BEA Tuxedo®

BEA Tuxedo のファイル形式とデータ記述方法

BEA Tuxedo 8.1
2003 年 1 月

Copyright

Restricted Rights Legend

Trademarks or Service Marks

目次

このマニュアルについて

このマニュアルでは、BEA Tuxedo システムのファイル形式、データ記述方法、管理情報ベース (MIB)、およびシステム・プロセスに関するリファレンス情報を提供します。このリファレンス・ページでは、ファイル形式、データ記述方法、MIB、システム・プロセスをアルファベット順に掲載しています。

対象読者

このマニュアルは、次のような読者を対象としています。

- BEA Tuxedo 環境でアプリケーションをコンフィギュレーションおよび管理するシステム管理者
- BEA Tuxedo 環境でアプリケーションをプログラミングするアプリケーション開発者

このマニュアルは、BEA Tuxedo プラットフォーム、および C 言語または COBOL 言語のプログラミングの知識があることを前提としています。

e-docs Web サイト

BEA 製品のマニュアルは BEA 社の Web サイト上で参照することができます。BEA ホーム・ページの [製品のドキュメント] をクリックするか、または http://edocs.beasys.co.jp/e-docs/index.html に直接アクセスしてください。

マニュアルの印刷方法

このマニュアルは、ご使用の Web ブラウザで一度に 1 ファイルずつ印刷できます。Web ブラウザの [ファイル] メニューにある [印刷] オプションを使用してください。

このマニュアルの PDF 版は、e-docs Web サイトの BEA Tuxedo マニュアル・ページから入手できます。また、マニュアルの CD-ROM にも収められています。この PDF を Adobe Acrobat Reader で開くと、マニュアル全体または一部をブック形式で印刷できます。PDF 形式を利用するには、BEA Tuxedo マニュアル・ページの [PDF 版] ボタンをクリックして、印刷するマニュアルを選択します。

Adobe Acrobat Reader は、Adobe 社の Web サイト (http://www.adobe.co.jp/) から入手できます。

関連情報

関連マニュアルは、各リファレンス・ページの「関連項目」の項に列挙されています。MIB については、各クラスではなく MIB 全体としての関連情報が列挙されています。

サポート情報

皆様の BEA Tuxedo マニュアルに対するフィードバックをお待ちしています。ご意見やご質問がありましたら、電子メールで docsupport-jp@bea.com までお送りください。お寄せいただきましたご意見は、BEA Tuxedo マニュアルの作成および改訂を担当する BEA 社のスタッフが直接検討いたします。

電子メール メッセージには、BEA Tuxedo 8.1 リリースのマニュアルを使用していることを明記してください。

BEA Tuxedo に関するご質問、または BEA Tuxedo のインストールや使用に際して問題が発生した場合は、http://www.bea.com の BEA WebSupport を通して BEA カスタマ・サポートにお問い合わせください。カスタマ・サポートへの問い合わせ方法は、製品パッケージに同梱されている カスタマ・サポート・カードにも記載されています。

カスタマ・サポートへお問い合わせの際には、以下の情報をご用意ください。

- お客様のお名前、電子メール・アドレス、電話番号、Fax 番号
- お客様の会社名と会社の住所
- ご使用のマシンの機種と認証コード
- ご使用の製品名とバージョン
- 問題の説明と関連するエラー・メッセージの内容

表記上の規則

このマニュアルでは、以下の表記規則が使用されています。

規則	項目
太字	用語集に定義されている用語を示します。
Ctrl + Tab	2 つ以上のキーを同時に押す操作を示します。
イタリック体	強調またはマニュアルのタイトルを示します。

等幅テキスト	コード・サンプル、コマンドとオプション、データ構造とメンバ、データ型、ディレクトリ、およびファイル名と拡張子を示します。また、キーボードから入力する文字も示します。
	例 :
	`#include <iostream.h> void main ( ) the pointer psz`
	`chmod u+w *`
	`\tux\data\ap`
	`.doc`
	`tux.doc`
	`BITMAP`
	`float`

等幅太字	コード内の重要な単語を示します。 例 : `void **commit** ( )`
等幅イタリック体	コード内の変数を示します。 例 : `String *expr*`
大文字	デバイス名、環境変数、および論理演算子を示します。 例 : LPT1 SIGNON OR
{ }	構文の行で選択肢を示します。かっこは入力しません。

規則	項目
[ ]	構文の行で省略可能な項目を示します。かっこは入力しません。 例: `buildobjclient [-v] [-o name ] [-f file-list]...[-l file-list]...`
|	構文の行で、相互に排他的な選択肢を分離します。記号は入力しません。
...	コマンド行で次のいずれかを意味します。 ■ コマンド行で同じ引数を繰り返し指定できること ■ 省略可能な引数が文で省略されていること ■ 追加のパラメータ、値、その他の情報を入力できること 省略符号は入力しません。 例: `buildobjclient [-v] [-o name ] [-f file-list]...[-l file-list]...`
. . .	コード例または構文の行で、項目が省略されていることを示します。省略符号は入力しません。

セクション 5 ―ファイル形式、データ記述方法、MIB、およびシステム・プロセスのリファレンス

表 1BEA Tuxedo **のファイル形式、データ記述方法、**MIB**、およびシステム・プロセス**

名前	説明
テーブルとファイルの紹介	このマニュアルの概要
ACL_MIB(5)	ACL の管理情報ベース
APPQ_MIB(5)	/Q の管理情報ベース
AUTHSVR(5)	サーバ提供のユーザ単位の認証
compilation(5)	BEA Tuxedo システムのアプリケーション・コンポーネントのコンパイル命令
DMADM(5)	ドメイン管理サーバ
DMCONFIG(5)	テキスト形式の Domains コンフィギュレーション・ファイル

表 1BEA Tuxedo のファイル形式、データ記述方法、MIB、およびシステム・プロセス (続き

名前	説明
DMCONFIG for GWTOPEND(5)	TOP END Domain Gateway のテキスト形式の Domains コンフィギュレーション・ファイル
DM_MIB(5)	ドメインの管理情報ベース
EVENTS(5)	システム生成イベントのリスト
EVENT_MIB(5) に関する追加情報	イベント・ブローカの管理情報ベース
factory_finder.ini(5)	FactoryFinder の Dmains コンフィギュレーション・ファイル
Ferror、Ferror32(5)	FML エラー・コード
field_tables(5)	フィールド名に対する FML マッピング・ファイル
GWADM(5)	ドメイン・ゲートウェイ管理サーバ
GWTDOMAIN(5)	TDomain ゲートウェイ・プロセス
GWTOPEND(5)	TOP END Domain Gateway プロセス
GWTUX2TE、GWTE2TUX(5)	BEA Tuxedo および BEA TOP END ゲートウェイ・サーバ
langinfo(5)	言語情報定数
LAUTHSVR	WebLogic Server 組み込み LDAP ベース認証サーバ
MIB(5)	管理情報ベース
nl_types(5)	ネイティブ言語データ型
servopts(5)	サーバ・プロセスの実行時オプション
TM_MIB(5)	BEA Tuxedo システムの管理情報ベース
TMFFNAME(5)	FactoryFinder および NameManager サービスを実行するサーバ
TMIFRSVR(5)	インターフェイス・リポジトリ・サーバ
TMQFORWARD(5)	メッセージ転送サーバ

表 1BEA Tuxedo **のファイル形式、データ記述方法、**MIB**、およびシステム・プロセス (続き**

名前	説明
TMQUEUE(5)	メッセージ・キュー・マネージャ
TMSYSEVT(5)	システム・イベント通知プロセス
tmtrace(5)	実行時のトレース機能
TMUSREVT(5)	ユーザ・イベント通知プロセス
tperrno(5)	BEA Tuxedo システム・エラー・コード
tpurcode(5)	アプリケーションが指定する戻りコードのための BEA Tuxedo システムのグローバル変数
tuxenv(5)	BEA Tuxedo システムの環境変数リスト
tuxtypes(5)	バッファ・タイプ・スイッチ、BEA Tuxedo システムによって提供されるバッファ・タイプの説明
typesw(5)	バッファ・タイプ・スイッチ構造体、各バッファ・タイプに必要なパラメータとルーチン
UBBCONFIG(5)	テキスト形式の BEA Tuxedo コンフィギュレーション・ファイル
viewfile(5)	VIEW 記述用のソース・ファイル
WS_MIB(5)	ワークステーションの管理情報ベース
WSL(5)	ワークステーション・リスト・サーバ

テーブルとファイルの紹介

説明　このセクションでは、各種のテーブルとファイルの形式ついて説明します。

compilation(5) ページでは、アプリケーションのソース・コードをコンパイルするときに必要なヘッダ・ファイル、ライブラリ、および環境変数について要約しています。

