.4 or later.

Variable Name	Object ID
<code>tuxTnetGrpName</code>	<code>.1.3.6.1.4.1.140.300.28.1.1</code>
<code>tuxTnetGrpNo</code>	<code>.1.3.6.1.4.1.140.300.28.1.2</code>
<code>tuxTnetGrpState</code>	<code>.1.3.6.1.4.1.140.300.28.1.3</code>
<code>tuxTnetGrpPrio</code>	<code>.1.3.6.1.4.1.140.300.28.1.4</code>

## tuxTnetGrpName

Syntax	<i>DisplayString</i> (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	<p>A GET request retrieves configuration information for the selected <code>tuxTnetGrpTbl</code> instance (or instances). The following states indicate the meaning of the value that is returned:</p> <p>GET: <code>valid(1)</code> The instance is defined. This is the only valid state for this object.</p> <p>SET: <code>invalid(2)</code> Delete the selected <code>tuxTnetGrpTbl</code> instance from the application.</p> <p>States not listed are not returned.</p>

### tuxTnetGrpPrio

Syntax	INTEGER (1..8191)
Access	read-write
Description	<p>The priority band for this network group. All network groups that have an equivalent band priority are used in parallel.</p>

# 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`.

**Note:** This table is supported only on TUXEDO 6.4 or later.

Variable Name	Object ID
<code>tuxTnetMapGrpName</code>	.1.3.6.1.4.1.140.300.33.1.1
<code>tuxTnetMapGrpNo</code>	.1.3.6.1.4.1.140.300.33.1.2
<code>tuxTnetMapLmid</code>	.1.3.6.1.4.1.140.300.33.1.3
<code>tuxTnetMapState</code>	.1.3.6.1.4.1.140.300.33.1.4
<code>tuxTnetMapNaddr</code>	.1.3.6.1.4.1.140.300.33.1.5
<code>tuxTnetMapMinEncryptBit</code>	.1.3.6.1.4.1.140.300.33.1.6
<code>tuxTnetMapMaxEncryptBit</code>	.1.3.6.1.4.1.140.300.33.1.7

## tuxTnetMapGrpName

Syntax	<code>DisplayString (SIZE(1..30))</code>
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	<i>DisplayString</i> (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 <code>tuxTnetMapTbl</code> instance (or instances). The following states indicate the meaning of the value of <code>tuxTnetMapState</code> that is returned:

GET: valid(1)

The instance is defined. This is the only valid state for this object.

SET: invalid(2)

Delete the selected `tuxTnetMapTbl` instance from the application. If any network links were active as a result of the mapping, they will be disconnected. This disconnection may cause a state change in `tuxTBridgeTbl` instances associated with the network links.

States not listed are not returned.

## tuxTnetMapNaddr

Syntax	<code>DisplayString (SIZE (1..78))</code>
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 <code>tuxTdomainOptions</code> is <code>lan(1)</code> . If the string is of the form <code>0xhex-digits</code> or <code>\\xhex-digits</code> , 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 <code>//hostname:port</code> or <code>#. #. #. #:port</code> format is used.

## tuxTnetMapMinEncryptBit

Syntax	<code>INTEGER { none(1), 40-bit(2), 128-bit(3), unknown(4) }</code>
Access	read-write
Description	Specifies the required level of encryption when establishing a network link to this machine. <code>none(1)</code> means no encryption while <code>40-bit(2)</code> and <code>128-bit(3)</code> 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 <code>none(1)</code> . Modifications to this object do not affect network links that have already been established.

## tuxTnetMapMaxEncryptBit

Syntax	<code>Integer {none(1), 40-bit(2), 128-bit(3), unknown(4) }</code>
Access	read-write
Description	Encryption can be negotiated up to the specified level when establishing a network link. <code>none(1)</code> means no encryption while <code>40-bit(2)</code> and <code>128-bit(3)</code> specify the encryption key length (in bits). The default value is <code>128-bit(3)</code> . Modifications to this object will not affect network links that are already established.

# beaEventFilters

You can use the TUXEDO event filters to define a subset of TUXEDO event notifications to be generated for each TUXEDO or M3 domain being monitored. The columnar objects in the `beaEvtFilterTable` correspond to fields in `TMEVENT_FILTER` entries in the BEA Manager configuration file (`beamgr.conf`). Refer to the “Configuration Files” chapter in the *Agent Integrator Reference Manual*.

Variable Name	Object ID
<code>beaEvtFilterTblStatus</code>	<code>.1.3.6.1.4.1.140.300.14.1</code>
<code>beaEvtFilterTable</code>	<code>.1.3.6.1.4.1.140.300.14.2</code>

## beaEvtFilterTblStatus

Syntax	<code>INTEGER { sync(1), dirty(2) }</code>
Access	read-write
Description	When the agent starts, this value is always <code>sync(1)</code> . If any change is done to <code>beaEvtFilterTable</code> through <code>SET</code> requests, the value of this object becomes <code>dirty(2)</code> and the changes made to <code>beaEvtFilterTable</code> do not take effect. The changes made to the <code>beaEvtFilterTable</code> take effect only when you set the value of this object to <code>sync(1)</code> . When you set the value to <code>sync(1)</code> , all changes since the last synchronization are applied to the event-processing module.



## beaEvtFilterTable

This MIB group represents all the event filters defined for the Agent Connection. These are used to determine the collection of events to be forwarded as SNMP trap notifications.

**Note:** Changes to this table are applied only once `beaEvtFilterStatus` is set to `sync(1)`.

Variable Name	Object ID
beaEvtFilterId	.1.3.6.1.4.1.140.300.14.1.1.1
beaEvtAgentName	.1.3.6.1.4.1.140.300.14.1.1.2
beaEvtExpr	.1.3.6.1.4.1.140.300.14.1.1.3
beaEvtFilter	.1.3.6.1.4.1.140.300.14.1.1.4
beaEvtFilterState	.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	<code>DisplayString (SIZE (1..255) )</code>
Access	read-write
Description	An event name expression. Consult the <i>BEA TUXEDO Reference Manual</i> entry for <code>recomp(3)</code> for the format of this expression. For a TUXEDO system event to be forwarded as an SNMP trap, its name should match this expression. Consult the <i>BEA TUXEDO Reference Manual</i> for a list of TUXEDO event names. The default for this object is all system events.
Examples	<p><code>\.Sys.*</code></p> <p>matches all system events. (This is the default.)</p> <p><code>\.SysServer.*</code></p> <p>matches all system events related to servers.</p> <p>A value of <code>NONE</code> blocks all events from being forwarded by the selected agent and overrides any other filter table entries for the same logical agent name.</p>

### beaEvtFilter

Syntax	<code>DisplayString (SIZE (1..255) )</code>
Access	read-write
Description	<p>An event filter expression. Each TUXEDO event is accompanied with an FML buffer containing 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 <code>TRUE</code> or the event is not forwarded.</p> <p>The Agent Connection uses this attribute as an argument to <code>tpsubscribe()</code>. Please refer to the <i>BEA TUXEDO Reference Manual</i> for further information.</p>
Example	<p><code>TA_EVENT_SEVERITY== 'ERROR'    TA_EVENT_SEVERITY== 'WARN'</code></p> <p>This filter selects events with a severity of either <code>ERROR</code> or <code>WARNING</code>.</p>

## 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 will retrieve configuration and runtime 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 will not be 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 will update 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 may not be set.

`active(1)`

Activate the event filter. This can be done only when the filter is in the `inactive(2)` state.

`inactive(2)`

Inactivate the event filter. This can be done only when the filter is in the `active(1)` state.

`invalid(3)`

Inactivate (if active) and remove this event filter.



# 5 BEA Domain List

The MIB group `beaDomainList` represents information about the M3 or TUXEDO domain the agent is monitoring, as specified at startup.

**Note:** Row creation is not allowed in this MIB group. A minimal `tuxconfig` file must exist before starting the agent.

Variable Name	Object ID
<code>beaDomainKey</code>	<code>.1.3.6.1.4.1.140.305.1.1</code>
<code>beaLogicalAgentName</code>	<code>.1.3.6.1.4.1.140.305.1.2</code>
<code>beaDomainID</code>	<code>.1.3.6.1.4.1.140.305.1.3</code>
<code>beaDomainTuxdir</code>	<code>.1.3.6.1.4.1.140.305.1.4</code>
<code>beaDomainTuxconfig</code>	<code>.1.3.6.1.4.1.140.305.1.5</code>
<code>beaDomainStatus</code>	<code>.1.3.6.1.4.1.140.305.1.6</code>

## beaDomainKey

Syntax    `INTEGER (32769..262143)`

Access    read-only

Description    Numeric key for the well-known address in a TUXEDO System/T 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 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	<i>DisplayString</i> (SIZE(1..32))
Access	read-only
Description	The logical agent name of the agent as specified in the -l option when the 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 is the agent which is monitoring the domain. If there are multiple Agent Connection 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 <i>simp_snmpd</i> and <i>bank_snmpd</i> , the default communities used to query values from these agents would be <i>public@simp_snmpd</i> and <i>public@bank_snmpd</i> , respectively. To run multiple agents on the same managed node, they must be run as subagents (without the -s option) with the Agent Integrator.

### beaDomainId

Syntax	<i>DisplayString</i> (SIZE(1..30))
Access	read-only
Description	This is the BEA domain identifier of the domain being managed by this agent. This object is optional.

### beaDomainTuxdir

Syntax	<i>DisplayString</i> (SIZE(1..78))
Access	read-only
Description	The <i>tuxdir</i> value for the domain being managed by this agent. <i>tuxdir</i> is the absolute path name to the directory where the TUXEDO software is found on the master machine.

---

## beaDomainTuxconfig

Syntax	<code>DisplayString(SIZE(1..64))</code>
Access	read-only
Description	The absolute location, including file name, for the configuration file of the domain being managed by this agent.

## beaDomainStatus

Syntax	<code>INTEGER { active(1), inactive(2) }</code>
Access	read-only
Description	This object represents the current state of the domain being managed. The values and their interpretation are the same as for <code>tuxTdomainState</code> .





# 6 M3 MIB Groups

This chapter describes five MIB tables that are specific to the M3 product. To access these MIB objects, the M3 version of the Agent Connection should be running on the machine where M3 application resources are accessible. Table 6-1 lists the M3 groups. In addition to the objects in these M3 specific groups, the TUXEDO Core MIB contains the following M3 specific objects:

- ◆ m3MaxObjects
- ◆ m3MaxInterfaces
- ◆ m3CurInterfaces
- ◆ m3HwInterfaces
- ◆ m3MachineMaxObjects
- ◆ m3MachineCurObjects
- ◆ m3MachineHwObjects
- ◆ m3SrvrCurObjsExt
- ◆ m3SrvrCurInterfaceExt

The object `tuxTranGstate` also has M3 specific states. For more information on these objects, refer to Chapter 4, “TUXEDO Core MIB.”

**Table 6-1 M3 Specific MIB Groups**

Group Name	Description
m3FactoryTable	This table represents occurrences of factories registered with the FactoryFinder. The available factories for the M3 application are reflected in this MIB group.
m3InterfaceTable	This group represents the configuration and runtime attributes of CORBA interfaces at both the domain and server-group levels.
m3LclInterfaceTable	The instances in this table return local m3InterfaceTable attributes for the local host on which the Agent Connection is running.
m3IfQueueTable	The instances in this table represent the runtime attributes of interface as it pertains to a particular server queue (tuxTqueue).
m3LclIfQueueTable	The instances in this table represent the local attributes of m3IfQueueTable instances. These values are specific to the host on which the Agent Connection is running.

# m3FactoryTable

This table represents occurrences of factories registered with the FactoryFinder.

Variable Name	Object ID
m3FactorySerNo	.1.3.6.1.4.1.140.300.48.1.1
m3FactoryId	.1.3.6.1.4.1.140.300.48.1.2
m3FactoryIfName	.1.3.6.1.4.1.140.300.48.1.3
m3FactoryState	.1.3.6.1.4.1.140.300.48.1.4

## m3FactorySerNo

Syntax	INTEGER
Access	read-only
Description	This object is the running number. This is used as the index to instances in this table.

## m3FactoryId

Syntax	<i>DisplayString</i> (SIZE(1..256))
Access	read-only
Description	The registered ID for the factory.

### m3FactoryIfName

Syntax	<i>DisplayString</i> (SIZE(1..128))
Access	read-only
Description	The fully qualified interface name for the factory. This is the interface repository ID for the factory. The format of this name is dependent on the options specified in the IDL which generates the interface implementation. Consult the CORBA 2.1 specification, section 7.6, for details.

### m3FactoryState

Syntax	INTEGER { active(1) }
Access	read-only
Description	A GET operation retrieves runtime information for the selected m3FactoryTable instance or instances. The returned value is 1 (active) if the instance is registered with the FactoryFinder.

# m3InterfaceTable

The `m3InterfaceTable` represents configuration and runtime attributes of CORBA interfaces at both the domain and server-group levels.

There are certain semantic differences in the objects in the `m3InterfaceTable` between server-group and domain level instances.

A domain-level `m3InterfaceTable` instance is an instance that is not associated with a Server group. In this case, its `m3IfSrvGrp` attribute will have the invalid value `*`.

A server-group level instance is an instance that has an associated Server group. In this case, its `m3IfSrvGroup` attribute 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 `m3IfState` object) and for collecting accumulated statistics.

Every CORBA interface that is activated in a server must have a server-group level `m3InterfaceTable` instance. The activation of interfaces in a server is controlled by the state of a `m3IfQueue` instance for the interface. Activation of an `m3IfQueue` instance causes its attributes to be initialized with values specified for the associated server-group level `m3InterfaceTable` instance. If such an instance does not already exist, then one will be dynamically created. This dynamically created server-group level `m3InterfaceTable` instance will be initialized with the attributes of the domain-level `m3InterfaceTable` instance for the interface if one exists. If an associated domain-level instance does not exist, system-specified default configuration values will be used. After they are activated, interfaces are always associated with a server-group level `m3InterfaceTable` instance.

