BEA Tuxedo®

New Features

Release 8.1
January 2003

Copyright

Restricted Rights Legend

Trademarks or Service Marks

Contents

About This Document

This document provides reference information on file formats, data descriptions, Management Information Bases (MIBs), and system processes for the BEA Tuxedo system. The reference pages are arranged in alphabetical order by the name of the file format, data description, MIB, or system process.

What You Need to Know

This document is intended for the following audiences:

- Administrators who are interested in configuring and managing applications in a BEA Tuxedo environment
- Application developers who are interested in programming applications in a BEA Tuxedo environment

This document assumes a familiarity with the BEA Tuxedo platform and either C or COBOL programming.

e-docs Web Site

BEA product documentation is available on the BEA corporate Web site. From the BEA Home page, click on Product Documentation or go directly to the "e-docs" Product Documentation page at http://e-docs.bea.com.

How to Print the Document

You can print a copy of this document from a Web browser, one file at a time, by using the File—>Print option on your Web browser.

A PDF version of this document is available on the BEA Tuxedo documentation Home page on the e-docs Web site (and also on the documentation CD). You can open the PDF in Adobe Acrobat Reader and print the entire document (or a portion of it) in book format. To access the PDFs, open the BEA Tuxedo documentation Home page, click the PDF files button and select the document you want to print.

If you do not have the Adobe Acrobat Reader, you can get it for free from the Adobe Web site at http://www.adobe.com.

Related Information

Related documents are listed in the See Also section of each reference page. For MIBs, related information is listed for the MIB as a whole rather than for each class.

Contact Us!

Your feedback on the BEA Tuxedo documentation is important to us. Send us e-mail at docsupport@bea.com if you have questions or comments. Your comments will be reviewed directly by the BEA professionals who create and update the BEA Tuxedo documentation.

In your e-mail message, please indicate that you are using the documentation for the BEA Tuxedo 8.1 release.

If you have any questions about this version of BEA Tuxedo, or if you have problems installing and running BEA Tuxedo, contact BEA Customer Support through BEA WebSupport at http://www.bea.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

Documentation Conventions

The following documentation conventions are used throughout this document.

Convention	Item
boldface text	Indicates terms defined in the glossary.
Ctrl+Tab	Indicates that you must press two or more keys simultaneously.
italics	Indicates emphasis or book titles.

Convention	Item

`monospace text`	Indicates code samples, commands and their options, data structures and their members, data types, directories, and filenames and their extensions. Monospace text also indicates text that you must enter from the keyboard.
	Examples:
	`#include <iostream.h> void main ( ) the pointer psz`
	`chmod u+w *`
	`\tux\data\ap`
	`.doc`
	`tux.doc`
	`BITMAP`
	`float`

`monospace boldface text`	Identifies significant words in code. *Example*: `void` **`commit`** `( )`
`monospace italic text`	Identifies variables in code. *Example*: `String` *`expr`*
UPPERCASE TEXT	Indicates device names, environment variables, and logical operators. *Example*s: LPT1 SIGNON OR
`{ }`	Indicates a set of choices in a syntax line. The braces themselves should never be typed.

`[ ]`	Indicates optional items in a syntax line. The brackets themselves should never be typed.
	Example:
	`buildobjclient [-v] [-o name ] [-f` *`file-list`*`]...`
	`[-l` *`file-list`*`]...`

`|`	Separates mutually exclusive choices in a syntax line. The symbol itself should never be typed.

Convention	Item

. . .	Indicates one of the following in a command line:
	■ That an argument can be repeated several times in a command line
	■ That the statement omits additional optional arguments
	■ That you can enter additional parameters, values, or other information
	The ellipsis itself should never be typed.
	Example:
	`buildobjclient [-v] [-o name ] [-f file-list]...`
	`[-l file-list]...`

. . .	Indicates the omission of items from a code example or from a syntax line. The vertical ellipsis itself should never be typed.