このセクションには、BEA Tuxedo システム提供のサーバの説明が含まれています。アプリケーションで BEA Tuxedo システム提供のサーバを使用する場合は、そのアプリケーションのコンフィギュレーション・ファイルにそれらのサーバを指定してください。

servopts のページでは、アプリケーション・サーバの CLOPT パラメータとしてコンフィギュレーション・ファイルに指定できるオプションについて説明します。

BEA Tuxedo 管理情報ベースについては、MIB(5) リファレンス・ページと次に示す MIB ページで説明します。

- ACL_MIB(5)
- APPQ_MIB(5)
- DM_MIB(5)
- EVENT_MIB(5) に関する追加情報
- TM_MIB(5)
- WS_MIB(5)

ACL_MIB(5)

名前 ACL_MIB—ACL の管理情報ベース

形式

```
#include <fml32.h>
#include <tpadm.h>
```

機能説明 BEA Tuxedo MIB は、アクセス制御リスト (ACL) を管理するためのクラスの集合を定義します。これらのクラスに対してアクセスや変更を行う前に、SECURITY を USER_AUTH、ACL、または MANDATORY_ACL に設定して BEA Tuxedo のコンフィギュレーションを作成しておく必要があります。管理要求のフォーマットと管理応答の解釈を行うには、ACL_MIB(5) を共通 MIB リファレンス・ページ MIB(5) と一緒に使用します。このリファレンス・ページで説明するクラスや属性を使用し、MIB(5) の説明に従ってフォーマットした要求を使用すると、アクティブなアプリケーションの既存の ATMI インターフェイスの 1 つを通じて管理サービスを要求できます。ACL_MIB(5) のすべてのクラス定義の追加情報については、14 ページの「ACL_MIB(5) に関する追加情報」を参照してください。

ACL_MIB(5) は、次のクラスで構成されています。

表 2ACL_MIB クラス

クラス名	属性
T_ACLGROUP	ACL グループ
T_ACLPERM	ACL パーミッション
T_ACLPRINCIPAL	ACL プリンシパル (ユーザまたはドメイン)

各クラスの説明セクションには、次の 4 つのサブセクションがあります。

概要
: このクラスに関連付けられている属性の概要

属性表
: クラスの各属性に関する名前、タイプ、パーミション、値、およびデフォルト値を示す表。属性表の形式については以下に示してあります。

属性の意味
: 各属性の意味の説明

制限事項
: このクラスにアクセスし、このクラスを解釈する場合の制限事項

属性表の形式

前述のように、この MIB に含まれる各クラスは、4 つの部分に分けて以下に定義されています。その 1 つが属性表です。属性表はクラス内の属性のリファレンス・ガイドであり、管理者、オペレータ、一般ユーザがそれらの属性を使用してアプリケーションと対話するための方法を説明しています。属性表の各属性の説明には、5 つの構成要素 (名前、タイプ、パーミッション、値、デフォルト) があります。各要素については、MIB(5) を参照してください。

TA_FLAGS 値

MIB(5) は、共通 TA_FLAGS 属性を定義します。この属性は long 型で、共通 MIB フラグ値とコンポーネント MIB 固有フラグ値の両方を持ちます。現時点では、ACL_MIB(5) 固有のフラグ値は定義されていません。

FML32 フィールド・テーブル

このリファレンス・ページで説明する属性のフィールド・テーブルは、システムにインストールした BEA Tuxedo システム・ソフトウェアのルート・ディレクトリからの相対パスで指定される udataobj/tpadm ファイルにあります。${TUXDIR}/udataobj ディレクトリは、FLDTBLDIR 環境変数で指定されるコロン区切りのリストにアプリケーションによって追加される必要があり、フィールド・テーブル名 tpadm() は、FIELDTBLS 環境変数で指定されるカンマ区切りのリストに追加される必要があります。

制限事項

この MIB のヘッダ・ファイルとフィールド・テーブルには、BEA Tuxedo リリース 6.0 以降が実行されているサイト (ネイティブとワークステーションの両方) からのみアクセスできます。

T_ACLGROUP クラスの定義

概要 T_ACLGROUP クラスは、BEA Tuxedo アプリケーションのユーザおよびドメインのグループを表します。

属性表

表 3ACL_MIB(5):T_ACLGROUP クラス定義の属性表

属性	タイプ	パーミッション	値	デフォルト値
TA_GROUPNAME(r)(*)	string	rU-------	*string*[1..30]	N/A
TA_GROUPID(k)	long	rw-------	0 <= *num* <16,384	最低の ID
TA_STATE	string	rw-------	GET: "INA" SET: "{NEW | INV}"	N/A N/A

(k)—GET キー・フィールド
(r)— オブジェクトの作成に必要なフィールド (SET TA_STATE NEW)
(*)—GET/SET キー、SET 操作では 1 つ以上必要

属性の意味

TA_GROUPNAME: *string*[1..30]
グループの論理名。グループ名は表示可能な文字列で、シャープ、カンマ、コロン、および改行文字は使用できません。

TA_GROUPID: 0 <= *num* < 16,384
ユーザに関連付けられるグループ識別子。0 はデフォルト・グループ "other." を示します。作成時に指定しない場合、次に使用可能な (一意な) 1 以上の識別子がデフォルトとして使用されます。

TA_STATE:

GET: {VALid}
GET 操作は、選択した T_ACLGROUP オブジェクトのコンフィギュレーション情報を検索します。次に示す状態は、GET 要求に対する応答で返される TA_STATE の意味を示します。

VALid	T_ACLGROUP オブジェクトが定義され、非アクティブ状態です。これがこのクラスの唯一の有効な状態です。ACL グループがアクティブになることはありません。

SET: {NEW | INValid}

SET 操作は、選択した T_ACLGROUP オブジェクトのコンフィギュレーション情報を更新します。次に示す状態は、SET 要求で設定される TA_STATE の意味を示します。これ以外の状態を設定することはできません。

NEW	アプリケーション用の T_ACLGROUP オブジェクトを作成します。状態の変更は INValid 状態でのみ可能です。正常終了すると、オブジェクトの状態は VALid になります。
unset	既存の T_ACLGROUP オブジェクトを変更します。この組み合わせは INValid 状態では使用できません。正常終了してもオブジェクトの状態は変わりません。
INValid	アプリケーション用の T_ACLGROUP オブジェクトを削除します。状態の変更は VALid 状態でのみ可能です。正常終了すると、オブジェクトの状態は INValid になります。

制限事項　1 人のユーザは 1 つの ACL グループにのみ関連付けることができます。複数の役割を持つユーザまたは複数のグループに関連付けられるユーザについては、複数のユーザ・エントリを定義する必要があります。

T_ACLPERM クラスの定義

概要 T_ACLPERM クラスは、BEA Tuxedo システム・エントリにアクセスできるグループを表します。これらのエンティティ名は文字列です。現在この名前はサービス名、イベント名、およびアプリケーション・キュー名を表しています。

属性表

表 4ACL_MIB(5):T_ACLPERM クラスの定義属性表

属性	タイプ	パーミッション	値	デフォルト値
TA_ACLNAME(r)(*)	string	rw-------	*string*[1..30]	N/A
TA_ACLTYPE(r)(*)	string	rw-------	"ENQ | DEQ | SERVICE | POSTEVENT"	N/A
TA_ACLGROUPIDS	string	rw-------	*string*	N/A
TA_STATE	string	rw-------	GET: "INA" SET: "{NEW | INV}"	N/A N/A

(r)— オブジェクトの作成に必要なフィールド (SET TA_STATE NEW)
(*)—GET/SET キー、SET 操作では 1 つ以上必要

属性の意味

TA_ACLNAME:*string*
パーミッションを与えるエンティティの名前。この名前は、サービス名、イベント名、およびキュー名を表すことができます。ACL 名は表示可能な文字列で、コロン、シャープ、および改行文字は使用できません。

TA_ACLTYPE: ENQ | DEQ | SERVICE | POSTEVENT
パーミッションを与えるエンティティの型。

TA_ACLGROUPIDS:*string*
関連付けられるエンティティへのアクセスが許可されるグループ識別子 (番号) のカンマ区切りリスト。*string* の長さは、マシンのディスク容量によってのみ制限されます。

TA_STATE:

GET: {VALid}

GET 操作は、選択した T_ACLPERM オブジェクトのコンフィギュレーション情報を検索します。次に示す状態は、GET 要求に対する応答で返される TA_STATE の意味を示します。

VALid	T_ACLPERM オブジェクトが定義され、非アクティブ状態です。これがこのクラスの唯一の有効な状態です。ACL パーミッションがアクティブになることはありません。

SET: {NEW | INValid}

SET 操作は、選択した T_ACLPERM オブジェクトのコンフィギュレーション情報を更新します。次に示す状態は、SET 要求で設定される TA_STATE の意味を示します。これ以外の状態を設定することはできません。