The specification of configuration attributes for interfaces at any level is optional. Interfaces offered by a server are identified through the ICF file used for generating skeletons and advertised automatically by the system when the server is activated.

The following table lists the objects within the `m3InterfaceTable`.

Variable Name	Object ID
m3IfSerNo	.1.3.6.1.4.1.140.300.53.1.1.1
m3IfName	.1.3.6.1.4.1.140.300.53.1.1.2
m3IfSrvGrp	.1.3.6.1.4.1.140.300.53.1.1.3
m3IfState	.1.3.6.1.4.1.140.300.53.1.1.4
m3IfAutoTran	.1.3.6.1.4.1.140.300.53.1.1.5
m3IfLoad	.1.3.6.1.4.1.140.300.53.1.1.6
m3IfPrio	.1.3.6.1.4.1.140.300.53.1.1.7
m3IfTimeout	.1.3.6.1.4.1.140.300.53.1.1.8
m3IfTranTime	.1.3.6.1.4.1.140.300.53.1.1.9
m3IfFbRoutingName	.1.3.6.1.4.1.140.300.53.1.1.10
m3IfLmid	.1.3.6.1.4.1.140.300.53.1.1.11
m3IfNumServers	.1.3.6.1.4.1.140.300.53.1.1.12
m3IfTpPolicy	.1.3.6.1.4.1.140.300.53.1.1.13
m3IfTxPolicy	.1.3.6.1.4.1.140.300.53.1.1.14

**m3IfSerNo**

Syntax	INTEGER
Access	read-only
Description	This is the running number. This object is used as an index to instances in this table.

## m3IfName

Syntax	<i>DisplayString</i> (SIZE(1..128))
Access	read-only
Description	The fully qualified interface name. This is the interface ID. The format of this name is one of the options specified in the IDL which generates the interface implementation. Consult the CORBA 2.1 specification, Section 7.6, for details.

## m3IfSrvGrp

Syntax	<i>DisplayString</i> (SIZE(1..30))
Access	read-write
Description	The server group name. Server group names cannot contain an asterisk, comma, or colon. If an asterisk (*) is specified as a value for this object, this specifies a domain level instance.

**Note:** This object can be SET only during creation of a new row.

## m3IfState

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 will retrieve configuration information for the selected m3InterfaceTable instance or instances. The only states that may be returned are: active, inactive, suspended, partitioned.

active(1)

The m3InterfaceTable instance is defined and at least one corresponding m3IfQueueTable instance is in the active state. For a server-group level m3InterfaceTable instance, corresponding m3IfQueueTable instances are those with matching m3IfName and m3IfSrvGrp objects. For a domain-level m3InterfaceTable

instance, corresponding `m3IfQueueTable` instances are those with matching `m3IfName` value regardless of their `m3IfSrvGrp` value.

`inactive(2)`

The `m3InterfaceTable` instance is defined and there are no corresponding `m3IfQueueTable` instances in any active state.

`suspended(3)`

The `m3InterfaceTable` instance is defined and amongst all corresponding `m3IfQueueTable` instances there are none in the active state and at least one in the suspended state.

`partitioned(4)`

The `m3InterfaceTable` instance is defined and amongst all the corresponding `m3IfQueueTable` 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 runtime and configuration information for the selected `m3InterfaceTable` instance. Modifications may affect more than one server group when making domain-level changes and runtime modifications may affect more than one server if multiple servers are currently offering an interface. Only the following values may be used in a SET request: `invalid`, `active`, `reactivate`, or `suspended`.

`invalid(5)`

Delete the `m3InterfaceTable` instance for the application. This state change is allowed only when the instance is in the inactive state.

`active(1)`

Activate the `m3InterfaceTable` instance for the application. Setting this state on a domain level instance has the effect of activating all corresponding `m3IfQueueTable` 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 `m3InterfaceTable` instance. Setting this state on a domain level instance has the effect of activating all corresponding `m3IfQueueTable` 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)` states. Successful return leaves the instance in the `active(1)` state. Setting this state permits global activation of `m3IfQueueTable` instances suspended at the server-group level without having to individually activate each server-group level `m3InterfaceTable` instance.

`suspended(3)`

Suspend the `m3InterfaceTable` instance. Setting this state on the domain-level object has the effect of suspending all corresponding `m3IfQueueTable` 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 when 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)`).

## m3IfAutoTran

Syntax    `INTEGER { yes(1), no(2) }`

Access    read-write

Description    Signifies whether a transaction will be automatically started for invocations made outside of a transaction context.

This object has the following limitations:

- ◆ Runtime updates to this attribute are not reflected in active equivalent `m3InterfaceTable` instances.
- ◆ The `m3IfTxPolicy` object may override the value specified for this attribute in the `ubbconfig` file. If `m3IfTxPolicy` is `always(1)`, an `m3IfAutoTran` value of `no(2)` will have no effect at runtime. Behavior will be as though the setting were `yes(1)`. If `m3IfTxPolicy` is `never(2)`, an `m3IfAutoTran` value of `yes(1)` will have no effect. The interface will never be involved in a transaction. If `m3IfTxPolicy` is `ignore(4)`, an `m3IfAutoTran` value of `yes(1)` will have no effect. The interface will never be involved in a transaction.

### m3IfLoad

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 will be less likely to be chosen for a new request.
	<b>Note:</b> Runtime updates to this attribute for domain level instances will not affect corresponding server-group level instances for the same interface.

### m3IfPrio

Syntax	INTEGER (1..100)
Access	read-write
Description	Dequeueing priority. If multiple interface requests are waiting on a queue for servicing, the higher priority requests are handled first.
	<b>Note:</b> Runtime updates to this attribute for domain-level instances will not affect corresponding server-group level instances for the same interface.

### m3IfTimeout

Syntax	INTEGER
Access	read-write
Description	The time limit (in seconds) for processing individual method invocations for this interface. Servers processing method invocations for this interface will be terminated abortively if they exceed the specified time limit in processing the request. A value of 0 for this attribute indicates that the server should not be terminated abortively.
	<b>Note:</b> Runtime updates to this attribute for domain-level instances will not affect corresponding server-group level instances for the same interface.

## m3IfTranTime

Syntax     INTEGER

Access     read-write

Description     Transaction timeout value in seconds for transactions automatically started for this `m3InterfaceTable` instance. Transactions are started automatically when a request not in transaction mode is received and the `m3IfAutoTran` object value for the interface is `yes(1)`.

**Note:** Runtime updates to this attribute for domain-level instances will not affect corresponding server-group level instances for the same interface.

## m3IfFbRoutingName

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

Access     read-write

Description     The factory-based routing criteria associated with this interface.

**Note:** This attribute may be set only for a domain-level `m3InterfaceTable` instance, that is, only if `m3IfSrvGrp` is `*`.

## m3IfLmid

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

Access     read-only

Description     Current logical machine with which the active equivalent server-group level `m3InterfaceTable` instance is associated. This attribute is `NULL` for domain-level instances.

## m3IfNumServers

Syntax     INTEGER

Access     read-only

Description     The number of corresponding servers offering this interface.

### m3IfTpPolicy

Syntax	INTEGER { method(1), transaction(2), process(3) }
Access	read-only
Description	The TP framework deactivation policy. This reflects the policy registered with the framework at server startup. The first server to register with the interface sets the value in m3InterfaceTable. This cannot be changed.

### m3IfTxPolicy

Syntax	INTEGER { always(1), never(2), optional(3), ignore(4) }
Access	read-only
Description	The transaction policy for the interface. The setting in this attribute affects the m3IfAutoTran object. This policy is set by the application developer and is registered when the server starts.

# m3LclInterfaceTable

The table returns values for the local host on which the Agent Connection is running. The following table lists the columnar objects that comprise each row (instance) in the table.

Variable Name	Object ID
m3LclIfSerNo	.1.3.6.1.4.1.140.300.53.2.1.1
m3LclIfName	.1.3.6.1.4.1.140.300.53.2.1.2
m3LclSrvGrp	.1.3.6.1.4.1.140.300.53.2.1.3
m3LclIfNcompleted	.1.3.6.1.4.1.140.300.53.2.1.4
m3LclIfNqueued	.1.3.6.1.4.1.140.300.53.2.1.5

## m3LclIfSerNo

Syntax INTEGER

Access read-only

Description This is the running number. This is used as an index into the table.

## m3LclIfName

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

Access read-only

Description The fully qualified interface name. The interface repository ID for the interface. The format of this name is dependent on the options specified in the IDL which generates the interface implementation. See the CORBA 2.1 Specification Section 7.6 [CORBA] for details.

### m3LclIfSrvGrp

Syntax	<i>DisplayString</i> (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.

### m3LclIfNcompleted

Syntax	INTEGER
Access	read-only
Description	The number of method invocations completed for the corresponding <code>m3IfQueueTable</code> instances since they were initially offered. The values returned are for the indicated interface on the local host where the Agent Connection is running.

**Note:** This attribute is returned only when `tuxTdomainLoadBalance` is `yes(1)`.

### m3LclIfNqueued

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 the Agent Connection is running.

**Note:** This attribute is returned only when `tuxTdomainLoadBalance` is `yes(1)`.

# m3IfQueueTable

This table represents the runtime attributes of an interface for a particular server queue. This group provides access to the inherited configuration attributes of an interface as well as statistics relating to the interface on the queue. This class 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, `m3IfQRqAddr` can be used to identify the corresponding server in the `tuxTsrvrTbl` and `tuxTsrvrTblExt` groups.

Variable Name	Object ID
<code>m3IfQueueSerNo</code>	<code>.1.3.6.1.4.1.140.300.53.3.1.1</code>
<code>m3IfQueueName</code>	<code>.1.3.6.1.4.1.140.300.53.3.1.2</code>
<code>m3IfQueueSrvGrp</code>	<code>.1.3.6.1.4.1.140.300.53.3.1.3</code>
<code>m3IfQueueRqAddr</code>	<code>.1.3.6.1.4.1.140.300.53.3.1.4</code>
<code>m3IfQueueState</code>	<code>.1.3.6.1.4.1.140.300.53.3.1.5</code>
<code>m3IfQueueAutoTran</code>	<code>.1.3.6.1.4.1.140.300.53.3.1.6</code>
<code>m3IfQueueLoad</code>	<code>.1.3.6.1.4.1.140.300.53.3.1.7</code>
<code>m3IfQueuePrio</code>	<code>.1.3.6.1.4.1.140.300.53.3.1.8</code>
<code>m3IfQueueTimeout</code>	<code>.1.3.6.1.4.1.140.300.53.3.1.9</code>
<code>m3IfQueueTranTime</code>	<code>.1.3.6.1.4.1.140.300.53.3.1.10</code>
<code>m3IfQueueFbRoutingName</code>	<code>.1.3.6.1.4.1.140.300.53.3.1.11</code>
<code>m3IfQueueLmid</code>	<code>.1.3.6.1.4.1.140.300.53.3.1.12</code>
<code>m3IfQueueNumServers</code>	<code>.1.3.6.1.4.1.140.300.53.3.1.13</code>
<code>m3IfQueueTpPolicy</code>	<code>.1.3.6.1.4.1.140.300.53.3.1.14</code>
<code>m3IfQueueTxPolicy</code>	<code>.1.3.6.1.4.1.140.300.53.3.1.15</code>

## **m3IfQueueSerNo**

Syntax	INTEGER
Access	read-only
Description	The running number used as an index into this table.

## **m3IfQueueName**

Syntax	<i>DisplayString</i> (SIZE(1..128))
Access	read-only
Description	The fully qualified interface name. The interface repository ID for the interface. The format of this name is dependent on the options specified in the IDL which generates the interface implementation. See the CORBA 2.1 specification Section 7.6 for details.

## **m3IfQueueSrvGrp**

Syntax	<i>DisplayString</i> (SIZE(1..30))
Access	read-only
Description	The server group name. Server group names cannot contain an asterisk, comma, or colon.

## **m3IfQueueRqAddr**

Syntax	<i>DisplayString</i> (SIZE(1..30))
Access	read-only
Description	The symbolic address of the request queue for an active server offering this interface. See <code>tuxTsrvrRqAddr</code> for more information about this attribute.



## m3IfQueueState

Syntax	INTEGER { active(1), suspended(2), partitioned(3), unknown(4) }
Access	read-write
Description	<p>The values for GET and SET operations are as follows:</p> <p>GET: {active(1)   suspended(2)   partitioned(3) }</p> <p>A GET request will retrieve configuration information for the selected m3IfQueueTable instances. The meaning of the possible return values are as follows:</p> <p>active(1) Represents an available interface in the running system.</p> <p>suspended(2) Represents a currently suspended interface in the running system.</p> <p>partitioned(3) Represents a currently partitioned interface in the running system.</p> <p>SET: {active(1)   suspended(2) }</p> <p>The values for SET are:</p> <p>active(1) Activates the m3IfQueueTable instance. This state change is allowed only when in the suspended(2) state. A successful return leaves instances in the active(1) state.</p> <p>suspended(2) Suspends the m3IfQueueTable instance. This state change is allowed only when in the active(1) state. A successful return leaves the object in the suspended(2) state.</p> <p><b>Note:</b> Dynamic advertisement of interfaces (i.e., a state change from inactive or invalid to active) is not supported, nor is unadvertisement (i.e., a state change from active to inactive).</p>

## **m3IfQueueAutoTran**

Syntax	INTEGER { yes(1), no(2) }
Access	read-only
Description	Signifies whether a transaction will be automatically started for invocations made outside a transaction context.