CHAPTER

1 New Features and Enhancements

The following sections describe the new features and enhancements offered in BEA Tuxedo release 8.1:

- Localized Install for Japan
- Multibyte Character Encoding
- XML Parser Integration
- Single Point Security Administration Option
- Domain Gateway Performance Improvement
- Remote Domain Connection Policy
- Domains Keepalive
- Multithreaded Bridge
- Parameter Length Expansion
- Global Maximum Transaction Timeout
- Enhanced CORBA C++ Client ORB

Localized Install for Japan

The following new installation enhancements enable customers to install and interface with the BEA Tuxedo system in English or Japanese:

- Upon startup, the installer program automatically chooses English or Japanese to be used for the installation based on an environment variable setting.
- During the installation, the installer program automatically installs an English-specific or Japanese-specific error message catalog based on the environment variable previously mentioned.

For more information about the new installation options, see *Installing the BEA Tuxedo System*.

Multibyte Character Encoding

The alphabetic characters in European languages, including standard English, can be accommodated with an 8-bit (single byte) character encoding scheme. Chinese, Japanese, and Korean languages, however, are based on a large set of symbols, or ideographs, that can be accommodated only with a multibyte character encoding scheme. The Multibyte character encoding feature supports the multibyte coded character sets required by Chinese, Japanese, Korean, and other Asian Pacific languages.

BEA Tuxedo 8.1 includes a new multibyte string data typed buffer named MBSTRING for transport of multibyte character user data. In addition, all Tuxedo typed buffers (FML, CARRAY, ..., MBSTRING) are now capable of carrying information identifying the code-set *character encoding*, or simply *encoding*, of their user data. For example, an MBSTRING buffer might indicate that its data is represented by the Shift-JIS (SJIS) encoding, while another MBSTRING buffer might indicate that its data is represented by the Extended UNIX Code (EUC) encoding.

Using the multibyte character encoding feature, the BEA Tuxedo system can convert user data from one encoding representation to another encoding representation when an MBSTRING buffer (or an `FLD_MBSTRING` field in an FML32 buffer) is transmitted

between processes running on different computer platforms. For example, upon receiving an MBSTRING buffer, the underlying Tuxedo system software might convert the encoding of the buffer's user data from SJIS to Japanese EUC. The conversion is neither a conversion between character code sets nor a translation between languages, but rather a conversion between different character encodings for the same language.

The encoding conversion capability enables an incoming message to be converted to an encoding representation supported by the machine on which the receiving process is running.

Multibyte Character Encoding Control

There are two ways of controlling code-set conversion:

- Administratively by using two environment variables named `TPMBENC` and `TPMBACONV`
- Programmatically by using APIs

Multibyte Character Encoding Support for GNU libiconv

GNU `libiconv`, an encoding conversion library that provides support for many coded character sets and encodings, is included with the BEA Tuxedo 8.1 software distribution. The multibyte character encoding feature uses the character conversion functions in this library to convert from any of the supported character encodings to any other supported character encoding, through Unicode conversion.

Note: GNU is a recursive acronym for *GNU's Not Unix*; it is pronounced "guh-New."

Multibyte Character Encoding Documentation

For more information about multibyte character encoding, see the following documents:

- Managing Typed Buffers in *Programming a BEA Tuxedo ATMI Application Using C*
- *Programming a BEA Tuxedo ATMI Application Using FML*
- *BEA Tuxedo ATMI FML Function Reference*
- `buffer(3c)`, `tpconvmb(3c)`, `tpgetmbenc(3c)`, `tpsetmbenc(3c)`, `tuxgetmbenc(3c)`, and `tuxsetmbenc(3c)`, `tuxgetmbaconv(3c)`, and `tuxsetmbaconv(3c)` in *BEA Tuxedo ATMI C Function Reference*
- `tuxenv(5)`, `tuxtypes(5)`, and `typesw(5)` in *BEA Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*

XML Parser Integration

This feature is the incorporation of the Apache Xerces C++ Version 1.7 parser into the BEA Tuxedo system for use by customer applications to read and write extensible markup language (XML) data. As XML continues to gain acceptance as a data standard, BEA Tuxedo customers are increasingly using XML typed buffers in their applications.

The Xerces-C++ 1.7 parser, named after the Xerces Blue butterfly and written in a portable subset of C++, comes with a shared library for parsing, generating, manipulating, and validating XML documents. It complies with the XML 1.0 recommendation and associated standards DOM 1.0, DOM 2.0. SAX 1.0, SAX 2.0, Namespaces, and W3C's XML Schema recommendation version 1.0.