NEW	アプリケーション用の T_ACLPERM オブジェクトを作成します。状態の変更は INValid 状態でのみ可能です。正常終了すると、オブジェクトの状態は VALid になります。
unset	既存の T_ACLPERM オブジェクトを変更します。この組み合わせは INValid 状態では使用できません。正常終了してもオブジェクトの状態は変わりません。
INValid	アプリケーション用の T_ACLPERM オブジェクトを削除します。状態の変更は VALid 状態でのみ可能です。正常終了すると、オブジェクトの状態は INValid になります。

制限事項 パーミッションは、個別のユーザ識別子ではなく、グループ・レベルで定義されます。

T_ACLPRINCIPAL クラスの定義

概要 T_ACLPRINCIPAL クラスは、BEA Tuxedo アプリケーションにアクセス可能なユーザまたはドメイン、およびそれらに関連付けられるグループを表します。特定ユーザとしてアプリケーションに参加するには、ユーザ固有のパスワードを提示する必要があります。

属性表

属性の意味

表 5ACL_MIB(5):T_ACLPRINCIPAL **クラス定義の属性表**

属性	タイプ	パーミッション	値	デフォルト値
TA_PRINNAME(r)(*)	string	rU-------	*string*[1..30]	N/A
TA_PRINCLTNAME(k)	string	rw-------	*string*[1..30]	"*"
TA_PRINID(k)	long	rU-------	1 <= *num* < 131,072	最低の ID
TA_PRINGRP(k)	long	rw-------	0 <= *num* < 16,384	0
TA_PRINPASSWD	string	rwx------	*string*	N/A
TA_STATE	string	rw-------	GET: "INA" SET: "{NEW | INV}"	N/A N/A

(k)—GET キー・フィールド
(r)— オブジェクトの作成に必要なフィールド (SET TA_STATE NEW)
(*)—GET/SET キー、SET 操作では 1 つ以上必要

TA_PRINNAME:*string*
: ユーザまたはドメイン (プリンシパル) の論理名。プリンシパル名は表示可能な文字列で、シャープ、コロン、および改行文字は使用できません。

TA_PRINCLTNAME:*string*
: ユーザに関連付けられるクライアント名。通常は、関連付けられたユーザの役割を表し、ユーザのエントリに関する付加的な情報となります。指定しない場合、デフォルト値はワイルドカードのアスタリス

ク (*) になります。クライアント名は表示可能な文字列で、コロンおよび改行文字は使用できません。

TA_PRINID: 1 <= *num* < 131,072
一意なユーザ識別番号。作成時に指定しない場合、次に使用可能な (一意な) 1 以上の識別子がデフォルトとして使用されます。

TA_PRINGRP: 0 <= *num* < 16,384
ユーザに関連付けられるグループ識別子。0 はデフォルト・グループ "other." を示します。作成時に指定しない場合、デフォルト値 0 が割り当てられます。

TA_PRINPASSWD:*string*

TA_STATE:

GET: {VALid}
GET 操作は、選択した T_ACLPRINCIPAL オブジェクトのコンフィギュレーション情報を検索します。次に示す状態は、GET 要求に対する応答で返される TA_STATE の意味を示します。

VALid	T_ACLPRINCIPAL オブジェクトが定義され、非アクティブ状態です。これがこのクラスの唯一の有効な状態です。ACL プリンシパルがアクティブになることはありません。

SET: {NEW | INValid}
SET 操作は、選択した T_ACLPRINCIPAL オブジェクトのコンフィギュレーション情報を更新します。次に示す状態は、SET 要求で設定される TA_STATE の意味を示します。これ以外の状態を設定することはできません。

NEW	アプリケーション用の T_ACLPRINCIPAL オブジェクトを作成します。状態の変更は INValid 状態でのみ可能です。正常終了すると、オブジェクトの状態は VALid になります。

unset	既存の T_ACLPRINCIPAL オブジェクトを変更します。この組み合わせは INValid 状態では使用できません。正常終了してもオブジェクトの状態は変わりません。
INValid	アプリケーション用の T_ACLPRINCIPAL オブジェクトを削除します。状態の変更は VALid 状態でのみ可能です。正常終了すると、オブジェクトの状態は INValid になります。

制限事項 1 人のユーザまたは 1 つのドメインは 1 つの ACL グループにのみ関連付けることができます。複数の役割を持つユーザまたは複数のグループに関連付けられるユーザについては、複数のプリンシパル・エントリを定義する必要があります。

ACL_MIB(5) に関する追加情報

診断

ACL_MIB(5) への接続時には、2 つの一般的なタイプのエラーがユーザに返される場合があります。1 つは、管理要求に対する応答を検索する 3 つの ATMI 関数 (tpcall()、tpgetrply()、および tpdequeue()) が返すエラーです。これらのエラーは、該当するリファレンス・ページの記述に従って解釈されます。

ただし、要求がその内容に対応できるシステム・サービスに正常にルーティングされても、システム・サービス側でその要求を処理できないと判断されると、アプリケーション・レベルのサービス障害としてエラーが返されます。このような場合、tpcall() と tpgetrply() は、tperrno() を TPESVCFAIL に設定してエラーを返し、以下のようにエラーの詳細を示す TA_ERROR、TA_STATUS、および TA_BADFLD フィールドと一緒に、元の要求を含む応答メッセージを返します。TMQFORWARD(5) サーバ経由でシステムに転送された要求に対してサービス障害が発生すると、元の要求で識別される異常終了キューに障害応答メッセージが追加されます (TMQFORWARD に対して -d オプションが指定されたと見なされる)。

管理要求の処理中にサービス・エラーが発生すると、TA_STATUS という FML32 フィールドにエラーの内容を説明したテキストが設定され、TA_ERROR という FML32 フィールドにはエラーの原因 (下記参照) を示す値が設定されます。以下のエラー・コードは、いずれもマイナスであることが保証されています。

以下の診断コードは TA_ERROR で戻されるもので、管理要求が正常に完了したことを示します。これらのコードはマイナスでないことが保証されています。

[*other*]
すべてのコンポーネント MIB に共通のその他のリターン・コードは、MIB(5) リファレンス・ページに指定されています。これらのコードは、ここに定義する ACL_MIB(5) 固有のリターン・コードと相互に排他関係にあることが保証されています。

相互運用性

このリファレンス・ページで定義されているヘッダ・ファイルおよびフィールド・テーブルは、BEA Tuxedo リリース 6.0 以降で利用できます。これらのヘッダやテーブルで定義するフィールドはリリースが変わっても変更されません。ただし、以前のリリースで定義されていない新しいフィールドが追加される場合があります。AdminAPI には、要求を作成するために必要な

ヘッダ・ファイルとフィールド・テーブルがあれば、どのサイトからでもアクセスできます。T_ACLPRINCIPAL、T_ACLGROUP、および T_ACLPERM クラスは、BEA Tuxedo リリース 6.0 で追加されたものです。

移植性 BEA Tuxedo システムの MIB を使用した管理作業をサポートするために必要な既存の FML32 および ATMI 関数、さらにこのリファレンス・ページに定義するヘッダ・ファイルとフィールド・テーブルは、すべてのサポート対象ネイティブ・プラットフォームとワークステーション・プラットフォームで使用可能です。

使用例 以下に、ユーザをグループに追加し、当該グループに対するパーミッションをサービス名に追加するコードを示します。

フィールド・テーブル 属性フィールド識別子にアクセスするには、フィールド・テーブル *tpadm* が必要です。そのためには、次のようにシェルで入力します。

```
$ FIELDTBLS=tpadm
$ FLDTBLDIR=${TUXDIR}/udataobj
$ export FIELDTBLS FLDTBLDIR
```

ヘッダ・ファイル 次のヘッダ・ファイルがインクルードされます。

```
#include <atmi.h>
#include <fml32.h>
#include <tpadm.h>
```

ユーザの追加 以下のコードでは、デフォルト・グループ "other." にユーザを追加します。

```
/* 入力バッファと出力バッファを割り当てる */

ibuf = tpalloc("FML32", NULL, 1000);


 obuf = tpalloc("FML32", NULL, 1000);

 /* 要求タイプを定義する MIB(5) 属性を設定 *
 Fchg32(ibuf, TA_OPERATION, 0, "SET", 0);
 Fchg32(ibuf, TA_CLASS, 0, "T_ACLPRINCIPAL", 0);

 /* Set ACL_MIB(5) attributes */
 Fchg32(ibuf, TA_PRINNAME, 0, ta_prinname, 0);
 Fchg32(ibuf, TA_PRINID, 0, (char *)ta_prinid, 0);
 Fchg32(ibuf, TA_STATE, 0, (char *)"NEW", 0);

 Fchg32(ibuf, TA_PRINPASSWD, 0, (char *)passwd, 0);
```

```
/* 要求を作成 */
if (tpcall(".TMIB", (char *)ibuf, 0, (char **)obuf, olen, 0) 0)  {
fprintf(stderr, "tpcall failed: %s\en", tpstrerror(tperrno));
if (tperrno == TPESVCFAIL) {
Fget32(obuf, TA_ERROR, 0,(char *)ta_error, NULL);
ta_status = Ffind32(obuf, TA_STATUS, 0, NULL);
fprintf(stderr, "Failure: %ld, %s\en",
ta_error, ta_status);
}
/* 追加のエラー処理 */
}
```