This object has the following limitations:

- ◆ Runtime updates to this attribute are not reflected in active equivalent `m3InterfaceTable` instances.
- ◆ The `m3IfTxPolicy` object may override the value specified for this attribute in the `ubbconfig` file. If `m3IfTxPolicy` is `always(1)`, an `m3IfQueueAutoTran` value of `no(2)` will have no effect at runtime. Behavior will be as though the setting were `yes(1)`. If `m3IfTxPolicy` is `never(2)`, an `m3IfQueueAutoTran` value of `yes(1)` will have no effect. The interface will never be involved in a transaction. If `m3IfTxPolicy` is `ignore(4)`, an `m3IfQueueAutoTran` value of `yes(1)` will have no effect. The interface will never be involved in a transaction.

## **m3IfQueueLoad**

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.

## **m3IfQueuePrio**

Syntax	INTEGER (1..101)
Access	read-only
Description	Dequeueing priority. If multiple interface requests are waiting on a queue for servicing, the higher priority requests will be handled first.

## m3IfQueueTimeout

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 will be abortively terminated if they exceed the specified time limit in processing the request. A value of 0 for this attribute indicates that the server should not be abortively terminated.

## m3IfQueueTranTime

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 m3IfAutoTran attribute value for the interface is <code>yes(1)</code> .

## m3IfQueueFbRoutingName

Syntax	<i>DisplayString</i> (SIZE(1..15))
Access	read-only
Description	The factory-based routing criterion associated with this interface.

## m3IfQueueLmid

Syntax	<i>DisplayString</i> (SIZE(1..30))
Access	read-only
Description	The current logical machine on which this queue is offering this interface.

### **m3IfQueueNumServers**

Syntax	INTEGER
Access	read-only
Description	The number of corresponding servers offering this interface on this queue.

### **m3IfQueueTpPolicy**

Syntax	INTEGER { method(1), transaction(2), process(3) }
Access	read-only
Description	The TP framework deactivation policy. This 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.

# m3LclIfQueueTable

This table represents the local attributes of the m3IfQueueTable. These values are specific to the host on which the Agent Connection is running.

Variable Name	Object ID
m3LclIfQueueSerNo	.1.3.6.1.4.1.140.300.53.4.1.1
m3LclIfQueueName	.1.3.6.1.4.1.140.300.53.4.1.2
m3LclIfQueueSrvGrp	.1.3.6.1.4.1.140.300.53.4.1.3
m3LclIfQueueRqAddr	.1.3.6.1.4.1.140.300.53.4.1.4
m3LclIfQueueNcompleted	.1.3.6.1.4.1.140.300.53.4.1.5
m3LclIfQueueNqueued	.1.3.6.1.4.1.140.300.53.4.1.6
m3LclIfQueueCurObjs	.1.3.6.1.4.1.140.300.53.4.1.7
m3LclIfQueueCurTrans	.1.3.6.1.4.1.140.300.53.4.1.8

## m3LclIfQueueSerNo

Syntax	INTEGER
Access	read-only
Description	The running number used as an index into this table.

## m3LclIfQueueName

Syntax	<i>DisplayString</i> (SIZE(1..128))
Access	read-only
Description	The fully qualified interface name. 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. See the CORBA 2.1 specification Section 7.6 for details.

### m3LclIfQueueSrvGrp

Syntax	<i>DisplayString</i> (SIZE(1..30))
Access	read-only
Description	The server group name. Server group names cannot contain an asterisk, comma, or colon.

### m3LclIfQueueRqAddr

Syntax	<i>DisplayString</i> (SIZE(1..30))
Access	read-only
Description	The symbolic address of the request queue for an active server offering this interface. See <code>tuxTsrvrRqAddr</code> for more information about this attribute.

### m3LclIfQueueNcompleted

Syntax	INTEGER
Access	read-only
Description	The number of interface method invocations completed since the interface was initially offered.
<b>Note:</b>	This attribute is returned only when <code>tuxTdomainLoadBalance</code> is equal to <code>yes(1)</code> .

### m3LclIfQueueNqueued

Syntax	INTEGER
Access	read-only
Description	The number of requests currently enqueued for this interface.
<b>Note:</b>	This attribute is returned only when <code>tuxTdomainLoadBalance</code> is equal to <code>yes(1)</code> .

## m3LclIfQueueCurObjs

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 includes objects that are not in memory but were invoked within an active transaction.

## m3LclIfQueueCurTrans

Syntax     INTEGER

Access     read-only

Description     The number of active global transactions associated with this interface for its associated queue.





# 7 Access Control List MIB

An *access control list* (ACL) is a list that specifies who and what is authorized to access TUXEDO system objects. The ACL MIB enables a system manager to administer TUXEDO security through authenticating users, setting permissions, and controlling access. The ACL MIB defines the objects controlled by the ACL facility. These MIB objects are grouped into three major categories. The following table lists groups that comprise the ACL MIB.

Group Name	Description
tuxTAclGrpTable	ACL group
tuxTAclPermTable	ACL permissions
tuxTAclPrinTbl	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 ACL 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 this are defined in the `tuxTAclPermTable` group. All of these ACL MIB groups and their objects are described in the following sections.

# tuxTAclGrpTable

The tuxTAclGrpTable group represents groups of TUXEDO application users and domains. The following table lists the managed objects that are part of the tuxTAclGrpTable group. To create a new row in the table, it is necessary to issue a SET request for a non-existing row.

Variable Name	Object ID
tuxTAclGrpName	.1.3.6.1.4.1.140.300.11.1.1.1.1
tuxTAclGrpId	.1.3.6.1.4.1.140.300.11.1.1.1.2
tuxTAclGrpState	.1.3.6.1.4.1.140.300.11.1.1.1.3

## tuxTAclGrpName

Syntax	<i>DisplayString</i> (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.

## tuxTAclGrpId

Syntax	INTEGER (0..16384)
Access	read-write
Description	Group identifier associated with this user. A value of 0 indicates the default group other. If 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 will retrieve 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 will not be returned.

`valid(1)`

`tuxTAcLGrpTable` instance is defined and inactive. Note that this is the only valid state for this class. ACL groups are never active.

SET: `invalid(2)`

A SET operation will update configuration information for the selected `tuxTAcLGrpTable` instance. The following state indicates the meaning of a `tuxTAcLGrpState` set in a SET request. States not listed may 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 via 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`.

Variable 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	<i>DisplayString</i> (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	<p>The values for GET and SET operations are as follows:</p> <p>GET: valid(1)</p> <p>A GET operation will retrieve 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 will not be returned.</p> <p>valid(1)</p> <p>tuxTAcLPermState instance is defined. Note that this is the only valid state for this class. ACL permissions are never active.</p> <p>SET: invalid(2)</p> <p>A SET operation will update configuration information for the selected tuxTAcLPermState instance. The following state indicates the meaning of a tuxTAcLPermState set in a SET request. States not listed may not be set.</p> <p>invalid(2)</p> <p>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.</p>

Note that the tuxTAcLPermTable instance refers to all groupids related to a particular tuxTAcLPermName in the table.

# tuxTAclPrinTbl

The `tuxTAclPrinTbl` group represents 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).

Variable Name	Object ID
<code>tuxTAclPrinName</code>	<code>.1.3.6.1.4.1.140.300.11.3.1.1.1</code>
<code>tuxTAclCltnName</code>	<code>.1.3.6.1.4.1.140.300.11.3.1.1.2</code>
<code>tuxTAclPrinId</code>	<code>.1.3.6.1.4.1.140.300.11.3.1.1.3</code>
<code>tuxTAclPrinGrp</code>	<code>.1.3.6.1.4.1.140.300.11.3.1.1.4</code>
<code>tuxTAclPrinPasswd</code>	<code>.1.3.6.1.4.1.140.300.11.3.1.1.5</code>
<code>tuxTAclPrinState</code>	<code>.1.3.6.1.4.1.140.300.11.3.1.1.6</code>

## 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	<i>DisplayString</i> (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 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 <i>other</i> . If not specified at creation time, the default value 0 is assigned.

## tuxTAclPrinPasswd

Syntax	<i>DisplayString</i>
Access	read-write
Description	The clear-text authentication password for the associated user. Note that the system will automatically encrypt this information on behalf of the administrator.

## **tuxTAclPrinState**

Syntax	INTEGER { valid(1), invalid(2) }
Access	read-write
Description	<p>The values for GET and SET operations are as follows:</p> <p>GET: valid(1)  A GET operation will retrieve configuration information for the selected tuxTAclPrinTbl instance(s). The following state indicates the meaning of tuxTAclPrinState:</p> <p>valid(1)  tuxTAclPrinTbl instance is defined and inactive. Note that this is the only valid state for this class. ACL principals are never active.</p> <p>SET: invalid(2)  A SET operation will update configuration information for the selected tuxTAclPrinTbl instance. The following state indicates the meaning of a tuxTAclPrinState set in a SET request. States not listed may not be set.</p> <p>invalid(2)  Delete tuxTAclPrinTbl instance for application. State change allowed only when in the valid(1) state. Successful return leaves the object in the invalid(2) state.</p>



# 8 Workstation MIB

TUXEDO systems include the capability to require that clients run on a workstation for purposes of security, performance, and convenience. A network administrator can define the environment required to control the workstation clients using the Workstation MIB. This MIB is an extension of the TUXEDO 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
<code>tuxTwshTbl</code>	Workstation Handler
<code>tuxTws1Tbl</code>	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 runtime attributes 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.

Variable Name	Object ID
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

Variable Name	Object ID
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
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	<p>State for the WSH client within the application. Any state defined for the <code>tuxTclientTbl</code> group may be returned or set. State changes to the <code>suspended(2)</code> state are transitive to all clients associated with this WSH as is the resetting of a <code>suspended(2)</code> WSH to <code>active(1)</code>. Additionally, <code>suspended(2)</code> WSH clients will not be assigned any additional incoming clients by the WSL.</p> <p>Note that the state of a WSH client may not be set to <code>dead(3)</code> when accessing the <code>tuxTclientTbl</code> group; however, the state transition to <code>dead(3)</code> is allowed via the <code>tuxTwshTbl</code> group and will result in all connections being handled by the targeted WSH to be dropped abortively.</p>

**tuxTwshTaLmid**

Syntax	<i>DisplayString</i> (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	<i>DisplayString</i> ( 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 via this WSH.

## tuxTwshTaMultiplex

Syntax	INTEGER ( 1 .. 32767 )
Access	read-only
Description	Maximum number of clients that may access the application via this WSH.

## tuxTwshTaCurClients

Syntax	INTEGER ( 1 .. 32767 )
Access	read-only
Description	Current number of clients accessing application via this WSH.

### **tuxTwshTaTimeleft**

Syntax	INTEGER
Access	read-only
Description	A non-0 value for this attribute 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 amongst a set of WSH processes.

## **tuxTwshTaFlowcnt**

Syntax	INTEGER
Access	read-only
Description	Number of times flow control has been encountered by this WSH. This attribute should be considered only in relation to recent past values as it may 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 attribute should be considered only in relation to recent past values as it may 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 of its present and past workstation clients. This attribute should be considered only in relation to recent past values as it may wrap around during the lifetime of the WSH.

### **tuxTwshTaRcvdNum**

Syntax	INTEGER
Access	read-only
Description	Number of TUXEDO System/T messages received from the network by this WSH from all of its present and past workstation clients. This attribute should be considered only in relation to recent past values as it may 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 of its present and past workstation clients. This attribute should be considered only in relation to recent past values as it may wrap around during the lifetime of the WSH.

### **tuxTwshTaSentNum**

Syntax	INTEGER
Access	read-only
Description	Number of TUXEDO System/T messages sent to the network by this WSH to all of its present and past workstation clients. This attribute should be considered only in relation to recent past values as it may wrap around during the lifetime of the WSH.



# tuxTwsLTbl

The `tuxTwsLTbl` table represents configuration and runtime attributes of WSL server processes configured to manage workstation groups. These attribute values identify and characterize workstation-specific configuration attributes 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 `tuxTwsLTaSrvGrp`, `tuxTwsLTaSrvId`, and `tuxTwsLTaNaddr`.

Variable Name	Object ID
<code>tuxTwsLTaSrvGrp</code>	<code>.1.3.6.1.4.1.140.300.1.2.1.1</code>
<code>tuxTwsLTaSrvId</code>	<code>.1.3.6.1.4.1.140.300.1.2.1.2</code>
<code>tuxTwsLTaGrpNo</code>	<code>.1.3.6.1.4.1.140.300.1.2.1.3</code>
<code>tuxTwsLTaState</code>	<code>.1.3.6.1.4.1.140.300.1.2.1.4</code>
<code>tuxTwsLTaLmid</code>	<code>.1.3.6.1.4.1.140.300.1.2.1.5</code>
<code>tuxTwsLTaPid</code>	<code>.1.3.6.1.4.1.140.300.1.2.1.6</code>
<code>tuxTwsLTaDevice</code>	<code>.1.3.6.1.4.1.140.300.1.2.1.7</code>
<code>tuxTwsLTaNaddr</code>	<code>.1.3.6.1.4.1.140.300.1.2.1.8</code>
<code>tuxTwsLTaWshName</code>	<code>.1.3.6.1.4.1.140.300.1.2.1.9</code>
<code>tuxTwsLTaMinHandlers</code>	<code>.1.3.6.1.4.1.140.300.1.2.1.10</code>
<code>tuxTwsLTaMaxHandlers</code>	<code>.1.3.6.1.4.1.140.300.1.2.1.11</code>
<code>tuxTwsLTaMultiplex</code>	<code>.1.3.6.1.4.1.140.300.1.2.1.12</code>
<code>tuxTwsLTaMaxIdleTime</code>	<code>.1.3.6.1.4.1.140.300.1.2.1.13</code>
<code>tuxTwsLTaMaxInitTime</code>	<code>.1.3.6.1.4.1.140.300.1.2.1.14</code>
<code>tuxTwsLTaClOpt</code>	<code>.1.3.6.1.4.1.140.300.1.2.1.15</code>
<code>tuxTwsLTaEnvFile</code>	<code>.1.3.6.1.4.1.140.300.1.2.1.16</code>
<code>tuxTwsLTaGrace</code>	<code>.1.3.6.1.4.1.140.300.1.2.1.17</code>

Variable Name	Object ID
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	<i>DisplayString</i> ( 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.