Because the Xerces-C++ 1.7 parser does not cache the Document Type Definition (DTD) and XML schema files when validation is required, or cache external entity files used in DTD, the BEA Tuxedo developers improved the performance of the Xerces-C++ 1.7 parser by adding an option to cache external DTD, schema, and entity files that might otherwise be retrieved repeatedly over the Web.

XML Parser Control

There are two ways to turn on/off caching for the Xerces-C++ 1.7 parser:

- Administratively by using two environment variable named `URLENTITYCACHING` and `URLENTITYCACHEDIR`
- Programmatically by using APIs

XML Parser Support for ICU

The International Components for Unicode (ICU) 2.1 library, a C/C++ library that supports over 200 different coded character sets (encoding forms) on a wide variety of platforms, is included with the BEA Tuxedo 8.1 distribution. The Xerces-C++ 1.7 parser is built with the ICU 2.1 library.

XML Parser Sample Application

A sample application for using the Xerces-C++ 1.7 parser APIs is provided in the BEA Tuxedo user documentation. Among other things, the sample demonstrates how to write a wrapper for the Xerces-C++ 1.7 parser so that Tuxedo clients and servers written in C can call the Xerces-C++ APIs.

XML Parser Documentation

For more information about XML parser integration, see the following documents:

- To be determined
- *"Tutorial for xmlstockapp, a C and C++ XML Parser Application"* in *Tutorials for Developing BEA Tuxedo ATMI Applications*
- `getURLEntityCacheDir(3c)`, `setURLEntityCacheDir(3c)`, `getURLEntityCaching(3c)`, and `setURLEntityCaching(3c)` in *BEA Tuxedo ATMI C Function Reference*
- `tuxenv(5)` in *BEA Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*

Single Point Security Administration Option

This feature allows BEA Tuxedo and BEA WebLogic Server applications to share the WebLogic Server security database, thereby strengthening the integration of Tuxedo with WebLogic Server. This feature applies only to deployments involving a Tuxedo 8.1 application and a WebLogic Server 7.0 or later application.

With the availability of single point security administration, three security configurations are possible:

- Tuxedo and WebLogic Server security using a single user security database—A Tuxedo customer administers security for both Tuxedo and WebLogic Server using a single point of administration, *the WebLogic Server Administration Console*, and a single point of record, *the WebLogic Server user security database*. This security configuration supports a single sign-on to Tuxedo and WebLogic Server applications.
- Tuxedo and WebLogic Server security using two user security databases—A Tuxedo customer administers security for both Tuxedo and WebLogic Server using two points of administration, *the Tuxedo Administration Console and the WebLogic Server Administration Console*, and two points of record, *the Tuxedo user security database and the WebLogic Server user security database*.

- Tuxedo-only security—A Tuxedo customer administers security using the Tuxedo Administration Console and the Tuxedo user security database.

In BEA Tuxedo 8.1, single point security administration is limited to user authentication. In subsequent Tuxedo releases, single point security administration will be extended to include both user authentication and user authorization.

Single Point Security Administration Components

The single point security administration feature is based on the following components:

- Tuxedo LDAP authentication server—a Tuxedo 8.1 system authentication server named `LAUTHSRV`
- WebLogic Server embedded LDAP server—the WebLogic Server 7.0 or later system security server
- Tuxedo default authentication plug-in—a modular component
- WebLogic Server default authentication security provider—a modular component

LAUTHSRV Configuration File

The `LAUTHSRV` server is controlled by a new configuration file named `tpldap` located in the *`tux_prod_dir`*`\udataobj` directory, where *`tux_prod_dir`* represents the directory in which the BEA Tuxedo 8.1 distribution is installed. This configuration file is a text file containing keywords that can be edited using any text editor.

User Security Information Migration Utility

An administer can use the Tuxedo `tpmigldap(1)` command to migrate the content of the local Tuxedo user (`tpusr`) and group (`tpgrp`) security files to the WebLogic Server user security database. This command uses the LDAP protocol to contact the WebLogic Server embedded LDAP server and to transfer the Tuxedo user security information to WebLogic Server.