ファイル ${TUXDIR}/include/tpadm.h, ${TUXDIR}/udataobj/tpadm,

関連項目 tpacall(3c)、tpalloc(3c)、tpcall(3c)、tpdequeue(3c), tpenqueue(3c)、tpgetrply(3c)、tprealloc(3c), FML 関数の紹介、Fadd、Fadd32(3fml)、Fchg、Fchg32(3fml), Ffind、Ffind32(3fml)、MIB(5)、TM_MIB(5)

『BEA Tuxedo アプリケーションの設定』

『C 言語を使用した BEA Tuxedo アプリケーションのプログラミング』

『FML を使用した BEA Tuxedo アプリケーションのプログラミング』

APPQ_MIB(5)

名前 APPQ_MIB—/Q の管理情報ベース

形式

```
#include <fml32.h>
#include <tpadm.h>
```

機能説明 /Q MIB は、アプリケーション・キューを管理するためのクラスを定義します。

管理要求のフォーマットと管理応答の解釈を行うには、APPQ_MIB(5) を共通 MIB リファレンス・ページ MIB(5) と一緒に使用します。このリファレンス・ページで説明するクラスと属性を使用し、MIB(5) の説明に従ってフォーマットした要求を使用すると、アクティブなアプリケーションの既存の ATMI インターフェイスの 1 つを通じて管理サービスを要求できます。非アクティブなアプリケーションのアプリケーション・キューは、tpadmcall() 関数インターフェイスを使用して管理できます。APPQ_MIB(5) のすべてのクラス定義の追加情報については、53 ページの「APPQ_MIB(5) に関する追加情報」を参照してください。

APPQ_MIB(5) は、次のクラスで構成されています。

表 6APPQ_MIB **クラス**

クラス名	属性
T_APPQ	キュー・スペース内のアプリケーション・キュー
T_APPQMSG	アプリケーション・キュー内のメッセージ
T_APPQSPACE	アプリケーション・キュー・スペース
T_APPQTRANS	アプリケーション・キューに対応したトランザクション

この MIB は、サーバ・キュー (TM_MIB(5) コンポーネントの T_QUEUE クラス) ではなく、アプリケーションで定義される永続的 (信頼性の高いディスク・ベースの) キューおよび非永続的 (メモリ内の) キュー (つまり /Q キュー) を示していることに注意してください。

各クラスの説明セクションには、次の 4 つのサブセクションがあります。

概要
: このクラスに関連付けられている属性の概要

属性表
: クラスの各属性に関する名前、タイプ、パーミション、値、およびデフォルト値を示す表。属性表の形式については以下に示してあります。

属性の意味
: 各属性の意味の説明

制限事項
: このクラスにアクセスし、このクラスを解釈する場合の制限事項

属性表の形式

この MIB に含まれる各クラスは、4 つの部分に分けて定義されています。その 1 つが属性表です。属性表はクラス内の属性のリファレンス・ガイドであり、管理者、オペレータ、一般ユーザがそれらの属性を使用してアプリケーションと対話するための方法を説明しています。

属性表の各属性の説明には、5 つの構成要素 (名前、タイプ、パーミッション、値、デフォルト) があります。各要素については、MIB(5) を参照してください。.

TA_FLAGS 値

MIB(5) は、共通 TA_FLAGS 属性を定義します。この属性は long 型で、共通 MIB フラグ値とコンポーネント MIB 固有フラグ値の両方を持ちます。APPQ_MIB(5) コンポーネントには、次に示すフラグ値が定義されます。これらのフラグ値は、共通 MIB フラグと一緒に使用する必要があります。

QMIB_FORCECLOSE
: T_APPQSPACE オブジェクトの TA_STATE 属性を CLEaning に設定する場合、このフラグはキュー・スペースの状態が ACTive であっても状態を変更できることを示します。

QMIB_FORCEDELETE
: T_APPQSPACE オブジェクトの TA_STATE 属性を INValid に設定する場合、このフラグはキュー・スペースの状態が ACTive であっても、そのキュー・スペースのいずれかにメッセージが存在していても状態を変更できることを示します。同様に、T_APPQ オブジェクトの TA_STATE 属性を INValid に設定する場合、キューにメッセージが

あっても、キュー・スペースにプロセスがアタッチされていても、このフラグによってキューを削除できます。

QMIB_FORCEPURGE

T_APPQ オブジェクトの TA_STATE 属性を INValid に設定する場合、このフラグはメッセージがキューに存在していても状態を変更できることを示します。ただし、選択した T_APPQ オブジェクトに格納されているメッセージがトランザクションにかかわっていると、状態の変更は行われず、ユーザ・ログにエラーが書き込まれます。

FML32 フィールド・テーブル

このリファレンス・ページで説明する属性のフィールド・テーブルは、システムにインストールした BEA Tuxedo ソフトウェアのルート・ディレクトリからの相対パスで指定される udataobj/tpadm ファイルにあります。${TUXDIR}/udataobj ディレクトリは、FLDTBLDIR 環境変数で指定されるパス・リスト (Windows の場合はセミコロンで区切り、それ以外はコロンで区切る) にアプリケーションによって追加される必要があり、フィールド・テーブル名 tpadm() は、FIELDTBLS 環境変数で指定されるカンマ区切りのリストに追加される必要があります。

制限事項

この MIB は、BEA Tuxedo システム 6.0 以降が実行されているサイト (ネイティブおよび Workstation の両方) からのみアクセスできます。

BEA Tuxedo 6.0 より前のリリースが実行されているサイトがアプリケーション内でアクティブ化された場合、この MIB による管理アクセスは次のとおり制限されます。

- SET 操作は許可されません。
- リリース 6.0 より前のサイトのローカル情報にはアクセスできません。

T_APPQ クラスの定義

概要 T_APPQ クラスは、アプリケーション・キューを表します。1 つのアプリケーション・キュー・スペースには、1 つまたは複数のアプリケーション・キューが存在します。

制限事項 すべてのキー・フィールドを未設定にすると、このクラスのインスタンスをすべて検索できません。反対に、1 つのアプリケーション・キュー・スペースを明示的に指定するには、適切なキー・フィールドを指定する必要があります。これらの必須キー・フィールドは、TA_APPQSPACENAME、TA_QMCONFIG、および TA_LMID です。ただし、アプリケーションの環境がコンフィギュレーションされていない (TUXCONFIG 環境変数が設定されていない) 場合を除きます。この場合、TA_LMID を省略する必要があります。たとえば、tpcall() を使用して要求で TA_APPQSPACENAME、TA_QMCONFIG、および TA_LMID 属性を設定した場合、指定したキュー・スペース内のすべての T_APPQ オブジェクトが検索されます。

属性表

表 7APPQ_MIB(5):T_APPQ **クラス定義の属性表**

属性 [a]	タイプ	パーミッション	値	デフォルト値
TA_APPQNAME(k)(r)(*)	string	ru-r--r--	*string*[1..15]	N/A
TA_APPQSPACENAME(k)(r)(*)	string	ru-r--r--	*string*[1..15]	N/A
TA_QMCONFIG(k)(r)(*)	string	ru-r--r--	*string*[1..78]	N/A
TA_LMID(k)(r)(*) [b]	string	ru-r--r--	*string*[1..30]	N/A
TA_STATE [c]	string	rw-r--r--	GET: "VAL" SET: "{NEW | INV}"	N/A N/A
TA_APPQORDER [d]	string	rw-r--r--	{PRIO|TIME|LIFO|FIFO | EXPIR}	FIFO
TA_DEFEXPIRATIONTIME	string	rw-r--r--	{+*seconds* | NONE}	N/A

表 7APPQ_MIB(5):T_APPQ クラス定義の属性表 (続き)

属性 [a]	タイプ	パーミッション	値	デフォルト値
TA_DEFDELIVERYPOLICY	string	rw-r--r--	{PERSIST | NONPERSIST}	PERSIST
TA_CMD	string	rw-r--r--	*shell-command-string*[0..127] [e]	""
TA_CMDHW	string	rw-r--r--	0 <= *num* [bBm%]	100%
TA_CMDLW	string	rw-r--r--	0 <= *num* [bBm%]	0%
TA_CMDNONPERSIST	string	rw-r--r--	*shell-command-string* [0..127] [e]	""
TA_CMDNONPERSISTHW	string	rw-r--r--	0 <= *num*[bB%]	100%
TA_CMDNONPERSISTLW	string	rw-r--r--	0 <= *num*[bB%]	0%
TA_MAXRETRIES	long	rw-r--r--	0 <= *num*	0
TA_OUTOFORDER	string	rw-r--r--	{NONE | TOP | MSGID}	NONE
TA_RETRYDELAY	long	rw-r--r--	0 <= *num*	0
TA_CURBLOCKS	long	r--r--r--	0 <= *num*	N/A
TA_CURMSG	long	r--r--r--	0 <= *num*	N/A
TA_CURNONPERSISTBYTES	long	r--r--r--	0 <= *num*	N/A
TA_CURNONPERSISTMSG	long	r--r--r--	0 <= *num*	N/A