## tuxTws1TaSrvId

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 may 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 is a required value for access to a network via a TLI-based TUXEDO System/T binary. This attribute is not needed for sockets-based TUXEDO System/T binaries.

### tuxTwslTaNaddr

Syntax	<i>DisplayString</i> ( SIZE(1..78) )
Access	read-write
Description	<p>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.</p> <p>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.</p>

### tuxTwslTaWshName

Syntax	<i>DisplayString</i> ( SIZE(1..78) )
Access	read-write
Description	<p>The name of the executable providing workstation handler services for this workstation listener. The default value for this is WSH, which corresponds to the system provided workstation handler. Workstation handlers may be customized using the command <code>buildwsh</code>.</p>

### tuxTwslTaMinHandlers

Syntax	INTEGER (0..256)
Access	read-write
Description	<p>The minimum number of handlers that should be available in conjunction with this WSL at any given time. The WSL will start this many WSHs immediately upon being activated and will not deplete the supply of WSHs below this number until the administrator issues a shutdown to the WSL. Modifications to this attribute for a running WSL may cause additional handlers to be activated.</p>

## **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 attribute 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 to be 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 will be abortively disconnected from the application by the handler. A value of 0 allows clients to be idle as long as they wish 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.

## tuxTwsITaCLOpt

Syntax	<i>DisplayString</i> ( SIZE ( 0..128 ) )
Access	read-write
Description	Command-line options to be passed to WSL server when it is activated. See the <i>servopts(5)</i> reference page for details.
	<b>Note:</b> Runtime modifications to this attribute will not affect a running WSL server. Server-specific options (i.e., those after a double-dash "--") may not be set and will not be returned.

## tuxTwsITaEnvFile

Syntax	<i>DisplayString</i> ( SIZE ( 0..78 ) )
Access	read-write
Description	WSL server-specific environment file. See <i>tuxTmachineEnvFile</i> for a complete discussion of how this file is used to modify the environment.
	<b>Note:</b> Runtime modifications to this attribute will not affect a running WSL server.

## tuxTwsITaGrace

Syntax	INTEGER
Access	read-write
Description	The period of time, in seconds, over which the <i>tuxTwsITaMaxGen</i> limit applies. This attribute is meaningful only for restartable WSL servers, i.e., if the <i>tuxTwsITaRestart</i> attribute is set to <i>yes(1)</i> . When a restarting server would exceed the <i>tuxTwsITaMaxGen</i> limit but the <i>tuxTwsITaGrace</i> period has expired, the system resets the current generation ( <i>tuxTsrvrGeneration</i> ) to 1 and resets the initial boot time ( <i>tuxTsrvrTimeStart</i> ) to the current time. A value of 0 for this attribute indicates that the WSL server should always be restarted.

## tuxTwsITaMaxGen

Syntax	INTEGER (0..256)
Access	read-write
Description	Number of generations allowed for a restartable WSL server ( <code>tuxTwsITaRestart == yes(1)</code> ) over the specified grace period ( <code>tuxTwsITaGrace</code> ). The initial activation of the WSL 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 <code>tuxTwsITaGrace</code> .

## tuxTwsITaRcmd

Syntax	<i>DisplayString</i> (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 ( <code>yes(1)</code> ) or non-restartable ( <code>no(2)</code> ) WSL server. If server migration is specified for this server group ( <code>tuxTdomainOptions = migrate(2)</code> and <code>tuxTgroupLMID</code> with alternate site), then this attribute must be set to <code>yes(1)</code> .

### **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. tuxTwsITbl instances added without a tuxTwsITaSequence attribute specified or with an invalid value will 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. Runtime modifications to this attribute affect only tmboot(1) and tmshutdown(1) and will affect the order in which running servers may 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/T /WS protocol version number for this /WS group. Note that /WS clients connecting to this group may 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 may not connect through this tuxTwsITbl 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 disallowing new incoming connections. A value of none(3) indicates that no suspension characteristics are in effect.

## tuxTwsITaViewRefresh

Syntax	INTEGER { yes(1), no-value-returned(2) }
Access	read-write
Description	Setting a value of yes(1) will cause 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.

### **tuxTwsITaKeepAlive**

Syntax	INTEGER {client(1), handler(2), both(3), none(4), not-available(5)}
Access	read-write
Description	The network “keep alive” option will be 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 connections that will be established in the future. This object is supported only on TUXEDO 6.4 or later.

### **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 receiving a response from WSL/WSH. A value of 0 indicates no network time-out. Changing this value affects only connections established in the future. This object is supported only on TUXEDO 6.4. -1 is returned if the object is not available.

# 9 Application Queue MIB

The TUXEDO system incorporates 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, queues spaces, and queue transactions. The following table lists the groups for managing each of the queue components.

Group Name	Description
tuxTAppQctrl	Access control to application queues
tuxTAppQTbl	Application queues within a queue space
tuxTAppQmsgTbl	Messages within an application queue
tuxTQspaceTbl	Application queue spaces
tuxTQtransTbl	Transactions associated with application queues

# tuxTAppQctrl

This is a control MIB to enable controlled access to all Application Queue related MIB groups.

Variable Name	Object ID
tuxTAppQctrlLmid	.1.3.6.1.4.1.140.300.12.5.1
tuxTAppQctrlQmConfig	.1.3.6.1.4.1.140.300.12.5.2
tuxTAppQctrlSpaceName	.1.3.6.1.4.1.140.300.12.5.3
tuxTAppQctrlQname	.1.3.6.1.4.1.140.300.12.5.4
tuxTAppQctrlMsgLoPrio	.1.3.6.1.4.1.140.300.12.5.5
tuxTAppQctrlMsgHiPrio	.1.3.6.1.4.1.140.300.12.5.6
tuxTAppQctrlMsgEndTime	.1.3.6.1.4.1.140.300.12.5.7
tuxTAppQctrlMsgStartTime	.1.3.6.1.4.1.140.300.12.5.8

## tuxTAppQctrlLmid

Syntax	INTEGER { local(1), all(2) }
Access	read-write
Description	<p>This applies to all Application Queue related MIB groups. The value of this object controls the machines for which the values are returned.</p> <p>If the value is <code>local(1)</code>, only the local host where the SNMP agent is running is considered; alternatively, all LMIDs known to the application are considered if the value is <code>all(2)</code>.</p> <p>The default value of this object is <code>local(1)</code>.</p>

## tuxTAppQctrlQmConfig

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

Access read-write

Description This applies to all Application Queue related MIB groups. The value of this object controls the device for which the values are returned.

The default value of 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. The value of this object controls the queue space for which the values are returned.

The default value of this object is “\*”, in which case all queue spaces for the devices (qualified by tuxTAppQctrlQmConfig) are considered.

## tuxTAppQctrlQname

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

Access read-write

Description The value of this object controls the queue for which the values are returned. This applies to tuxTAppQTbl and tuxTAppQmsgTbl.

The default value of 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 <code>tuxTAppQmsgTbl</code> . The lowest priority within which to search for occurrences of <code>tuxTAppQmsgTbl</code> instances. This 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 <code>tuxTAppQmsgTbl</code> . The highest priority within which to search for occurrences of <code>tuxTAppQmsgTbl</code> instances. This 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	<i>DisplayString</i> (SIZE(1..15))
Access	read-write
Description	This object applies only to <code>tuxTAppQmsgTbl</code> . The end time within which to search for occurrences of <code>tuxTAppQmsgTbl</code> instances. The range is inclusive. This 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 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 represents application queues. One or more application queues may 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. Please 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`.

Variable Name	Object ID
<code>tuxTAppQname</code>	.1.3.6.1.4.1.140.300.12.1.1.1
<code>tuxTAppQspaceName</code>	.1.3.6.1.4.1.140.300.12.1.1.2
<code>tuxTAppQmConfig</code>	.1.3.6.1.4.1.140.300.12.1.1.3
<code>tuxTAppQlmid</code>	.1.3.6.1.4.1.140.300.12.1.1.4
<code>tuxTAppQgrpNo</code>	.1.3.6.1.4.1.140.300.12.1.1.5
<code>tuxTAppQstate</code>	.1.3.6.1.4.1.140.300.12.1.1.6
<code>tuxTAppQorder</code>	.1.3.6.1.4.1.140.300.12.1.1.7
<code>tuxTAppQcmd</code>	.1.3.6.1.4.1.140.300.12.1.1.8
<code>tuxTAppQcmdHw</code>	.1.3.6.1.4.1.140.300.12.1.1.9
<code>tuxTAppQcmdLw</code>	.1.3.6.1.4.1.140.300.12.1.1.10
<code>tuxTAppQmaxRetries</code>	.1.3.6.1.4.1.140.300.12.1.1.11



Variable Name	Object ID
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

## tuxTAppQname

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

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	<i>DisplayString</i> (SIZE(1..30))
Access	read-write
Description	Identifier of the logical machine where the application queue space is located.
<b>Note:</b>	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 <code>tuxTgroupOpenInfo</code> contains the device name and queue space name for this queue.
<b>Note:</b>	This object can be updated only during row creation.

## tuxTAppQstate

Syntax	INTEGER { valid(1), invalid(2) }
Access	read-write
Description	<p>The values for GET and SET operations are as follows:</p> <p>GET: valid(1)</p> <p>A GET operation retrieves information about the selected application queues. The following list describes the meaning of the <code>tuxTAppQstate</code> attribute returned in response to a GET request. States not listed will not be returned.</p> <p>valid(1)</p> <p>The specified queue exists.</p>

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` attribute 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 will not be deleted. In addition, if the queue has messages in it, it will not be 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 `Prio` or `Time`, followed by a comma, optionally followed by another occurrence of `Prio` 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..78))`

Access read-write

Description The command to be automatically executed when the high water mark, `tuxTAppQcmdHw`, is reached. The command will be re-executed when the high water mark is reached again after the low water mark, `tuxTAppQcmdLw`, has been reached.

## tuxTAppQcmdHw

Syntax	<i>DisplayString</i>
Access	read-write
Description	The high water mark. Refer to tuxTAppQcmdLw for further information.

## tuxTAppQcmdLw

Syntax	<i>DisplayString</i>
Access	read-write
Description	<p>The low water marks that control the automatic execution of the command specified in the tuxTAppQcmd attribute. 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.</p> <p>b                   The high and low water marks pertain to the number of bytes used by messages in the queue.</p> <p>B                   The high and low water marks pertain to the number of blocks used by messages in the queue.</p> <p>m                   The high and low water marks pertain to the number of messages in the queue.</p> <p>%                   The high and low water marks are expressed in terms of a percentage of queue capacity.</p>

For example, if tuxTAppQcmdLw is 50m and tuxTAppQcmdHw is 100m, then the command specified in tuxTAppQcmd will be executed when 100 messages are on the queue, and it will not be executed again until the queue has been drained below 50 messages and has 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.

# tuxTAppQmsgTbl

The `tuxTAppQmsgTbl` group represents 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 attributes 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.

Variable Name	Object ID
<code>tuxTAppQmsgId</code>	<code>.1.3.6.1.4.1.140.300.12.2.1.1</code>
<code>tuxTAppQmsgSerNo</code>	<code>.1.3.6.1.4.1.140.300.12.2.1.2</code>
<code>tuxTAppQmsgGrpNo</code>	<code>.1.3.6.1.4.1.140.300.12.2.1.3</code>
<code>tuxTAppQmsgQname</code>	<code>.1.3.6.1.4.1.140.300.12.2.1.4</code>
<code>tuxTAppQmsgQmConfig</code>	<code>.1.3.6.1.4.1.140.300.12.2.1.5</code>
<code>tuxTAppQmsgQspaceName</code>	<code>.1.3.6.1.4.1.140.300.12.2.1.6</code>
<code>tuxTAppQmsgLmid</code>	<code>.1.3.6.1.4.1.140.300.12.2.1.7</code>
<code>tuxTAppQmsgState</code>	<code>.1.3.6.1.4.1.140.300.12.2.1.8</code>
<code>tuxTAppQmsgNewQname</code>	<code>.1.3.6.1.4.1.140.300.12.2.1.9</code>
<code>tuxTAppQmsgPrior</code>	<code>.1.3.6.1.4.1.140.300.12.2.1.10</code>
<code>tuxTAppQmsgTime</code>	<code>.1.3.6.1.4.1.140.300.12.2.1.11</code>
<code>tuxTAppQmsgCorId</code>	<code>.1.3.6.1.4.1.140.300.12.2.1.12</code>
<code>tuxTAppQmsgCurRetries</code>	<code>.1.3.6.1.4.1.140.300.12.2.1.13</code>
<code>tuxTAppQmsgSize</code>	<code>.1.3.6.1.4.1.140.300.12.2.1.14</code>

## tuxTAppQmsgId

Syntax	<i>DisplayString</i> (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	<i>DisplayString</i> (SIZE(1..15))
Access	read-only
Description	Name of the application queue in which the message is stored.

## tuxTAppQmsgQmConfig

Syntax	<i>DisplayString</i> (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	<i>DisplayString</i> (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	<i>DisplayString</i> (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	<p>The values for GET and SET operations are as follows:</p> <p>GET: valid(1)</p> <p>A GET operation retrieves information about the selected messages. The following list describes the meaning of the tuxTAppQmsgState attribute returned in response to a GET request. States not listed will not be returned.</p> <p>valid(1)</p> <p>The message exists.</p>



SET: `invalid(2)`

A SET operation changes characteristics of the selected message. The following list describes the meaning of the `tuxTAppQmsgState` attribute 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 attribute is not returned by a GET operation.