Single Point Security Administration Setup

To enable single point security administration for a Tuxedo application, an administrator sets the following configurations in the Tuxedo configuration (UBBCONFIG) file:

- Sets the SECURITY parameter in the RESOURCES section of the Tuxedo configuration file to USER_AUTH, ACL, or MANDATORY_ACL
- Configures the LAUTHSRV server in the SERVERS section of the Tuxedo configuration file

When a Tuxedo application is configured for single point security administration, the LAUTHSRV server accesses the user security information stored in the WebLogic Server embedded LDAP server to authenticate Tuxedo users who want to join the Tuxedo application and/or the associated WebLogic Server application. If caching is specified in the tpldap configuration file, the LAUTHSRV server caches the user security information locally for improved performance.

Compatibility with Traditional Tuxedo Security

The use of the traditional Tuxedo AUTHSVR authentication server is not affected by the single point security administration feature: Tuxedo administrators and designers can continue to use AUTHSVR or their own custom implementation of Tuxedo authentication server. If administrators or designers want to integrate Tuxedo user security information with WebLogic Server, they simply need to configure the new Tuxedo LAUTHSVR authentication server in the Tuxedo UBBCONFIG file.

Additionally, the ACL portion of traditional Tuxedo security is not affected by the single point security administration feature, and the traditional SECURITY values—NONE, APP_PW, USER_AUTH, ACL, MANDATORY_ACL—in the Tuxedo UBBCONFIG file continue to be supported.

Single Point Security Administration Documentation

For more information about single point security administration, see the following documents:

- "Implementing Single Point Security Administration" in *Using Security in ATMI Applications*
- tpmigldap(1) in *BEA Tuxedo Command Reference*
- LAUTHSRV(5) in *BEA Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*
- *Introduction to WebLogic Security* at http://e-docs.bea.com/wls/docs70/secintro/index.html

Domain Gateway Performance Improvement

The performance of the TDomain gateway has been improved by (1) improving various internal algorithms for scheduler and message receivers and (2) using cache whenever feasible to look up configuration information. The performance improvement requires no changes in the user interface.

Customers with services that span large numbers of domains are expected to benefit significantly by this performance improvement. In addition, because the TDomain gateway communicates with the WebLogic Tuxedo Connector (WTC) gateway in a Tuxedo/ WebLogic Server deployment, this feature also improves the performance of the WTC gateway and thereby strengthens the integration of Tuxedo with WebLogic Server.

Remote Domain Connection Policy

This feature changes the behavior of the ON_STARTUP type connection policy, used by the TDomain gateway, to allow users to selectively establish connections on a per remote domain basis. With previous releases of BEA Tuxedo, if the Domains connection policy is set to ON_STARTUP in the DM_LOCAL section—also known as the DM_LOCAL_DOMAINS section—of the Tuxedo Domains configuration (DMCONFIG) file,

the TDomain gateway tries to connect to all remote domains at boot time, even if some of the remote domains will not be used initially. With a large number of remote domains, the boot time may be substantial.

With BEA Tuxedo 8.1 comes the addition of a CONNECTION_POLICY optional parameter in the DM_TDOMAIN section of the DMCONFIG file, which allows users to specify the connection policy—LOCAL, ON_DEMAND, ON_STARTUP, INCOMING_ONLY—on a per local or per remote domain basis. The objective of this feature is to speed up boot time.

How the Remote Domain Connection Policy Works

Here is an example of how the new remote domain connection policy works.

1. In the DM_LOCAL or DM_TDOMAIN section of the Tuxedo DMCONFIG file, a user specifies the connection policy for the local domain. For example:

```
*DM_LOCAL
LOCAL1  GWGRP=GWTGROUP
        TYPE=TDOMAIN
        ACCESSPOINTID="BA.CENTRAL01"
        CONNECTION_POLICY=ON_STARTUP
```

   -OR-

```
*DM_TDOMAIN
LOCAL1  NWADDR="//albany.acme.com:4051"
        CONNECTION_POLICY=ON_STARTUP
```

2. In the DM_TDOMAIN section of the Tuxedo DMCONFIG file, a user specifies the connection policy for the remote domains. For example:

```
*DM_TDOMAIN
REMOT1  NWADDR="//newyork.acme.com:65431"
        CONNECTION_POLICY=ON_DEMAND
REMOT2  NWADDR="//philly.acme.com:65431"
```

The connection policy specified for the remote domain takes precedence over the connection policy specified for the local domain. So, in the preceding example, the connection policy configurations will be:

LOCAL1 to REMOT1 — ON_DEMAND
LOCAL1 to REMOT2 — ON_STARTUP

With the new remote domain connection policy, a user can specify which remote domain connections *not* to establish at boot time by specifying a connection policy of `ON_DEMAND` or `INCOMING_ONLY` for those remote domains.