(k)—GET キー・フィールド [f]
(r)— オブジェクト作成に必要なフィールド
(*)— 必須の SET キー・フィールド

[a] T_APPQ クラスの属性はすべてローカルです。
[b] アプリケーションがコンフィギュレーションされていない (TUXCONFIG 環

境変数が設定されていない) 場合を除き、TA_LMID はキー・フィールドとして指定する必要があります。
[c] T_APPQ オブジェクトのすべての操作 (GET と SET) は、関連付けられているキュー・スペースを自動的にオープンします。つまり、キュー・スペースの状態が OPEn または ACTive になっていない場合、暗黙的に OPEn に設定します。キュー・スペースが大きいと、この操作は時間がかかります。
[d] アプリケーション・キューの作成後は、TA_APPQORDER を変更できません。
[e] BEA Tuxedo 8.0 以前のリリースの場合、この属性の文字列の長さは最大 78 バイトです。
[f] 1 つのアプリケーション・キュー・スペースを明示的に指定するには、GET 操作で適切なキー・フィールドを指定する必要があります。

属性の意味

TA_APPQNAME: *string*[1..15]
アプリケーション・キューの名前。

TA_APPQSPACENAME: *string*[1..15]
アプリケーション・キューが存在するアプリケーション・キュー・スペースの名前。

TA_QMCONFIG: *string*[1..78]
アプリケーション・キュー・スペースが存在するファイルまたはデバイスの絶対パス名。

TA_LMID: *string*[1..30] (no comma)
アプリケーション・キュー・スペースが存在する論理マシンの識別子。

TA_STATE:

GET: {VALid}
GET 操作は、選択したアプリケーション・キューに関する情報を検索します。以下に、GET 要求に対する応答で返される TA_STATE の意味を示します。

VALid	指定したキューが存在します。この状態は INActive と同等で、パーミッションのチェックに使用します。

SET: {NEW | INValid}
SET 操作は、選択したアプリケーション・キューの特性を変更するか、または新しいキューを作成します。以下に、SET 要求

によって返される TA_STATE の意味を示します。States not listed cannot be set.

NEW	指定されたキュー・スペースに新しいキューを作成します。正常に作成されると、キューの状態は VALid になります。
INValid	指定されたキューを削除します。削除するには、キューの状態が VALid でなければなりません。キュー・スペースにプロセスがアタッチされている (ACTive 状態) 場合、TA_FLAGS 属性に QMIB_FORCEDELETE フラグが設定されていない限り、キューは削除されません。また、キューにメッセージが存在する場合、QMIB_FORCEPURGE が制定されていない限りキューは削除されません。正常終了すると、オブジェクトの状態は INValid になります。
unset	アプリケーション・キューを変更します。正常終了してもオブジェクトの状態は変わりません。

TA_APPQORDER:

キュー内のメッセージを処理する順序。有効値は、PRIO、TIME、または EXPIR です。ソート基準の組み合わせは、最初に最上位基準、次にその他の基準、最後に相互に排他的な LIFO または FIFO (省略可) の順に指定します。EXPIR を指定すると、期限切れ時間のないメッセージは、期限切れ時間のあるすべてのメッセージの後で処理されます。FIFO と LIFO のいずれも指定しない場合は、FIFO が使用されます。キューの作成時に順序を指定しない場合、デフォルトの順序は FIFO となります。たとえば、次の設定はいずれも有効です。

```
PRIO
PRIO,TIME,LIFO
TIME,PRIO,FIFO
TIME,FIFO
EXPIR
EXPIR,PRIO,FIFO
TIME,EXPIR,PRIO,FIFO
```

TA_CMD: *shell-command-string*[0..127]
永続的 (ディスク・ベースの) メッセージの上限値 TA_CMDHW に達すると、指定したコマンドが自動的に実行されます。このコマンドは、下限値 TA_CMDLW に達した後に再び上限値に達すると再実行されます。

BEA Tuxedo 8.0 以前のリリースでは、TA_CMD 属性の長さは最大 78 バイトです。

TA_CMDHW: 0 <= *num*[bBm%]

TA_CMDLW: 0 <= *num*[bBm%]
TA_CMD 属性で指定したコマンドの自動実行を制御する上限値と下限値。どちらも 0 以上の整数です。TA_CMDHW と TA_CMDLW の後には以下のいずれかのキー文字が続きます。キー文字は、TA_CMDHW および TA_CMDLW と一致している必要があります。

b
上限値と下限値を、キュー内の永続的 (ディスク・ベース) メッセージで使用するバイト数で設定します。

B
上限値と下限値を、キュー内の永続的メッセージで使用するブロック数で設定します。

m
上限値と下限値を、キュー内の永続的メッセージと一時的メッセージの数で設定します。

%
上限値と下限値を、キューの容量のパーセンテージで設定します。これには永続メッセージだけが含まれます。

たとえば、TA_CMDLW が 50m で TA_CMDHW が 100m であれば、TA_CMD に指定されたコマンドはキューに 100 メッセージ追加されたときに実行され、キュー内のメッセージが 50 に減少した後に再び 100 まで増加しない限り再実行されません。

TA_CMDNONPERSIST: *shell-command-string*[0..127]
この属性は、一時的メッセージ (メモリ・ベースの配信メッセージ) の上限値 TA_CMDNONPERSISTHW に達すると自動的に実行されるコマンドを指定します。このコマンドは、一時的メッセージの下限値

TA_CMDNONPERSISTLW に達した後に再び上限値に達すると再実行されます。

BEA Tuxedo 8.0 以前のリリースでは、TA_CMDNONPERSIST 属性の文字列の長さは最大 78 バイトです。

TA_CMDNONPERSISTHW: 0 <= *num*[bB%]

TA_CMDNONPERSISTLW: 0 <= *num*[bB%]

これらの属性は、TA_CMDNONPERSIST 属性で指定したコマンドの自動実行を制御する上限値と下限値を指定します。どちらも 0 以上の整数で、次のいずれかのキー文字が続きます。キー文字は、TA_CMDNONPERSISTHW および TA_CMDNONPERSISTLW と一致する必要があります。

b
: 上限値と下限値を、キュー内の一時的 (メモリ内の) メッセージで使用するバイト数で表します。

B
: 上限値と下限値を、キュー内の一時的 (メモリ内の) メッセージで使用するブロック数で表します。

%
: 上限値と下限値を、キューによって使用されるキュー・スペース内の一時的メッセージのために確保されている共用メモリ容量のパーセンテージで表します。

TA_CMDHW および TA_CMDLW 属性 (後に m が続く) を通して指定するメッセージしきい値タイプは、キュー内の永続的メッセージと一時的メッセージの両方に適用されます。したがって、TA_CMDNONPERSISTHW と TA_CMDNONPERSISTLW のしきい値タイプとしては使用されません。

TA_CURBLOCKS: 0 <= *num*
: 現在キューによって使用されているディスク・ページの数。

TA_CURMSG: 0 <= *num*
: 現在キューによって使用されている永続的メッセージの数。キュー内のメッセージの合計数を調べるには、この値に TA_CURMEMMSG を追加します。

TA_DEFAULTEXPIRATIONTIME:

この属性は、期限切れ時間が明示的に設定されていないキュー内のメッセージに対して期限切れ時間を指定します。期限切れ時間は、相対期限切れ時間または NONE のいずれかに設定できます。相対期限切れ時間は、メッセージがキュー・マネージャ・プロセスに到着した後、一定の時間数をメッセージと関連付けることによって決定されます。メッセージの期限切れ時間に達した時点で、メッセージがキューから取り出されるか、管理インターフェイスを介して削除されない場合、そのメッセージに関連付けられているすべてのリソースはシステムによって再使用され、統計情報は更新されます。トランザクション中にメッセージの期限が切れてもトランザクションは失敗しません。トランザクション内でキューへの登録、またはキューからの取り出し中に有効期限が切れたメッセージは、トランザクションが終了した時点でキューから削除されます。メッセージの有効期限が切れたことの通知は行われません。デフォルトの期限切れ時間がキューに指定されていない場合、明示的な期限切れ時間のないメッセージが期限切れになることはありません。キューの期限切れ時間を変更しても、変更前にキュー内に存在していたメッセージの期限切れ時間は変わりません。