## tuxTAppQmsgPrior

Syntax    `INTEGER`

Access    read-write

Description    The priority of the message. This attribute 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 will be processed. This attribute 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 will be 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 `topenqueue(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.

# tuxTQspaceTbl

This group represents 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`. Refer to the description of the above group for details.

To create a new row in this table, a SET request should be issued with an index (`tuxTQspaceGrpNo`) of 40000. This 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) will not be visible until it is attached to some server group.

Variable Name	Object ID
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

Variable Name	Object ID
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
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

## 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	<i>DisplayString</i> (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 <code>tuxTgroupOpenInfo</code> 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` attribute returned in response to a GET request. States not listed will not be 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` attribute 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. This 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. This 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	<i>DisplayString</i> (SIZE(0..15))
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	The value of this object 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 <code>tuxTQspaceMaxPages</code> attribute is increased, a new extent is allocated (see <code>tuxTQspaceCurExtent</code> ). It is not possible to decrease the number of pages by setting this attribute 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 <code>tuxTQspaceMaxPages</code> attribute 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 <code>open(3)</code> or <code>active(4)</code> , 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 <code>open(3)</code> or <code>active(4)</code> , 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 <code>cleaning(5)</code> .

### 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 <code>cleaning(5)</code> .

### 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 <code>cleaning(5)</code> .

## 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 <code>cleaning(5)</code> .

## 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.

# tuxTQtransTbl

This group represents runtime attributes of transactions associated with application queue spaces. Objects in this table are only accessible through a TUXEDO SNMP agent installed on the local machine.

Variable 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
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	<i>DisplayString</i> (SIZE(1..78))
Access	read-only
Description	Transaction identifier as returned by <code>tx_info(3)</code> and mapped to a string representation. The data in this field should not be interpreted directly by the user except for equality comparison.

## tuxTQtransIdx1

Syntax	INTEGER
Access	read-only
Description	An integer index for <code>tuxTQtransTbl</code> . 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 <code>Idx1</code> through <code>Idx5</code> .

## tuxTQtransIdx2

Syntax	INTEGER
Access	read-only
Description	An integer index for <code>tuxTQtransTbl</code> . 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 <code>Idx1</code> through <code>Idx5</code> .

## tuxTQtransIdx3

Syntax	INTEGER
Access	read-only
Description	An integer index for <code>tuxTQtransTbl</code> . 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 <code>Idx1</code> through <code>Idx5</code> .

## tuxTQtransIdx4

Syntax	INTEGER
Access	read-only
Description	An integer index for <code>tuxTQtransTbl</code> . 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 <code>Idx1</code> through <code>Idx5</code> .

## tuxTQtransIdx5

Syntax	INTEGER
Access	read-only
Description	An integer index for <code>tuxTQtransTbl</code> . 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 <code>Idx1</code> through <code>Idx5</code> .

## 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 <code>openinfo</code> string <code>tuxTgroupOpenInfo</code> contains the device name and queue space name for the queue space concerning this transaction.

## tuxTQtranSpaceName

Syntax	<i>DisplayString</i> (SIZE(1..15))
Access	read-only
Description	Name of the application queue space associated with the transaction.

## tuxTQtransQmConfig

Syntax	<i>DisplayString</i> (SIZE(1..78))
Access	read-only
Description	Absolute pathname of the file or device where the application queue space is located.

## tuxTQtransLmid

Syntax	<i>DisplayString</i> (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	<p>The values for GET and SET operations are as follows:</p> <p>GET: {actdive(1) abort-only(2) aborted(3) com-called(4) ready(5) decided(6) suspended(7)}</p> <p>A GET operation retrieves runtime information about the selected transactions. The following list describes the meaning of the tuxTQtransState attribute returned in response to a GET request. States not listed will not be returned.</p> <p>active(1) The transaction is active.</p> <p>abort-only(2) The transaction has been identified for rollback.</p> <p>aborted(3) The transaction has been identified for rollback and rollback has been initiated.</p> <p>com-called(4) The initiator of the transaction has called tpcommit(3) and the first phase of two-phase commit has begun.</p> <p>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.</p>

`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` attribute 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.



# 10 Event Broker MIB

TUXEDO events are of two types: 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 the TUXEDO system are detected. Application programs (clients or services) can subscribe to these system events.

The Event Broker MIB defines the characteristics of an event subscription. You can use the Event Broker 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 TUXEDO Core MIB.

Event subscriptions can be temporary or persistent. Persistent subscriptions survive across application activations and can be removed through the Event Broker MIB. The TUXEDO Event Broker MIB contains five groups of event subscriptions through which the Event Broker can be managed. The following table lists the event broker subscription groups.

Group Name	Description
<code>tuxEventClientTbl</code>	Subscriptions that trigger unsolicited notification
<code>tuxEventCmdTbl</code>	Subscriptions that trigger system commands
<code>tuxEventQueTbl</code>	Subscriptions for queue-based notification
<code>tuxEventSvcTbl</code>	Subscriptions for server-based notification
<code>tuxEventUlogTbl</code>	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 Event Broker 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 represents a set of subscriptions registered with the Event Broker 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`.

Variable 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
<code>tuxEventClientFilter</code>	.1.3.6.1.4.1.140.300.2.1.1.1.3
<code>tuxEventClientState</code>	.1.3.6.1.4.1.140.300.2.1.1.1.4
<code>tuxEventClientId</code>	.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	<i>DisplayString</i> (SIZE(1..255))
Access	read-only
Description	Event pattern expression. This expression, in <code>recomp(3)</code> format, controls which event names match this subscription.

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

### tuxEventClientFilter

Syntax	<i>DisplayString</i> (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: <p>GET: active(1) A GET operation will retrieve configuration information for the matching tuxEventClientTbl row(s).</p> <p>SET: invalid(2) A SET operation will update configuration information for the row in tuxEventClientTbl. The following state indicates the meaning of a tuxEventClientState set in a SET request. States not listed may not be set.</p> <p>invalid(2) Delete row. Successful return leaves the row in the invalid(2) state.</p>

### tuxEventClientId

Syntax	<i>DisplayString</i> (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 represents a set of subscriptions registered with the Event Broker 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.

Variable 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	<i>DisplayString</i> (SIZE(1..255))
Access	read-write
Description	Event pattern expression. This expression, in <i>recomp</i> (3) format, controls which event names match this subscription.

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

### **tuxEventCmdFilter**

Syntax	<i>DisplayString</i> (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	<p>The values for GET and SET operations are as follows:</p> <p>GET: active(1)  A GET operation will retrieve configuration information for the tuxEventCmdTbl instance(s).</p> <p>SET: invalid(2)  A SET operation will update configuration information for the tuxEventCmdTbl instance. The following state indicates the meaning of a tuxEventCmdState set in a SET request. States not listed may not be set.</p> <p>invalid(2)  Delete tuxEventCmdTbl instance. Successful return leaves the object in the invalid(2) state.</p>

## 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 represents a set of subscriptions registered with the Event Broker 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.

Variable Name	Object ID
tuxEventQueIndx	.1.3.6.1.4.1.140.300.2.3.1.1.1
tuxEventQueExpr	.1.3.6.1.4.1.140.300.2.3.1.1.2
tuxEventQueFilter	.1.3.6.1.4.1.140.300.2.3.1.1.3
tuxEventQueState	.1.3.6.1.4.1.140.300.2.3.1.1.4
tuxEventQspace	.1.3.6.1.4.1.140.300.2.3.1.1.5
tuxEventQname	.1.3.6.1.4.1.140.300.2.3.1.1.6
tuxEventQctlQtop	.1.3.6.1.4.1.140.300.2.3.1.1.7
tuxEventQctlBeforeMsgid	.1.3.6.1.4.1.140.300.2.3.1.1.8
tuxEventQctlQtimeAbs	.1.3.6.1.4.1.140.300.2.3.1.1.9
tuxEventQctlQtimeRel	.1.3.6.1.4.1.140.300.2.3.1.1.10
tuxEventQctlDeqTime	.1.3.6.1.4.1.140.300.2.3.1.1.11
tuxEventQctlPrior	.1.3.6.1.4.1.140.300.2.3.1.1.12
tuxEventQctlMsgId	.1.3.6.1.4.1.140.300.2.3.1.1.13
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



Variable Name	Object ID
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	<i>DisplayString</i> (SIZE(1..255))
Access	read-write
Description	Event pattern expression. This expression, in <code>recomp(3)</code> format, controls which event names match this subscription.

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

## tuxEventQueFilter

Syntax	<i>DisplayString</i> (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	<p>The values for GET and SET operations are as follows:</p> <p>GET: active(1) A GET operation will retrieve configuration information for the matching tuxEventQueTbl row(s).</p> <p>SET: invalid(2) A SET operation will update configuration information for the tuxEventQueTbl instance. The following state indicates the meaning of a tuxEventQueState set in a SET request. States not listed may not be set.</p> <p>invalid(2) Delete tuxEventQueTbl row. Successful return leaves the object in the invalid(2) state.</p>

### **tuxEventQspace**

Syntax	<i>DisplayString</i> (SIZE(1..15))
Access	read-write
Description	<p>Enqueue a notification message to a reliable queue in this queue space when a matching event is detected.</p> <p><b>Note:</b> This object can be updated only during row creation.</p>

### **tuxEventQname**

Syntax	<i>DisplayString</i> (SIZE(1..15))
Access	read-write
Description	<p>Enqueue a notification message to this reliable queue when a matching event is detected.</p> <p><b>Note:</b> This object can be updated only during row creation.</p>

## 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.

### **tuxEventQctlQtimeRel**

Syntax	INTEGER
Access	read-write
Description	This value, if present, is passed in to <code>tpenqueue(3)</code> '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.

### **tuxEventQctlDeqTime**

Syntax	INTEGER
Access	read-write
Description	This value, if present, is passed in to <code>tpenqueue(3)</code> 's TPQCTL control structure.

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

### **tuxEventQctlPrior**

Syntax	INTEGER
Access	read-write
Description	This value, if present, is passed in to <code>tpenqueue(3)</code> 's TPQCTL control structure.

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

### **tuxEventQctlMsgId**

Syntax	<i>DisplayString</i> (SIZE(1..31))
Access	read-write
Description	This value, if present, is passed in to <code>tpenqueue(3)</code> '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..15))

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.

## tuxEventQctlFailQ

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

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 <code>tppost(3)</code> call is transactional, include the <code>tpenqueue(3)</code> call in the client's transaction.

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

# tuxEventSvcTbl

This represents a set of subscriptions registered with the Event Broker 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`.

Variable Name	Object ID
tuxEventSvcIndx	.1.3.6.1.4.1.140.300.2.4.1.1.1
tuxEventSvcExpr	.1.3.6.1.4.1.140.300.2.4.1.1.2
tuxEventSvcFilter	.1.3.6.1.4.1.140.300.2.4.1.1.3
tuxEventSvcState	.1.3.6.1.4.1.140.300.2.4.1.1.4
tuxEventSvcName	.1.3.6.1.4.1.140.300.2.4.1.1.5
tuxEventSvcPersist	.1.3.6.1.4.1.140.300.2.4.1.1.6
tuxEventSvcTran	.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	<i>DisplayString</i> (SIZE(1..255))
Access	read-only
Description	Event pattern expression. This expression, in <i>recomp</i> (3) format, controls which event names match this subscription.

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

### tuxEventSvcFilter

Syntax	<i>DisplayString</i> (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	<p>The values for GET and SET operations are as follows:</p> <p>GET: active(1)  A GET operation will retrieve configuration information for the matching tuxEventSvcTbl instance(s).</p> <p>SET: invalid(2)  A SET operation will update configuration information for the tuxEventSvcTbl instance. The following state indicates the meaning of a tuxEventSvcState set in a SET request. States not listed may not be set.</p> <p>invalid(2)  Delete tuxEventSvcTbl row. Successful return leaves the object in the invalid(2) state.</p>



## tuxEventSvcName

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

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 represents a set of subscriptions registered with the Event Broker 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.

Variable Name	Object ID
tuxEventUlogIndx	.1.3.6.1.4.1.140.300.2.5.1.1.1
tuxEventUlogExpr	.1.3.6.1.4.1.140.300.2.5.1.1.2
tuxEventUlogFilter	.1.3.6.1.4.1.140.300.2.5.1.1.3
tuxEventUlogState	.1.3.6.1.4.1.140.300.2.5.1.1.4
tuxEventUserlog	.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	<code>DisplayString (SIZE(1..255))</code>
Access	read-write
Description	Event pattern expression. This expression, in <code>recomp(3)</code> format, controls which event names match this subscription.

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

## tuxEventUlogFilter

Syntax	<code>DisplayString (SIZE(1..255))</code>
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	<code>INTEGER { active(1), invalid(2) }</code>
Access	read-write
Description	<p>The values for GET and SET operations are as follows:</p> <p>GET: <code>active(1)</code> A GET operation will retrieve configuration information for the matching <code>tuxEventUlogTbl</code> instance(s).</p> <p>SET: <code>invalid(2)</code> A SET operation will update configuration information for the <code>tuxEventUlogTbl</code> instance. The following state indicates the meaning of a <code>tuxEventUlogState</code> set in a SET request. States not listed may not be set.</p> <p><code>invalid(2)</code> Delete <code>tuxEventUlogTbl</code> row. Successful return leaves the object in the <code>invalid(2)</code> state.</p>

### **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.

# 11 TUXEDO Traps MIB

The TUXEDO system event monitor feature detects and reports certain predefined events — primarily failures that a system operator should be aware of. The TUXEDO 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 the SNMP agent to receive TUXEDO system events, the TUXEDO system Event Broker (TMSYSEVT) must be running as that is the entity that generates the TUXEDO system events.