Remote Domain Connection Policy Documentation

For more information about the remote domain connection policy, see the following documentation:

- *"About Domains"* in *Using the BEA Tuxedo Domains Component*
- `CONNECTION_POLICY`, `MAXRETRY`, and `RETRY_INTERVAL` parameters in the `DM_TDOMAIN` section of `DMCONFIG(5)` in *BEA Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*

Domains Keepalive

Most domains span across firewalls, and firewalls typically time out idle connections. Not only will the Domains keepalive feature keep Tuxedo interdomain connections open during periods of inactivity, but it will also enable TDomain gateways to quickly detect interdomain connection failures. Currently, a TDomain gateway learns of an interdomain connection failure through the underlying TCP stack, which may report the failure 15 minutes or more (depending on the local operating system configuration) after the failure occurs.

Domains keepalive allows users to enable and configure a keepalive protocol at the TCP level and/or application level for each TDomain gateway connection. TCP-level keepalive and application-level keepalive are not mutually exclusive, meaning that you can configure a Domains connection using both options.

The following table provides more details about Domains keepalive.

Table 1-1 About Domains Keepalive

Level	Interoperate With Older Tuxedo Release?	Individual Timer?	Quicker Connection Failure Detection?	Keepalive Event With Firewall?
TCP-Level Keepalive	Yes	No	Yes *	Yes
Application-Level Keepalive	No	Yes	Yes	Yes

* For TCP-level keepalive to quickly detect a TDomain gateway connection failure, it must be set to a small time interval. Doing so may flood the network with TCP packets.

For the Domains keepalive feature, the TCP-level keepalive option is named `TCPKEEPALIVE`, and the application-level keepalive option is named `DMKEEPALIVE`. These parameters, as well as another parameter named `DMKEEPALIVEWAIT`, have been added as optional parameters in the `DM_TDOMAIN` section of the Tuxedo `DMCONFIG` file. They allow users to configure Domains keepalive on a per local or per remote domain basis. The Domains keepalive value specified for a remote domain takes precedence over the Domains keepalive value specified for the local domain.

Keepalive Compatibility with Earlier Releases

Domains TCP-level keepalive is compatible with BEA Tuxedo 8.0 or earlier software. The BEA Tuxedo software running at the other end of the TCP connection may be any release of BEA Tuxedo because Domains TCP-level keepalive is executed at the network transport (TCP) layer.

Domains application-level keepalive is not compatible with BEA Tuxedo 8.0 or earlier software. The BEA Tuxedo software running at the other end of the TCP connection must be BEA Tuxedo 8.1 to be able to understand an application-level keepalive message. When connected to a TDomain gateway running an earlier release of BEA Tuxedo software, the TDomain gateway does *not* send an application-level keepalive message; instead, it logs a warning message in the local user log (ULOG) stating that the remote domain is running an earlier release of BEA Tuxedo software and does not support Domains application-level keepalive.

Domains Keepalive Documentation

For more information about Domains keepalive, see the following documentation:

- "About Domains" in *Using the BEA Tuxedo Domains Component*
- TCPKEEPALIVE, DMKEEPALIVE, and DMKEEPALIVEWAIT parameters in the DM_TDOMAIN section of DMCONFIG(5) in *BEA Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*

Multithreaded Bridge

This feature introduces a Tuxedo Bridge server process that has both single-threaded and multithreaded execution capabilities. A configuration setting determines whether the Bridge process is configured for single-threaded execution or multithreaded execution.

Because only one Bridge process is running per host machine in a multiple-machine Tuxedo domain, all traffic from a host machine passes through a single Bridge process to all other host machines in the domain. This architecture implies that after a threshold level of load, the Bridge process becomes the bottleneck for Tuxedo performance.