形式は *+seconds* です。*seconds* は、キュー・マネージャが操作を正常終了してからメッセージが期限切れになるまでの経過秒数です。*seconds* をゼロ (0) に設定すると、メッセージはすぐに期限切れになります。

この属性の値は、文字列 NONE に設定することもできます。文字列 NONE は、明示的な期限切れ時間を設定せずにキューに登録されたメッセージは期限切れにならないことを示します。既にキュー内にあるメッセージの期限切れ時間を変更するには、APPQ_MIB の T_APPQMSG クラスの TA_EXPIRETIME 属性を使用します。

TA_DEFDELIVERYPOLICY:

この属性は、キューに登録されるメッセージに対して配信モードが指定されていない場合、キューにデフォルトの配信ポリシーを指定します。値が PERSIST の場合、配信モードが明示的に指定されずにキューに登録されたメッセージは、永続的な (ディスク・ベースの) 方法で配信されます。値が NONPERSIST の場合、配信モードが明示的に指定されずにキューに登録されたメッセージは、非永続的な (メモリ内の) 方法で配信されます。キューのデフォルトの配信ポリシーを変更しても、変更前にキュー内にあったメッセージに対する配信サービスの

品質は変わりません。現在キュー・スペース内にあるメッセージに対する応答キューを変更する場合、キューのデフォルトの配信ポリシーの変更によって、サービスの応答基準が変更されることはありません。

非永続的配信では、メモリ領域のすべてが使用されている場合や分割されたメッセージをキューに登録できない場合、メッセージ用の永続ストレージが十分にあっても、メッセージのキュー登録は異常終了します。同様に、永続ストレージのすべてが使用されている場合や分割されたメッセージをキューに登録できない場合、メッセージ用の非永続ストレージが十分にあっても、メッセージの登録操作は異常終了します。キュー・スペースの T_APPQSPACE クラスの TA_MEMNONPERSIST 属性がゼロ (0) の場合、一時的メッセージ用の領域は確保されません。このような場合には、一時的メッセージをキューに登録しようとしても異常終了します。たとえば、メッセージに対してサービスの配信品質が指定されずに、ターゲット・キューの TA_DEFDELIVERYPOLICY 属性が NONPERSIST に設定された場合などです。

TA_MAXRETRIES: 0 <= *num*

異常終了キュー・メッセージに対する最大リトライ回数。指定リトライ回数に達すると、メッセージは対応するアプリケーション・キュー・スペースのエラー・キューに入れられます。エラー・キューがない場合、メッセージは削除されます。デフォルト値はゼロです。

TA_OUTOFORDER: {NONE | TOP | MSGID}

順序を無視したメッセージ処理を行う方法。デフォルト値は NONE です。

TA_RETRYDELAY: 0 <= *num*

異常終了キュー・メッセージのリトライ間遅延 (秒)。デフォルト値はゼロです。

TA_CURNONPERSISTBYTES: 0 <= *num*

キュー内の一時的メッセージが現在使用している共用メモリのバイト数。

TA_CURNONPERSISTMSG: 0 <= *num*

現在キューにある一時的メッセージの数。キュー内のメッセージの合計数を調べるには、この値に TA_CURMSG を追加します。

T_APPQMSG クラスの定義

概要 T_APPQMSG クラスは、アプリケーション・キューに格納されたメッセージを表します。メッセージは管理者が作成するのではなく、tpenqueue() 呼び出しによって作成されます。メッセージは、tpdequeue() 呼び出しまたは管理者によって削除されます。また、管理者はメッセージの特定の属性を変更できます。たとえば、管理者はメッセージを同一キュー・スペース内の別のキューに移動したり、メッセージの優先順位を変更したりできます。

制限事項 すべてのキー・フィールドを未設定にすると、このクラスのインスタンスをすべて検索できません。反対に、1 つのアプリケーション・キュー・スペースを明示的に指定するには、適切なキー・フィールドを指定する必要があります。これらの必須キー・フィールドは、TA_APPQSPACENAME、TA_QMCONFIG、および TA_LMID です。ただし、アプリケーションの環境がコンフィギュレーションされていない (TUXCONFIG 環境変数が設定されていない) 場合を除きます。この場合、TA_LMID を省略する必要があります。たとえば、tpcall() を使用して、要求で TA_APPQSPACENAME、TA_QMCONFIG、および TA_LMID 属性を設定した場合、指定したキュー・スペース内のすべての T_APPQMSG オブジェクトが検索されます。

属性表

表 8APPQ_MIB(5):T_APPQMSG クラス定義の属性表

属性 [a]	タイプ	パーミッション	値	デフォルト値
TA_APPQMSGID(k)(*)	string	r--r--r--	*string*[1..32]	N/A
TA_APPQNAME(k)(*)	string	r--r--r--	*string*[1..15]	N/A
TA_APPQSPACENAME(k) (*)	string	r--r--r--	*string*[1..15]	N/A
TA_QMCONFIG(k)(*)	string	r--r--r--	*string*[1..78]	N/A
TA_LMID(k)(*)[b]	string	r--r--r--	*string*[1..30]	N/A
TA_STATE[c]	string	rw-r--r--	GET: "VAL" SET: "INV"	N/A N/A

表 8APPQ_MIB(5):T_APPQMSG **クラス定義の属性表 (続き)**

属性 [a]	タイプ	パーミッション	値	デフォルト値
TA_NEWAPPQNAME	string	-w--w----	*string*[1..15]	N/A
TA_PRIORITY	long	rw-rw-r--	{ 1 <= *num* <= 100 | -1 }	N/A
TA_TIME	string	rw-rw-r--	{*YY*[*MM*[*DD*[*hh*[*mm*[*ss*]]]]] | +*seconds*}	N/A
TA_EXPIRETIME	string	rw-rw-r--	{*YY*[*MM*[*DD*[*hh*[*mm*[*ss*]]]]] | +*seconds*}	N/A
TA_CORRID(k)	string	r--r--r--	*string*[0..32]	N/A
TA_PERSISTENCE (k)	string	r--r--r--	{PERSIST|NONPERSIST}	N/A
TA_REPLYPERSISTENCE	string	r--r--r--	{PERSIST | NONPERSIST | DEFAULT}	N/A
TA_LOWPRIORITY(k)	long	k--k--k--	1 <= *num* <= 100	1
TA_HIGHPRIORITY(k)	long	k--k--k--	1 <= *num* <= 100	100
TA_MSGENDTIME(k)	string	k--k--k--	{*YY*[*MM*[*DD*[*hh*[*mm*[*ss*]]]]] | +*seconds*}	MAXLONG
TA_MSGSTARTTIME(k)	string	k--k--k--	{*YY*[*MM*[*DD*[*hh*[*mm*[*ss*]]]]] | +*seconds*}	0
TA_MSGEXPIREENDTIME(k)	string	k--k--k--	{*YY*[*MM*[*DD*[*hh*[*mm*[*ss*]]]]] | +*seconds*|*NONE*}	MAXLONG
TA_MSGEXPIRESTARTTIME(k)	string	k--k--k--	{*YY*[*MM*[*DD*[*hh*[*mm*[*ss*]]]]] | +*seconds*}	0

表 8APPQ_MIB(5):T_APPQMSG **クラス定義の属性表 (続き)**

属性 [a]	タイプ	パーミッション	値	デフォルト値
TA_CURRETRIES	long	r--r--r--	0 <= *num*	N/A
TA_MSGSIZE	long	r--r--r--	0 <= *num*	N/A

(k)—GET キー・フィールド [d]
(*)— 必須の SET キー・フィールド

[a] T_APPQMSG クラスの属性はすべてローカルです。
[b] アプリケーションがコンフィギュレーションされていない (TUXCONFIG 環境変数が設定されていない) 場合を除き、TA_LMID はキー・フィールドとして指定する必要があります。
[c] T_APPQMSG オブジェクトのすべての操作 (GET と SET) は、関連付けられているキュー・スペースを自動的にオープンします。つまり、キュー・スペースの状態が OPEn または ACTive になっていない場合、暗黙的に OPEn に設定します。キュー・スペースが大きいと、この操作は時間がかかります。
[d] 1 つのアプリケーション・キュー・スペースを明示的に指定するには、GET 操作で適切なキー・フィールドを指定する必要があります。

属性の意味

TA_APPQMSGID: *string*[1..32]
キュー・メッセージの一意な識別子。GET 操作または SET 操作のためにメッセージを選択する際に使用できます。等号比較に使用する場合を除き、この値に意味を持たせることはできません。

TA_APPQNAME: *string*[1..15]
メッセージの格納先となるアプリケーション・キューの名前。

TA_APPQSPACENAME: *string*[1..15]
メッセージが存在するアプリケーション・キュー・スペースの名前。

TA_QMCONFIG: *string*[1..78]
アプリケーション・キュー・スペースが存在するファイルまたはデバイスの絶対パス名。

TA_LMID: *string*[1..30] (no comma)
アプリケーション・キュー・スペースが存在する論理マシンの識別子。

TA_STATE:

GET: {VALid}

GET 操作は、選択したメッセージに関する情報を検索します。以下に、GET 要求に対する応答で返される TA_STATE の意味を示します。

VALid	メッセージが存在します。この状態は INActive と同等で、パーミッションのチェックに使用します。

SET: {INValid}

GET 操作は、選択したメッセージの特性を変更します。以下に、SET 要求によって返される TA_STATE の意味を示します。これ以外の状態を設定することはできません。

INValid	メッセージがキュー・スペースから削除されます。この操作を行うには、メッセージの状態が VALid でなければなりません。正常終了すると、オブジェクトの状態は INValid になります。
unset	メッセージを変更します。正常終了してもオブジェクトの状態は変わりません。

TA_CURRETRIES: 0 <= *num*

このメッセージに対して現在までに行われたリトライ回数。

TA_CORRID: *string*[0..32]

tpenqueue(3c) 要求にアプリケーションが設定するこのメッセージの相関識別子。空文字列は、相関識別子がないことを示します。

TA_EXPIRETIME:

この属性は、メッセージが期限切れになる時間を指定します。つまり、ここで指定した時間になると、キューに残っているか、管理インターフェイスを介して削除されなかったメッセージがキューから削除されます。メッセージが期限切れになると、そのメッセージによって使用されていたすべてのリソースはシステムによって再使用され、統計情報は更新されます。トランザクション中にメッセージの期限が切れてもトランザクションは失敗しません。トランザクション内でキューへの登録、またはキューからの取り出し中に有効期限が切れたメッセージは、トランザクションが終了した時点でキューから削除されます。メッセージの有効期限が切れたことの通知は行われません。

期限切れ時間の値を変更するキュー・マネージャがメッセージの期限切れをサポートしていても、使用している BEA Tuxedo システムのバージョンがメッセージ期限切れをサポートしていない場合、その BEA Tuxedo システムではキューに登録されるメッセージに期限切れ時間を追加できません。期限切れ時間を追加しようとしても失敗します。

期限切れ時間が設定されない場合、GET 操作によって空文字列が返されます。TA_EXPIRETIME の形式は次のいずれかです。

+seconds

メッセージを指定秒数後に削除することを指定します。*seconds* の値をゼロ (0) に設定すると、メッセージはキューからすぐに削除されます。相対期限切れ時間は、MIB 要求が到着

し、対応するキュー・マネージャによって処理された時間に基づいて算出します。

YY[*MM*[*DD*[*hh*]*mm*[*ss*]]]]
: キューから取り出されていないか、管理インターフェイスを介して削除されていないメッセージに対して、そのメッセージを削除する年、月、日、時、分、秒を指定します。省略された単位のデフォルト値はそれぞれの最小値となります。たとえば、9506 は 950601000000 と同じです。00 から 37 の年は 2000 から 2037、70 から 99 は 1970 から 1999 として処理されます。38 から 69 は無効です。絶対時間による期限は、キュー・マネージャ・プロセスが存在するマシンの時間によって決まります。

NONE
: メッセージが期限切れにならないことを指定します。

TA_LOWPRIORITY: 1 <= *num* <= 100
TA_HIGHPRIORITY: 1 <= *num* <= 100
: T_APPQMSG オブジェクトを検索する最高 / 最低優先順位。この属性は、GET 操作のキー・フィールドとしてのみ使用できます。

TA_MSGEXPIRESTARTTIME:
TA_MSGEXPIREENDTIME:
: T_APPQMSG オブジェクトを検索する期限切れ開始 / 終了時間。範囲にはどちらの値も含まれます。開始 / 終了時間は絶対時間値で指定しなければなりません。形式については、TA_EXPIRETIME を参照してください。この属性は、GET 操作のキー・フィールドとしてのみ使用できます。

TA_MSGSIZE: 0 <= *num*
: メッセージのサイズ (バイト)。

TA_MSGSTARTTIME:
TA_MSGENDTIME:
: T_APPQMSG オブジェクトを検索する。開始 / 終了時間。範囲にはどちらの値も含まれます。開始 / 終了時間は絶対時間値で指定しなければなりません。形式については、TA_TIME を参照してください。この属性は、GET 操作のキー・フィールドとしてのみ使用できます。

TA_NEWAPPQNAME: *string*[1..15]
: 選択したメッセージの移動先となるキューの名前。このキューは同一キュー・スペースに存在しなければなりません。この操作を行うには、メッセージの状態が VALid でなければなりません。この属性は GET 操作では返されません。移動するメッセージに対するサービス配信の品質は、新しいキューのデフォルトの配信ポリシーによって変更されません。期限付きのメッセージが移動された場合、移動元では相対時間による期限が指定されていたとしても、移動先のキューでの絶対時間による期限になります。

TA_PERSISTENCE:
: メッセージが配信されるサービス品質。この読み取り専用の状態は、一時的メッセージでは NONPERSIST、永続的メッセージでは PERSIST に設定されます。

TA_PRIORITY: 1 <= *num* <= 100
: メッセージの優先順位。

TA_REPLYPERSISTENCE:
: メッセージの応答を配信するときのサービス品質。この読み取り専用の状態は、一時的メッセージでは NONPERSIST に、永続的メッセージでは PERSIST に、応答がその登録先になるキューに対して確立されているデフォルトの配信ポリシーを使用するときは DEFAULT に設定されます。

  デフォルトの配信ポリシーは、メッセージに対する応答がキューに登録されるときに決定される点に注意してください。つまり、元のメッセージがキューに登録されてから応答が登録されるまでの間に、応答キューのデフォルトの配信ポリシーが変更された場合、応答が最後に登録される時点で有効な方針が使用されます。

TA_TIME:
: メッセージの処理時間。形式は次のいずれかです。

  +seconds
  : メッセージを *seconds* 秒後に処理することを指定します。ゼロ (0) を指定すると、メッセージは即座に処理されます。

  YY[*MM*[*DD*[*hh*[*mm*[*ss*]]]]]
  : メッセージを処理する年、月、日、時、分、秒を指定します。省略された単位のデフォルト値はそれぞれの最小値となりま

す。たとえば、9506 は 950601000000 と同じです。00 から 37 の年は 2000 から 2037、70 から 99 は 1970 から 1999 として処理されます。38 から 69 は無効です。

T_APPQSPACE クラスの定義

概要

T_APPQSPACE クラスは、アプリケーション・キュー・スペースを表します。アプリケーション・キュー・スペースとは、BEA Tuxedo システム・デバイス内の領域です。デバイスとその属性については、TM_MIB(5) の T_DEVICE クラスを参照してください。一般に、各キュー・スペースには 1 つまたは複数のアプリケーション・キューがあり、各キューにはメッセージが格納されます。

キュー・スペースには、名前 (TA_APPQSPACENAME 属性)、キュー・スペースが存在するデバイス (TA_QMCONFIG 属性)、およびデバイスが存在する論理マシン (TA_LMID 属性) によって一意に識別されます。

キュー・スペースは通常、コンフィギュレーション済みのアプリケーションの中では 1 つのサーバ・グループだけに関連付けられます。キュー・スペース名とデバイス名は、T_GROUP オブジェクトの TA_OPENINFO 属性のコンポーネントです。

制限事項

すべてのキー・フィールドを未設定にすると、このクラスのインスタンスをすべて検索できません。反対に、アプリケーション・キュー・スペースを 1 つでも明示的に指定するには、すべてのキー・フィールドを指定する必要があります。コンフィギュレーションされていない (TUXCONFIG 環境変数が未設定の) アプリケーションのコンテキストで tpadmcall() を介してローカル・キュー・スペースにアクセスすると、例外が 1 つ発生します。この場合、TA_LMID キー・フィールドを省略する必要があります。

/Q MIB 内のすべてのオブジェクトに対する操作は暗黙的にキュー・スペースを使用するため、上記のキュー・スペースのアクセスに関する制限は、T_APPQ、T_APPQMSG、および T_APPQTRANS オブジェクトにも適用されます。

属性表

表 9APPQ_MIB(5):T_APPQSPACE **クラス定義の属性表**

属性 [a]	タイプ	パーミッション	値	デフォルト値
TA_APPQSPACENAME(k)(r)(*)	string	ru-r--r--	*string*[1..15]	N/A
TA_QMCONFIG(k)(r)(*)	string	ru-r--r--	*string*[1..78]	N/A
TA_LMID(k)(r)(*)b	string	ru-r--r--	*string*[1..30]	N/A