The Event Traps MIB defines all the traps which will be generated and the objects which are passed in the variable bindings for these traps. The cause and recommended action for each event is described in the following sections.

## Specific Trap Number

Each enterprise-specific trap notification generated by the SNMP agent has a value in the specific trap ID field of the SNMP trap packet which 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 the Agent Connection contain 12 variables (attribute/value pairs) in the variable bindings of the trap packet:

`beaEventsDomainId`

This is the ID of the domain which generated the TUXEDO event notification.

`beaEventsIpckey`

This is the IPC key of the TUXEDO domain.

`beaLogicalAgentName`

The logical agent name of the SNMP agent generating the trap. The executable name is the default logical agent name.

The `tuxEventTrapVars` group contains all objects which 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	<i>DisplayString</i> ( SIZE(1..30) )
Access	not-accessible
Description	A string identifying 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 will always be microseconds, the actual resolution depends on the underlying operating system and hardware.

## tuxEventsDescription

Syntax	<i>DisplayString</i>
Access	not-accessible
Description	A one-line string summarizing the event.

### tuxEventsClass

Syntax	<i>DisplayString</i>
Access	not-accessible
Description	The class of the object associated with the event. Depending on TA_CLASS, the event notification buffer will contain additional fields specific to an object of this class.

### tuxEventsUlogCat

Syntax	<i>DisplayString</i>
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	<i>DisplayString</i> (SIZE (0..30))
Access	not-accessible
Description	Domain identification string. Refer to Chapter 4, “TUXEDO Core MIB.”



## tuxTdomainKey

Syntax	INTEGER (32769..262143)
Access	not-accessible
Description	Numeric key for the well-known address in a TUXEDO System/T 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 4, “TUXEDO Core MIB.”

## beaLogicalAgentName

Syntax	<i>DisplayString</i>
Access	not-accessible
Description	<p>The logical name of the agent as provided in the <code>-l</code> option (service name in case of Windows NT) when the agent was started. This is the agent which is monitoring this domain. If there are multiple Agent Connection 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, <code>simp_snmpd</code> and <code>bank_snmpd</code>, the communities used to query values from these agents would be <code>public@simp_snmpd</code> and <code>public@bank_snmpd</code> respectively. The component after the @ sign is used to identify the agent to which the MIB query is to be sent.</p> <p>This object is passed in the variable binding of all SNMP traps generated on behalf of TUXEDO system events.</p>

**Note:** To run multiple Agent Connection agents on the same managed node, they must be started as sub-agents (without `-s` option) and run after starting the Agent Integrator.

# Trap Definitions

This section defines all the traps generated by the TUXEDO SNMP agent when system events occur.

## 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 is an informational message.
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 <code>.SysMachineBroadcast</code> occurs. It denotes broadcast delivery failure. This message indicates that <code>tpbroadcast()</code> 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 <code>.SysMachineConfig</code> occurs. It denotes a change in a particular machine configuration.
Action	This is an informational message.
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 <code>.SysMachineFullMaxgtt</code> 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 <code>.SysMachineFullMaxwsclients</code> 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 may 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 <code>.SysMachineSlow</code> 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 which helps the DBBL maintain the pulse of the system.
Action	<p>This can occur due to several reasons:</p> <ul style="list-style-type: none"> <li>◆ The entire network may be bogged down due to heavy traffic.</li> <li>◆ The BBL on the non-master may be either dead or slow.</li> <li>◆ The BRIDGE process on the non-master is extremely busy.</li> </ul> <p>This problem can be intermittent.</p>
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 <code>.SysMachineState</code> occurs. This denotes that a particular machine changed its state.
Action	This is an informational message.
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 <code>.SysNetworkConfig</code> occurs. This message indicates the network link between the two machines specified changed to a new state.
Action	This is an informational message.
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 <code>.SysNetworkDropped</code> 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 <code>.SysNetworkFailure</code> occurs. This indicates a network connection failure between <b>BRIDGE</b> processes.
Action	This can happen either because the <b>BRIDGE</b> 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 <code>.SysNetworkFlow</code> occurs. This message states that the virtual circuit between machines changed to a new state.
Action	This is an informational message.
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 it is restarted again.

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 it is restarted again.

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 server have been updated.

Action        This is an informational message.

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 it is restarted again.

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 <code>.SysServerInit</code> occurs. This message indicates that the server specified above failed in <code>tpsvrinit()</code> 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 <code>.SysServerMaxgen</code> occurs. This message indicates that the server died abnormally. Since this has been marked as a restartable server, it has been restarted <code>MAXGEN-1</code> times in the specified <code>GRACE</code> 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 it again.
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 <code>.SysServerRestarting</code> 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 it again.
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 <code>.SysServerState</code> occurs. This message indicates that the server changed state.
Action	This is an informational message.
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 `tpreturn(TPEXIT)` while the server was executing application-specific code.

Action    This is an informational message.

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 is an informational message.

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 <code>.SysClientDied</code> occurs. This message indicates that the client exited the application without doing a <code>tpterm()</code> . Normally, clients should do a <code>tpterm()</code> before exiting the application.
Action	This is an informational message.
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 <code>.SysClientSecurity</code> 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 is an informational message.

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 <code>.SysTransactionHeuristicCommit</code> 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 <code>.SysEventDelivery</code> 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

---

# Index

## A

- access control list
  - MIB groups for 7-1
  - what it is 7-1
- access control lists
  - retrieving and modifying values of 7-1
- access to managed objects
  - local or global 2-15
- access to TUXEDO System entities, of
  - groups 7-4
- active applications
  - runtime statistics by machine 4-74
- active transactions, attributes of
  - as MIB objects 4-130
- administrative processes on master machine,
  - deactivating or activating 4-37
- agent, for TUXEDO
  - starting 2-8
- agents 1-3
  - unsolicited messages from 1-3
- alarms 1-5
- application attributes
  - by machine 4-60
- application events 10-1
- application queues, attributes of
  - as MIB objects 4-87
- application queues, features of
  - as MIB objects 9-1
- application security, type of
  - as MIB object 4-46
- application users and domains, attributes of

- groups of 7-2
- applications, attributes of
  - for server group 4-54
- ASN.1 1-4
- attributes of domain
  - as MIB objects 4-32

## B

- BBL
  - See Also bulletin board* 4-47
- bea.asn1
  - default location of 3-1
- BEA\_SMUX\_PASSWD 2-10
- beaDomainId 5-2
- beaDomainKey 5-1
- beaDomainList 5-1
- beaDomainStatus 5-3
- beaDomainTuxconfig 5-3
- beaDomainTuxdir 5-2
- beaEvtAgentName 4-153
- beaEvtExpr 4-154
- beaEvtFilter 4-154
- beaEvtFilterid 4-153
- beaEvtFilterState 4-155
- beaEvtFilterTable 4-153
- beaLogicalAgentName 5-2
- beaRoutingType 4-95
- bridge, TUXEDO
  - MIB table 4-3
- bulletin board

---

- machine table 4-41
- number of queue table entries 4-49
- number of server table entries 4-49
- number of service table entries 4-50
- number of type table entries 4-50
- queue table 4-41
- routing criteria range table 4-41
- routing table entries 4-48
- server group table entries 4-48
- server table 4-42
- service table 4-42
- starting 4-63
- string pool table 4-42

## C

- client
  - network address of 4-20
- client, death of
  - trap 11-19
- client, elapsed time of
  - calculation of 4-19
- clientConfigTrap 11-18
- clientDiedTrap 11-19
- clients, attributes of
  - as MIB objects 4-10
- clientStateTrap 11-20
- community names
  - how specified 2-10
- compilation
  - of MIB on management platform 2-4
- conversations, attributes of
  - as MIB objects 4-23
- CORBA 6-4
- CORBA interfaces
  - configuration and runtime attributes as MIB objects 6-5

## D

- destination host

- for traps 2-10
- device identifier
  - for tuxTDeviceTbl 4-31
- device lists, TUXEDO System /T
  - files storing 4-28
- dip-in notification method
  - activating 4-43
- domain, global application attributes of
  - as MIB objects 4-32
- domains, multiple
  - retrieving or modifying values in 3-6
- domains, TUXEDO or M3
  - list of as MIB table 5-1
- domain-specific traps 11-6
- DTP transaction log size, by machine
  - as MIB object 4-69

## E

- encryption level 4-151
  - tuxTBridgeCurEncryptBits 4-8
- enterprise identifiers 3-7
- enterprise OID
  - as value of enterprise field in traps 1-10
- entry object in a table 1-10
- environment variables
  - LD\_LIBRARY\_PATH 2-2
  - PATH system variable 2-2
  - setting 2-2
- Event Broker subscriptions
  - for service-based notifications 10-15
  - for system commands 10-5
  - for writing system userlog messages 10-18
- event filters, TUXEDO
  - as MIB objects 4-153
- event subscription requests
  - as MIB objects 10-2
- eventDeliveryTrap 11-21
- eventFailureTrap 11-22
- events, TUXEDO

---

as SNMP traps 1-14

## F

factory-based routing specifications 4-92

## G

GET

specifying when managing multiple domains 3-6

GET request 1-4

GET-NEXT request 1-4

groups with access to TUXEDO System  
as MIB objects 7-4

## H

handler, workstation

*See WSH* 8-6

host

destination of traps 2-10

## I

installation

of TUXEDO or M3 SNMP agent 2-1

ISO 1-3

## L

LD\_LIBRARY\_PATH 2-2

LD\_LIBRARY\_PATH environment variable  
2-11

LIBPATH environment variable 2-2, 2-11

listener processes, attributes of  
as MIB objects 4-129

listener, workstation

*See WSL* 8-9

load balancing

as MIB object 4-43

local machine, statistics of

as MIB objects 4-74

logical agent name 5-2

logical machine identifier

where originator is accessing application  
4-25

logical machines

network group membership 4-149

logical machines, connectivity of

as MIB objects 4-3

## M

m3CurInterfaces 4-53

m3FactoryId 6-3

m3FactoryIfName 6-4

m3FactorySerNo 6-3

m3FactoryState 6-4

m3FactoryTable 6-3

m3HwInterfaces 4-53

m3IfAutoTran 6-9

effect of m3IfTxPolicy on 6-9, 6-18

m3IfFbRoutingName 6-11

m3IfLmid 6-11

m3IfLoad 6-10

m3IfName 6-7

m3IfNumServers 6-11

m3IfPrio 6-10

m3IfSerNo 6-6

m3IfSrvGrp 6-7

m3IfState 6-7

m3IfTimeout 6-10

m3IfTpPolicy 6-12

m3IfTranTime 6-11

m3IfTxPolicy 6-12

effect on m3IfAutoTran 6-9, 6-18

m3InterfaceInfo 6-5

m3LclIfNcompleted 6-14

m3LclIfNqueued 6-14

m3LclIfSerNo 6-13

m3LclInterfaceTable 6-13

m3LclSrvGrp 6-14

---

- m3MachineCurObjects 4-82
- m3MachineHwObjects 4-82
- m3MachineMaxObjects 4-73
- m3MaxInterfaces 4-52
- m3MaxObjects 4-52
- m3RoutingFieldType 4-96
- m3SrvrCurInterfaceExt 4-117
- m3SrvrCurObjsExt 4-117
- machineBroadcastTrap 11-7
- machineConfigTrap 11-7
- machineFullMaxAccessesTrap 11-8
- machineFullMaxConvTrap 11-8
- machineFullMaxGttTrap 11-9
- machineFullMaxWsClientsTrap 11-9
- machineMsgQTrap 11-10
- machinePartitionedTrap 11-10
- machineSlowTrap 11-11
- machineStateTrap 11-11
- managed objects
  - what they are 1-3
- managed resource
  - what it is 1-3
- Management Information Base
  - See MIB 1-3
- management of TUXEDO
  - preparing for 2-1
- management platforms
  - functions of 1-5
  - using with TUXEDO Agent Connection 1-13
- manager
  - event integration with 3-3
- master machine
  - deactivating or activating administrative processes on 4-36
- message queues, TUXEDO System/T
  - as MIB objects 4-83
- message, moving
  - via SNMP SET request 9-15
- messages, in queues
  - as MIB objects 9-12

- MIB
  - loading in management platform 2-4
  - role in system management 1-3
- MIB file, for TUXEDO
  - default location of 3-1
- MIB objects
  - values returned when nonexistent 1-12
- MIB, TUXEDO standard
  - See TMIB 1-11
- MIBs
  - how to use 1-12
- MIBs, for TUXEDO
  - list of 1-8

## N

- network address
  - of workstation clients 4-20
- network address of logical machine 4-70
- network groups, attributes of
  - as MIB objects 4-147
- network management platforms
  - role of 1-3
- networkConfigTrap 11-12
- networkDroppedTrap 11-12
- networkFailureTrap 11-13
- networkFlowTrap 11-13
- networkStateTrap 11-14
- NT, Windows
  - TUXEDO DLLs in PATH 2-11

## O

- object definitions
  - format of 1-9
- object identifier
  - of TUXEDO 1-10
- OpenView
  - event categories 3-6
  - event configuration 3-6
  - loading the BEA MIB 3-3

---

## P

- password
  - how specified by tux\_snmpd 2-10
- port, SMUX
  - specifying 2-9, 2-14
- port, SNMP
  - specifying 2-9, 2-13
- print
  - book xxxii
  - topic xxxii

## Q

- queue space, attributes of
  - as MIB objects 9-17
- queue space, state of
  - as MIB object 9-19
- queue-based notifications
  - subscriptions for 10-8
- queues
  - maximum number of in bulletin board
    - queue table 4-41
- queues, creating
  - via SNMP SET request 9-6
- queues, number of
  - as MIB object 9-23