Prior to the BEA Tuxedo 8.1 release, even with multiple network links between machines, data throughput increased only slightly, mainly because the Bridge process was single threaded. Thus, changing the Bridge to a multithreaded execution process seemed to be a natural solution to improving data throughput.

BRTHREADS Configuration Parameter

The execution mode of the Bridge process is controlled by a new configuration parameter named BRTHREADS, which has been added as an optional parameter in the MACHINES section of the Tuxedo UBBCONFIG file. Setting this parameter to Y (yes) configures the Bridge process for multithreaded execution; setting this parameter to N (no, the default) configures the Bridge process for single-threaded execution.

Configurations with BRTHREADS set to "Y" on the local machine and BRTHREADS set (or defaulted) to "N" on the remote machine are allowed, but the throughput between the machines will not be greater than that for the single-threaded Bridge process.

Other important details pertaining to the BRTHREADS parameter are as follows:

- Setting BRTHREADS to Y makes sense only if a machine has multiple CPUs. However, having multiple CPUs is not a prerequisite for setting BRTHREADS to Y.
- If the MODEL parameter in the RESOURCES section of the UBBCONFIG file is set to SHM, the BRTHREADS parameter has no effect and is ignored.
- In a Tuxedo multiple-machine domain, setting BRTHREADS to Y for a machine running BEA Tuxedo pre-release 8.1 software has no effect and is ignored.
- If BRTHREADS=Y and the Bridge environment contains TMNOTHREADS=Y, the Bridge starts up in threaded mode and logs a warning message to the effect that the Bridge is ignoring the TMNOTHREADS setting. The TMNOTHREADS environment variable was added to the BEA Tuxedo product in release 8.0.

Multithreaded Bridge Compatibility with Earlier Releases

A Bridge process configured for single-threaded or multithreaded execution can interoperate with a Bridge process running in an earlier release of BEA Tuxedo or WebLogic Enterprise: BEA Tuxedo release 8.0 or earlier, WebLogic Enterprise release 5.1 or earlier. In general, a threaded Bridge can interoperate with an unthreaded Bridge because there are no external functional or behavioral changes due to the threading.

Multithreaded Bridge Documentation

For more information about the multithreaded bridge, see the BRTHREADS parameter in the MACHINES section of UBBCONFIG(5) in *BEA Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*.

Parameter Length Expansion

This enhancement increases the maximum allowable length of certain BEA Tuxedo configuration parameters from 64 or 78 characters to 256 characters. The main purpose of this enhancement is to increase the maximum length of pathname-holding parameters so that they can hold longer pathname strings.

The following parameters are affected by this enhancement:

- In the UBBCONFIG file:
  - AOUT (in SERVERS section only)
  - APPDIR
  - ENVFILE
  - FADDR
  - NADDR
  - NLSADDR
  - RCMD
  - TLOGDEVICE
  - TMSNAME
  - TUXCONFIG
  - TUXDIR
  - ULOGPFX
- In the DMCONFIG file:
  - AUDITLOG
  - DMTLOGDEV
  - NWADDR

In addition, the maximum allowable length of the following parameters in the T_APPQ class of APPQ_MIB(5) has been increased from 78 bytes to 127 bytes:

- TA_CMD
- TA_CMDNONPERSIST

And lastly, the maximum allowable length of the comparable parameters in the SNMP Agent MIB for BEA Tuxedo 8.1 has been increased to 256 or 127 bytes.

Common String Pool

The parameter length expansion enhancement is implemented as a common string pool in the BEA Tuxedo bulletin board. The string pool implementation saves memory because the bulletin board does not need to allocate all the memory required to hold every possible 256-byte capable string.

Note: Developers may include long strings for parameters *other than those previously mentioned* as part of the common string pool. Developers should also ensure that any fixed-size buffers in the `userlog(3c)` function are large enough to accommodate messages involving long strings.

The creation of a common string pool avoids any significant increase in the size of the bulletin board for applications that do not make use of the parameter length expansion enhancement. The common string pool is separate from the string pool currently used in the bulletin board for data-dependent routing.