## R

- request/response commands
  - in SNMP architecture 1-4
- resourceConfigTrap 11-6
- RFC 1155 1-4
- RFC 1157 1-3
- RFC 1212 1-8
- RFC 1215 1-10
- routing specifications, of applications
  - as MIB objects 4-92

## S

- security, application
  - as MIB object 4-46
- security, of application
  - as MIB object 4-46
- security, TUXEDO 7-1
- server group
  - application attributes 4-54
- server table, extension of 4-110
- server workload
  - as MIB object 4-114
- server, death of
  - trap 11-15
- serverCleaningTrap 11-14
- serverConfigTrap 11-15
- serverInitTrap 11-16
- serverMaxgenTrap 11-16
- serverRestartingTrap 11-17
- servers
  - maximum number of in bulletin board
    - server table 4-42
- servers in an application, attributes of
  - as MIB objects 4-97
- servers, administrative
  - starting 4-63
- servers, TUXEDO
  - traps specific to 11-14
- serverStateTrap 11-17
- serverTpExitTrap 11-18
- services
  - maximum number of in bulletin board
    - service table 4-42
- services in applications, attributes of
  - as MIB objects 4-118
- services within an application, attributes of
  - as MIB objects 4-123
- SET
  - specifying when managing multiple domains 3-6
- SET request 1-4

---

- SETs
  - dependent on states of TUXEDO 2-16
- severity, of system event 11-2
- SHLIB\_PATH 2-2
- SHLIB\_PATH environment variable 2-11
- Simple Network Management Protocol
  - See SNMP 1-3
- slow, response to DBBL
  - trap 11-11
- SNMP agent, for TUXEDO
  - starting 2-4
- SNMP architecture 1-3
- SNMP communities
  - how specified 2-10
- specific trap ID 1-10
- starting TUXEDO SNMP agent 2-4
- support
  - customer xxxv
  - documentation xxxv
- system events, TUXEDO
  - as trap notifications 1-5

## T

- tables
  - Entry object under 1-10
- timeout, of service requests
  - as MIB object 4-126
- TMIB 1-12
  - differences from SNMP MIBs 1-11
- transactions
  - traps specific to 11-20
- transactions, active
  - MIB table for 4-130
  - state of 4-132
- transactions, attributes of
  - as MIB objects 9-26
- transHeuristicAbortTrap 11-20
- transHeuristicCommitTrap 11-21
- trap notification
  - what it is 1-3

- traps
  - domain-specific 11-6
  - machine-specific 11-7
  - setting destination host, port, and community 2-10
  - TUXEDO bridge-specific 11-12
  - TUXEDO client-specific 11-18
  - TUXEDO event-specific 11-21
  - TUXEDO server-specific 11-14
  - TUXEDO transaction-specific 11-20
- traps, SNMP
  - generated by TUXEDO agent 11-1
- tusTdomainState 4-36
- TUXEDO
  - object identifier of 1-10
  - objects accessible only locally 2-15
  - shared libraries 2-11
- TUXEDO application, access to 7-6
- TUXEDO configuration file, by machine
  - as MIB object 4-62
- TUXEDO events
  - as SNMP traps 1-13
  - types of 10-1
- TUXEDO MIB
  - See *TMIB* 1-5
- TUXEDO resources
  - managing 1-13
- TUXEDO software pathname, by machine
  - as MIB object 4-62
- TUXEDO software, pathname to
  - as MIB object 5-2
- TUXEDO standard MIB
  - See *TMIB* 1-11
- TUXEDO System /T
  - listener processes 4-129
- TUXEDO system events 10-1
- TUXEDO, states of 2-16
- tuxEventClientExpr 10-3
- tuxEventClientFilter 10-4
- tuxEventClientId 10-4
- tuxEventClientIdx 10-3



---

tuxEventClientState 10-4	tuxEventSvcTbl 10-15
tuxEventClientTbl 10-3	tuxEventSvcTran 10-17
tuxEventCmd 10-7	tuxEventTran 10-14
tuxEventCmdExpr 10-6	tuxEventUlogExpr 10-19
tuxEventCmdFilter 10-6	tuxEventUlogFilter 10-19
tuxEventCmdIndx 10-5	tuxEventUlogIndx 10-18
tuxEventCmdState 10-6	tuxEventUlogState 10-19
tuxEventCmdTbl 10-5	tuxEventUlogTbl 10-18
tuxEventPersist 10-13	tuxEventUserLog 10-20
tuxEventQctlBeforeMsgid 10-11	tuxInternalIdx 4-96
tuxEventQctlCorrId 10-13	tuxTAclCltname 7-7
tuxEventQctlDeqTime 10-12	tuxTAclGrpId 7-2
tuxEventQctlFailQ 10-13	tuxTAclGrpName 7-2
tuxEventQctlMsgid 10-12	tuxTAclGrpState 7-3
tuxEventQctlPrior 10-12	tuxTAclGrpTable 7-2
tuxEventQctlQtimeAbs 10-11	tuxTAclPermGrpIds 7-5
tuxEventQctlQtimeRel 10-12	tuxTAclPermName 7-4
tuxEventQctlQtop 10-11	tuxTAclPermState 7-5
tuxEventQctlReplyQ 10-13	tuxTAclPermTable 7-4
tuxEventQname 10-10	tuxTAclPermType 7-4
tuxEventQSpace 10-10	tuxTAclPrinGrp 7-7
tuxEventQueExpr 10-9	tuxTAclPrinId 7-7
tuxEventQueFilter 10-9	tuxTAclPrinName 7-6
tuxEventQueIndx 10-9	tuxTAclPrinPasswd 7-7
tuxEventQueState 10-10	tuxTAclPrinState 7-8
tuxEventQueTbl 10-8	tuxTAclPrinTbl 7-6
tuxEventsClass 11-4	tuxTAppQcmd 9-9
tuxEventsDescription 11-3	tuxTAppQcmdHw 9-10
tuxEventsLmid 11-3	tuxTAppQcmdLw 9-10
tuxEventsName 11-2	tuxTAppQctrl 9-2
tuxEventsSeverity 11-2	tuxTAppQctrlLmid 9-2
tuxEventsTime 11-3	tuxTAppQctrlMsgEndTime 9-4
tuxEventsUlogCat 11-4	tuxTAppQctrlMsgHiPrio 9-4
tuxEventsUlogMsgNum 11-4	tuxTAppQctrlMsgLoPrio 9-4
tuxEventsUsec 11-3	tuxTAppQctrlMsgStartTime 9-5
tuxEventSvcExpr 10-16	tuxTAppQctrlQmConfig 9-3
tuxEventSvcFilter 10-16	tuxTAppQctrlQname 9-3
tuxEventSvcIndx 10-15	tuxTAppQctrlSpaceName 9-3
tuxEventSvcName 10-17	tuxTAppQcurBlocks 9-11
tuxEventSvcPersist 10-17	tuxTAppQcurMsg 9-11
tuxEventSvcState 10-16	tuxTAppQgrpNo 9-8

---

tuxTAppQlmid 9-8  
tuxTAppQmaxRetries 9-11  
tuxTAppQmConfig 9-7  
tuxTAppQmsgCorId 9-16  
tuxTAppQmsgCurRetries 9-16  
tuxTAppQmsgGrpNo 9-13  
tuxTAppQmsgId 9-13  
tuxTAppQmsgLmid 9-14  
tuxTAppQmsgNewQname 9-15  
tuxTAppQmsgPrior 9-15  
tuxTAppQmsgQmConfig 9-14  
tuxTAppQmsgQname 9-13  
tuxTAppQmsgQspaceName 9-14  
tuxTAppQmsgSerNo 9-13  
tuxTAppQmsgSize 9-16  
tuxTAppQmsgState 9-14  
tuxTAppQmsgTbl 9-12  
tuxTAppQmsgTime 9-15  
tuxTAppQname 9-7  
tuxTAppQorder 9-9  
tuxTAppQoutOfOrder 9-11  
tuxTAppQretryDelay 9-11  
tuxTAppQspaceName 9-7  
tuxTAppQstate 9-8  
tuxTAppQTbl 9-6  
tuxTBridgeConTime 4-6  
tuxTBridgeCurEncryptBits 4-8  
tuxTBridgeCurTime 4-6  
tuxTBridgeFlowCnt 4-8  
tuxTBridgeLmid 4-4  
tuxTBridgeNetworkGroupName 4-9  
tuxTBridgeNetworkGrpNo 4-8  
tuxTBridgeRcvdByte 4-7  
tuxTBridgeRcvdNum 4-7  
tuxTBridgeSentByte 4-7  
tuxTBridgeSentNum 4-8  
tuxTBridgeState 4-4  
tuxTBridgeSuspTime 4-7  
tuxTBridgeTbl 4-3  
tuxTclientBirthTime 4-14  
tuxTclientCCIntName 4-14

tuxTclientCmtRet 4-19  
tuxTclientCurConv 4-19  
tuxTclientCurReq 4-19  
tuxTclientCurTime 4-19  
tuxTclientId 4-22  
tuxTclientIdleTime 4-15  
tuxTclientLastGrp 4-20  
tuxTclientMachineId 4-14  
tuxTclientNaddr 4-20  
tuxTclientNotify 4-20  
tuxTclientNumConv 4-17  
tuxTclientNumDeque 4-17  
tuxTclientNumEnqueue 4-17  
tuxTclientNumPost 4-17  
tuxTclientNumReq 4-18  
tuxTclientNumSubscribe 4-18  
tuxTclientNumTran 4-18  
tuxTclientNumTranAbt 4-18  
tuxTclientNumTranCmt 4-18  
tuxTclientNumUnSol 4-20  
tuxTclientPid 4-15  
tuxTclientReg 4-14  
tuxTclientRelease 4-16  
tuxTclientRpid 4-21  
tuxTclientSrvGrp 4-15  
tuxTclientState 4-12  
tuxTclientTbl 4-10  
tuxTclientTimeLeft 4-21  
tuxTclientTimeStart 4-21  
tuxTclientTranLev 4-21  
tuxTclientUserName 4-15  
tuxTclientWsc 4-16  
tuxTclientWsh 4-16  
tuxTclientWshClientId 4-16  
tuxTclientWsProto 4-17  
tuxTconnClientId 4-25  
tuxTconnOgrpNo 4-25  
tuxTconnOlmid 4-25  
tuxTconnOpid 4-25  
tuxTconnOsndcnt 4-25  
tuxTconnOsrvid 4-26

---

tuxTconnSerNo 4-24	tuxTdomainHwServices 4-52
tuxTconnSgrpNo 4-26	tuxTdomainID 4-37
tuxTconnSlmid 4-26	tuxTdomainKey 4-34
tuxTconnSpid 4-26	tuxTdomainLoadBalance 4-43
tuxTconnSsndcnt 4-26	tuxTdomainMask 4-38
tuxTconnSsrvId 4-27	tuxTdomainMaster 4-35
tuxTconnState 4-24	tuxTdomainMaxAccessers 4-38
tuxTconnSvcName 4-24	tuxTdomainMaxACLgroups 4-42
tuxTconnTable 4-23	tuxTdomainMaxBufsType 4-39
tuxTdevCfgDev 4-29	tuxTdomainMaxBufType 4-40
tuxTdeviceName 4-29	tuxTdomainmaxConv 4-39
tuxTdeviceTbl 4-28	tuxTdomainMaxDRT 4-40
tuxTdevIndex 4-30	tuxTdomainMaxGroups 4-40
tuxTdevLmid 4-28	tuxTdomainMaxGTT 4-39
tuxTdevOffset 4-29	tuxTdomainMaxMachines 4-40
tuxTdevSize 4-29	tuxTdomainMaxQueues 4-41
tuxTdevState 4-30	tuxTdomainMaxRFT 4-41
tuxTdomain 4-32	tuxTdomainMaxRTData 4-41
tuxTdomainAuthsvc 4-46	tuxTdomainMaxServers 4-42
tuxTdomainBBLQuery 4-47	tuxTdomainMaxServices 4-42
tuxTdomainBlockTime 4-47	tuxTdomainMode 4-36
tuxTdomainCMTRET 4-43	tuxTdomainNotify 4-43
tuxTdomainCurDRT 4-48	tuxTdomainOptions 4-45
tuxTdomainCurGroups 4-48	tuxTdomainPerm 4-38
tuxTdomainCurMachines 4-49	tuxTdomainSanityScan 4-48
tuxTdomainCurQueues 4-49	tuxTdomainScanUnit 4-47
tuxTdomainCurRFT 4-49	tuxTdomainSecurity 4-46
tuxTdomainCurRTdata 4-49	tuxTdomainSignal 4-45
tuxTdomainCurServers 4-49	tuxTdomainSystemAccess 4-44
tuxTdomainCurServices 4-50	tuxTdomainUID 4-37
tuxTdomainCursType 4-50	tuxTgroupCloseInfo 4-58
tuxTdomainCurType 4-50	tuxTgroupCurLMID 4-58
tuxTdomainDBBLWait 4-48	tuxTgroupLMID 4-55
tuxTdomainGID 4-38	tuxTgroupName 4-54
tuxTdomainHwDRT 4-50	tuxTgroupNo 4-55
tuxTdomainHwGroups 4-50	tuxTgroupOpenInfo 4-58
tuxTdomainHwMachines 4-51	tuxTgroupState 4-56
tuxTdomainHwQueues 4-51	tuxTgroupTable 4-54
tuxTdomainHwRFT 4-51	tuxTgroupTMScount 4-59
tuxTdomainHwRTdata 4-51	tuxTgroupTMSname 4-59
tuxTdomainHwServers 4-51	tuxTlistenLmid 4-129