Of the `UBBCONFIG` parameters whose maximum allowable length has been increased to 256 bytes, only the `GROUPS` section `TMSNAME` parameter and the `SERVERS` section `AOUT` and `RCMD` parameters are actually stored in the bulletin board. The others are read in at process startup time and stored in process memory.

MAXSPDATA Configuration Parameter

A new configuration parameter named `MAXSPDATA` has been added as an optional parameter in the `RESOURCES` section of the Tuxedo `UBBCONFIG` file. This parameter is used to change the maximum string pool space to be allocated in the bulletin board for the common string pool. Its value, in bytes, must be greater than or equal to 0 and less than or equal to 2147483640. The default is 0.

In most cases, accepting the default for this parameter will result in the BEA Tuxedo system allocating sufficient string pool space for the parameter strings whose maximum allowed length has been increased to 256 bytes. For applications for which

extensive dynamic configuration is anticipated (for example, anticipating the addition of six more machines to a BEA Tuxedo application), administrators can use the MAXSPDATA parameter to increase the size of the common string pool.

Adjusting the size of the common string pool has no effect on the size of the routing string pool controlled by the MAXRTDATA parameter.

Parameter Length Expansion Compatibility with Earlier Releases

When the TUXCONFIG file (binary equivalent of the UBBCONFIG file) is unloaded using the tmunloadcf(1) command, for transport to a machine running BEA Tuxedo 8.0 or earlier software, the tmunloadcf(1) command silently truncates any parameters previously restricted to *n* bytes long *that are longer than n bytes* to the first *n*-3 characters followed by 3 dots (...). A configuration file with parameters truncated in this manner will load successfully on any other machine in the Tuxedo domain *except* when the truncated parameters are in the MACHINES section of the configuration file and pertain to the machine loading the file.

The BDMCONFIG file (binary equivalent of the DMCONFIG file) is typically not shipped from machine to machine in a Tuxedo domain, so executing the dmunloadcf(1) command does not truncate any configuration parameters.

Parameter Length Expansion Documentation

For more information about parameter length expansion, see the MAXSPDATA parameter in the RESOURCES section of UBBCONFIG(5) in *BEA Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*.

Global Maximum Transaction Timeout

This enhancement adds a global maximum transaction timeout parameter to cap ATMI transaction timeout values that are excessively long. Without a global maximum transaction timeout, aborted transactions tend to build up in the global transaction table (GTT), which may ultimately result in no free slots in the GTT. A full GTT means that no further transactions are possible for an application.

In any of the following situations, the global maximum transaction timeout value will become the transaction's timeout value:

- The transaction timeout value passed in a `tpbegin()` call is greater that the global maximum transaction timeout value.
- No transaction timeout value is passed in a `tpbegin()` call—current default transaction timeout value is 68+ years.
- The `TRANTIME` timeout value specified in the `SERVICES` section of the `UBBCONFIG` file is greater than the global maximum transaction timeout value.

MAXTRANTIME Configuration Parameter

A global maximum transaction timeout parameter named `MAXSPDATA` has been added as an optional parameter in the `RESOURCES` section of the Tuxedo `UBBCONFIG` file. The `MAXTRANTIME` parameter has been added as an optional parameter in the `RESOURCES` section of the Tuxedo `UBBCONFIG` file. Its value, in seconds, must be greater than or equal to 0 and less than or equal to 2147483647. The default is 0, which indicates that no global transaction timeout limit is in effect.

`MAXTRANTIME` has no effect on a transaction started on a machine running BEA Tuxedo 8.0 or earlier, except that when a machine running BEA Tuxedo 8.1 is infected by the transaction, the transaction timeout value is capped—reduced if necessary—to the `MAXTRANTIME` value configured for that node.

MAXTRANTIME Compatibility with Earlier Releases

Administrators of existing BEA Tuxedo applications can implement global maximum transaction timeout without changing the source code for their existing application programs.

MAXTRANTIME Documentation

For more information about global maximum transaction timeout, see the MAXTRANTIME parameter in the RESOURCES section of UBBCONFIG(5) in *BEA Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*.

Enhanced CORBA C++ Client ORB

This feature enables Tuxedo CORBA C++ clients to participate in global transactions with WebLogic Server application servers in the same way that WebLogic Server T3 clients do.