---

tuxTlistenState 4-129	tuxTmachineNumSubscribe 4-79
tuxTlistenTbl 4-129	tuxTmachineNumTran 4-79
tuxTmachineAcIcCacheAccess 4-81	tuxTmachineNumTranAbt 4-79
tuxTmachineAcIcCacheHits 4-81	tuxTmachineNumTranCmt 4-79
tuxTmachineAcIcFail 4-81	tuxTmachinePageSize 4-80
tuxTmachineActive 4-74	tuxTmachinePerm 4-66
tuxTmachineAppDir 4-63	tuxTmachinePmid 4-61
tuxTmachineBridge 4-70	tuxTmachineRelease 4-73
tuxTmachineCmpLimit 4-71	tuxTmachineRole 4-72
tuxTmachineCurAccessers 4-75	tuxTmachineSpinCount 4-72
tuxTmachineCurClients 4-76	tuxTmachineState 4-63
tuxTmachineCurConv 4-76	tuxTmachineSWrelease 4-80
tuxTmachineCurGTT 4-76	tuxTmachineTable 4-60
tuxTmachineCurLoad 4-76	tuxTmachineTlogName 4-69
tuxTmachineCurWsClients 4-76	tuxTmachineTlogSize 4-69
tuxTmachineEnvFile 4-65	tuxTmachineTmNetLoad 4-71
tuxTmachineGid 4-65	tuxTmachineTuxConfig 4-62
tuxTmachineHwAccessers 4-77	tuxTmachineTuxDir 4-62
tuxTmachineHwAcIcCache 4-80	tuxTmachineType 4-67
tuxTmachineHwClients 4-77	tuxTmachineUid 4-65
tuxTmachineHwConv 4-77	tuxTmachineUlogPfx 4-66
tuxTmachineHwGTT 4-77	tuxTmachineWkComplete 4-81
tuxTmachineHwWsClients 4-77	tuxTmachineWkInitiated 4-82
tuxTmachineLicExpires 4-79	tuxTmsgCbytes 4-84
tuxTmachineLicMaxUsers 4-80	tuxTmsgCtime 4-85
tuxTmachineLicSerial 4-80	tuxTmsgCurTime 4-84
tuxTmachineLmid 4-62	tuxTmsgId 4-83
tuxTmachineLogDevice 4-69	tuxTmsgLrPid 4-85
tuxTmachineMaxAccessers 4-67	tuxTmsgLsPid 4-85
tuxTmachineMaxActCache 4-68	tuxTmsgQbytes 4-85
tuxTmachineMaxConv 4-67	tuxTmsgQnum 4-85
tuxTmachineMaxGtt 4-68	tuxTmsgRtime 4-86
tuxTmachineMaxWsClients 4-68	tuxTmsgState 4-84
tuxTmachineMinor 4-72	tuxTmsgStime 4-86
tuxTmachineNaddr 4-70	tuxTmsgTable 4-83
tuxTmachineNlsaddr 4-70	tuxTnetGrpName 4-147
tuxTmachineNumConv 4-78	tuxTnetGrpNo 4-147
tuxTmachineNumDequeue 4-78	tuxTnetGrpPrio 4-148
tuxTmachineNumEnqueue 4-78	tuxTnetGrpState 4-148
tuxTmachineNumPost 4-78	tuxTnetGrpTbl 4-147
tuxTmachineNumReq 4-78	tuxTnetMapGrpName 4-149

---

tuxTnetMapGrpNo 4-150	tuxTQtransSpaceName 9-28
tuxTnetMapLmid 4-150	tuxTQtransState 9-29
tuxTnetMapMaxEncryptBit 4-151	tuxTQtransTbl 9-26
tuxTnetMapMinEncryptBit 4-151	tuxTQtransXid 9-26
tuxTnetMapNaddr 4-151	tuxTqueueNqueued 4-90
tuxTnetMapState 4-150	tuxTqueueRqAddr 4-87
tuxTnetMapTbl 4-149	tuxTqueueRqId 4-89
tuxTpTranId 4-131	tuxTqueueSource 4-90
tuxTQspaceBlocking 9-20	tuxTqueueSrvrCnt 4-89
tuxTQspaceCurExtent 9-23	tuxTqueueState 4-88
tuxTQspaceCurMsg 9-23	tuxTqueueTable 4-87
tuxTQspaceCurProc 9-23	tuxTqueueTotNqueued 4-89
tuxTQspaceCurQueues 9-23	tuxTqueueTotWkQueued 4-90
tuxTQspaceCurTrans 9-24	tuxTqueueWkQueued 4-91
tuxTQspaceErrQname 9-21	tuxTranCoordLmid 4-131
tuxTQspaceForceInit 9-21	tuxTranGrpCnt 4-134
tuxTQspaceGrpNo 9-19	tuxTranGrpIndex 4-134
tuxTQspaceHwMsg 9-24	tuxTranGrpNo 4-135
tuxTQspaceHwProc 9-24	tuxTranGstate 4-135
tuxTQspaceHwQueues 9-24	tuxTranIdx1 4-131
tuxTQspaceHwTrans 9-25	tuxTranIdx2 4-132
tuxTQspaceIpckey 9-21	tuxTranIdx3 4-132
tuxTQspaceLmid 9-19	tuxTranIdx4 4-132
tuxTQspaceMaxMsg 9-21	tuxTranIdx5 4-132
tuxTQspaceMaxPages 9-22	tuxTranState 4-132
tuxTQspaceMaxProc 9-22	tuxTranTbl 4-130
tuxTQspaceMaxQueues 9-22	tuxTranTimeOut 4-134
tuxTQspaceMaxTrans 9-22	tuxTranXid 4-131
tuxTQspaceName 9-18	tuxTroutingBufType 4-93
tuxTQspacePercentInit 9-25	tuxTroutingField 4-93
tuxTQspaceQmConfig 9-18	tuxTroutingName 4-93
tuxTQspaceState 9-19	tuxTroutingRanges 4-94
tuxTQspaceTbl 9-17	tuxTroutingState 4-95
tuxTQtransGrpNo 9-28	tuxTroutingTable 4-92
tuxTQtransIdx1 9-27	tuxTsrvrBaseSrvId 4-101
tuxTQtransIdx2 9-27	tuxTsrvrClOpt 4-101
tuxTQtransIdx3 9-27	tuxTsrvrCltLmid 4-114
tuxTQtransIdx4 9-27	tuxTsrvrClmPid 4-114
tuxTQtransIdx5 9-28	tuxTsrvrCltReply 4-114
tuxTQtransLmid 9-29	tuxTsrvrCmtRet 4-115
tuxTQtransQmConfig 9-28	tuxTsrvrConv 4-106

---

tuxTsrvrCurConv 4-115	tuxTsrvrTimeLeft 4-116
tuxTsrvrCurReq 4-115	tuxTsrvrTimeRestart 4-108
tuxTsrvrCurService 4-115	tuxTsrvrTimeStart 4-109
tuxTsrvrCurTime 4-116	tuxTsrvrTotReqC 4-113
tuxTsrvrEnvFile 4-102	tuxTsrvrTotWorkL 4-114
tuxTsrvrGeneration 4-107	tuxTsrvrTranAbt 4-113
tuxTsrvrGrace 4-102	tuxTsrvrTranCmt 4-113
tuxTsrvrGrp 4-98	tuxTsrvrTranLev 4-117
tuxTsrvrGrpExt 4-117	tuxTsvcAutoTran 4-120
tuxTsrvrGrpNo 4-99	tuxTsvcBufType 4-122
tuxTsrvrGrpNoExt 4-111	tuxTsvcGrp 4-123
tuxTsrvrId 4-98	tuxTsvcGrpAutoTran 4-126
tuxTsrvrIdExt 4-111	tuxTsvcGrpLoad 4-126
tuxTsrvrLastGrp 4-116	tuxTsvcGrpName 4-125
tuxTsrvrMax 4-103	tuxTsvcGrpNo 4-125
tuxTsrvrMaxgen 4-103	tuxTsvcGrpPrio 4-126
tuxTsrvrMin 4-104	tuxTsvcGrpState 4-125
tuxTsrvrName 4-99	tuxTsvcGrpSvcName 4-124
tuxTsrvrNumConv 4-112	tuxTsvcGrpSvcTimeOut 4-126
tuxTsrvrNumDeque 4-112	tuxTsvcGrpTranTime 4-127
tuxTsrvrNumEnque 4-112	tuxTsvcLoad 4-120
tuxTsrvrNumPost 4-112	tuxTsvcName 4-118
tuxTsrvrNumReq 4-112	tuxTsvcPrio 4-120
tuxTsrvrNumSubscribe 4-113	tuxTsvcrName 4-128
tuxTsrvrNumTran 4-113	tuxTsvcRoutingName 4-122
tuxTsrvrPid 4-107	tuxTsvcSrvrId 4-127
tuxTsrvrRcmd 4-104	tuxTsvcSrvrLmid 4-127
tuxTsrvrReplyQ 4-106	tuxTsvcSrvrNcompleted 4-128
tuxTsrvrRestart 4-104	tuxTsvcSrvrNqueued 4-128
tuxTsrvrRpid 4-108	tuxTsvcSrvrRqAddr 4-127
tuxTsrvrRpPerm 4-106	tuxTsvcState 4-119
tuxTsrvrRqAddr 4-106	tuxTsvcTbl 4-118
tuxTsrvrRqId 4-108	tuxTsvcTimeOut 4-121
tuxTsrvrRqPerm 4-107	tuxTsvcTranTime 4-121
tuxTsrvrSequence 4-105	tuxTsvcType 4-119
tuxTsrvrState 4-99	tuxTulogCat 4-141
tuxTsrvrStateExt 4-117	tuxTulogCatCtrl 4-145
tuxTsrvrSvcTimeOut 4-116	tuxTulogCtrl 4-142
tuxTsrvrSystemAccess 4-105	tuxTulogEndTimeCtrl 4-144
tuxTsrvrTbl 4-97	tuxTulogLine 4-140
tuxTsrvrTblExt 4-110	tuxTulogLineCtrl 4-144

---

tuxTulogLmid 4-139  
tuxTulogLmidCtrl 4-143  
tuxTulogMmDdDYy 4-139  
tuxTulogMmddyyCtrl 4-143  
tuxTulogMsg 4-140  
tuxTulogMsgCtrl 4-144  
tuxTulogMsgNum 4-141  
tuxTulogMsgNumCtrl 4-146  
tuxTulogPid 4-140  
tuxTulogPidCtrl 4-145  
tuxTulogPmid 4-139  
tuxTulogPmidCtrl 4-143  
tuxTulogPocNameCtrl 4-146  
tuxTulogProcName 4-141  
tuxTulogSerNo 4-139  
tuxTulogSeverity 4-141  
tuxTulogSeverityCtrl 4-145  
tuxTulogTable 4-138  
tuxTulogTime 4-139  
tuxTulogTimeCtrl 4-143  
tuxTulogTpTranId 4-140  
tuxTulogTptranIdCtrl 4-144  
tuxTulogXid 4-140  
tuxTulogXidCtrl 4-145  
tuxTwhichCfgDev 4-31  
tuxTwshTaActive 8-6  
tuxTwshTaClientId 8-3  
tuxTwshTaCurClients 8-5  
tuxTwshTaCurwork 8-7  
tuxTwshTaFlowcnt 8-7  
tuxTwshTaGrpNo 8-4  
tuxTwshTaHwClients 8-5  
tuxTwshTaLmid 8-4  
tuxTwshTaMultiplex 8-5  
tuxTwshTaNumblockQ 8-7  
tuxTwshTaPid 8-5  
tuxTwshTaRcvdByt 8-7  
tuxTwshTaRcvdNum 8-8  
tuxTwshTaSentByt 8-8  
tuxTwshTaSentNum 8-8  
tuxTwshTaSrvGrp 8-3

tuxTwshTaSrvId 8-4  
tuxTwshTaState 8-4  
tuxTwshTaTimeleft 8-6  
tuxTwshTaTotacttime 8-6  
tuxTwshTaTotidtime 8-6  
tuxTwshTaWshClientId 8-3  
tuxTwsITaCIOpt 8-14  
tuxTwsITaCurHandlers 8-16  
tuxTwsITaDevice 8-11  
tuxTwsITaEnvFile 8-14  
tuxTwsITaGrace 8-14  
tuxTwsITaGrpNo 8-11  
tuxTwsITaHwHandlers 8-16  
tuxTwsITaLmid 8-11  
tuxTwsITaMaxGen 8-15  
tuxTwsITaMaxHandlers 8-13  
tuxTwsITaMaxIdleTime 8-13  
tuxTwsITaMaxInitTime 8-13  
tuxTwsITaMinHandlers 8-12  
tuxTwsITaMultiplex 8-13  
tuxTwsITaNaddr 8-12  
tuxTwsITaPid 8-11  
tuxTwsITaRcmd 8-15  
tuxTwsITaRestart 8-15  
tuxTwsITaSequence 8-16  
tuxTwsITaSrvGrp 8-10  
tuxTwsITaSrvId 8-10  
tuxTwsITaState 8-11  
tuxTwsITaSuspended 8-17  
tuxTwsITaViewRefresh 8-17  
tuxTwsITaWshName 8-12  
tuxTwsITaWsProto 8-17  
tuxTwsITbl 8-9  
tuxWshTaNaddr 8-5

## U

unsolicited messages  
    from agents 1-3  
userlog messages  
    Event Broker subscriptions for 10-18

---

userlog table messages, control of 4-142  
userlogs, attributes of  
    as MIB objects 4-138

## **V**

variable bindings  
    contents of 1-10, 11-2  
variable bindings, trap  
    components of 11-2

## **W**

work station handler  
    *See WSH* 8-2  
workload, of server  
    as MIB object 4-114  
workstation client  
    status of 4-12  
WSH client processes, attributes of  
    as MIB objects 8-2  
WSL server processes, attributes of  
    as MIB objects 8-9