表 9APPQ_MIB(5):T_APPQSPACE **クラス定義の属性表 (続き)**

属性 [a]	タイプ	パーミッション	値	デフォルト値
TA_STATE(k)c	string	rwxrwxr--	GET: "{INA | INI | OPE | ACT}"	N/A
			SET: "{NEW | OPE | CLE | INV}"	N/A
TA_BLOCKING	long	rw-r--r--	0 <= *num*	16
TA_ERRORQNAME	string	rw-r--r--	*string*[0..15]	""
TA_FORCEINIT	string	rw-r--r--	{Y | N}	N
TA_IPCKEY(r)	long	rw-r--r--	32769 <= *num* <= 262143	N/A
TA_MAXMSG(r)	long	rw-r--r--	0 <= *num*	N/A
TA_MAXPAGES(r)	long	rw-r--r--	0 <= *num*	N/A
TA_MAXPROC(r)	long	rw-r--r--	0 <= *num*	N/A
TA_MAXQUEUES(r)[d]	long	rw-r--r--	0 <= *num*	N/A
TA_MAXTRANS(r)	long	rw-r--r--	0 <= *num*	N/A
TA_MAXACTIONS	long	rw-r--r--	0 <= *num*	0
TA_MAXHANDLES	long	rw-r--r--	0 <= *num*	0
TA_MAXOWNERS	long	rw-r--r--	0 <= *num*	0
TA_MAXTMPQUEUES	long	rw-r--r--	0 <= *num*	0
TA_MAXCURSORS	long	rw-r--r--	0 <= *num*	0
TA_MEMNONPERSIST	string	rw-r--r--	0 <= *num*[bB]	0
TA_MEMFILTERS	long	rw-r--r--	0 <= *num*	0
TA_MEMOVERFLOW	long	rw-r--r--	0 <= *num*	0
TA_MEMSYSTEMRESERVED	long	r--r--r--	0 <= *num*	N/A
TA_MEMTOTALALLOCATED	long	r--r--r--	0 <= *num*	N/A

表 9APPQ_MIB(5):T_APPQSPACE **クラス定義の属性表 (続き)**

属性 [a]	タイプ	パーミッション	値	デフォルト値
TA_CUREXTENT	long	r--r--r--	0 <= *num* <= 100	N/A
TA_CURMSG	long	r--r--r--	{ 0 <= *num* | -1 }	N/A
TA_CURPROC	long	r--r--r--	0 <= *num*	N/A
TA_CURQUEUES	long	r--r--r--	{ 0 <= *num* | -1 }	N/A
TA_CURTRANS	long	R--R--R--	0 <= *num*	N/A
TA_CURACTIONS	long	r--r--r--	0 <= *num*	N/A
TA_CURHANDLES	long	r--r--r--	0 <= *num*	N/A
TA_CUROWNERS	long	r--r--r--	0 <= *num*	N/A
TA_CURTMPQUEUES	long	r--r--r--	0 <= *num*	N/A
TA_CURCURSORS	long	r--r--r--	0 <= *num*	N/A
TA_CURMEMNONPERSIST	long	r--r--r--	0 <= *num*	N/A
TA_CURMEMFILTERS	long	r--r--r--	0 <= *num*	N/A
TA_CURMEMOVERFLOW	long	r--r--r--	0 <= *num*	N/A
TA_HWMSG	long	R--R--R--	0 <= *num*	N/A
TA_HWPROC	long	R--R--R--	0 <= *num*	N/A
TA_HWQUEUES	long	R--R--R--	0 <= *num*	N/A
TA_HWTRANS	long	R--R--R--	0 <= *num*	N/A
TA_HWACTIONS	long	R--R--R--	0 <= *num* <= 100	N/A
TA_HWHANDLES	long	R--R--R--	0 <= *num*	N/A
TA_HWOWNERS	long	R--R--R--	0 <= *num*	N/A
TA_HWTMPQUEUES	long	R--R--R--	0 <= *num*	N/A
TA_HWCURSORS	long	R--R--R--	0 <= *num*	N/A
TA_HWMEMNONPERSIST	long	R--R--R--	0 <= *num*	N/A
TA_HWMEMFILTERS	long	R--R--R--	0 <= *num*	N/A
TA_HWMEMOVERFLOW	long	R--R--R--	0 <= *num*	N/A
TA_PERCENTINIT	long	r--r--r--	0 <= *num*	N/A

(k)—GET キー・フィールド
(r)— オブジェクト作成に必要なフィールド
(*)— 必須の SET キー・フィールド

a. T_APPQSPACE クラスの属性はすべてローカルです。
b. アプリケーションがコンフィギュレーションされていない (TUXCONFIG 環境変数が設定されていない) 場合を除き、TA_LMID はキー・フィールドとして指定する必要があります。
c. T_APPQ、T_APPQMSG、および T_APPQTRANS オブジェクトのすべての操作 (GET と SET) は、関連付けられているキュー・スペースを自動的にオープンします。つまり、キュー・スペースの状態が OPEn または ACTive になっていない場合、暗黙的に OPEn に設定します。キュー・スペースが大きいと、この操作は時間がかかります。
d. アプリケーション・キューの作成後は、TA_MAXQUEUES を変更できません。

属性の意味

TA_APPQSPACENAME: string[1..15]
アプリケーション・キュー・スペースの名前。

TA_QMCONFIG: string[1..78]
アプリケーション・キュー・スペースが存在するファイルまたはデバイスの絶対パス名。

TA_LMID: *string*[1..30] (no comma)
アプリケーション・キュー・スペースが存在する論理マシンの識別子。

TA_STATE:

GET: {INActive | INItializing | OPEn | ACTive}
GET 操作は、選択したアプリケーション・キュー・スペースに関する情報を検索します。以下に、GET 要求に対する応答で返される TA_STATE の意味を示します。

INActive	キュー・スペースが存在します。つまり、キュー・スペースに対するディスク領域がデバイスに確保され、領域が初期化されています (要求時または必要時)。
INItializing	キュー・スペース用のディスク領域を初期化中です。この状態は ACTive と同等で、パーミッションのチェックに使用します。

OPEn	キュー・スペースに対する共用メモリおよびその他の IPC リソースが割り当てられ、初期化されています。しかし、現在共用メモリにアタッチされているプロセスがありません。この状態は INActive と同等で、パーミッションのチェックに使用します。
ACTive	キュー・スペースに対する共用メモリおよびその他の IPC リソースが割り当てられ、初期化されています。共用メモリには現在少なくとも 1 つのプロセスがアタッチされています。これらのプロセスは、キュー・スペースに関連付けられたキュー・サーバ (TMS_QM、TMQUEUE、および多くの場合 TMQFORWARD)、qmadmin(1) などの管理プロセス、または別のアプリケーションに関連付けられたプロセスです。

SET: {NEW | OPEn | CLEaning | INValid}

SET 操作は、選択したアプリケーション・キュー・スペースを変更するか、または新しいキュー・スペースを作成します。以下に、SET 要求によって返される TA_STATE の意味を示します。これ以外の状態を設定することはできません。

NEW	新しいキュー・スペースを作成します。SET が正常終了してこの状態になると、キュー・スペースの状態は INItializing または INActive になります。
OPEn	キュー・スペースに対する共用メモリおよび他の IPC リソースを割り当て、初期化します。これは、キュー・スペースの状態が INActive の場合にのみ実行できます。

CLEaning	キュー・スペースに対する共用メモリおよび他の IPC リソースを削除します。これは、キュー・スペースの状態が OPEn または ACTive 状態の場合にのみ実行できます。状態が ACTive の場合は、QMIB_FORCECLOSE フラグを指定する必要があります。正常終了すると、一時的メッセージはすべて完全に失われます。
INValid	キュー・スペースを削除します。状態が ACTive、またはメッセージがキュー・スペースのいずれかのキューに存在する場合、QMIB_FORCEDELETE フラグが渡されないとエラーが報告されます。正常終了すると、オブジェクトの状態は INValid になります。正常終了すると、一時的メッセージはすべて完全に失われます。
unset	アプリケーション・キュー・スペースを変更します。正常終了してもオブジェクトの状態は変わりません。

TA_BLOCKING: 0 <= *num*

キュー・スペースのディスク領域管理で使用するブロック化係数。新しいキュー・スペースが作成された場合のデフォルトは 16 です。

TA_CURACTIONS: 0 <= *num*

この属性は、キュー・スペースで使用される現在のアクション数を指定します。この数は、キュー・スペースが OPEn または ACTive の場合、またはキュー・スペースを新規に作成する場合にのみ指定できます。どの条件にも該当しない場合は、-1 が返されます。

TA_CURCURSORS: 0 <= *num*

この属性は、キュー・スペースで使用される現在のカーソル数を指定します。この数は、キュー・スペースが OPEn または ACTive の場合、またはキュー・スペースを新規に作成する場合にのみ指定できます。どの条件にも該当しない場合は、-1 が返されます。

TA_CUREXTENT: 0 <= *num* <= 100

キュー・スペースで使用される現在のエクステント数。最大値は 100 です。TA_MAXPAGES 属性の値が増加するごとに、新規にエクステント

が割り当てられます。この属性を変更すると、キュー・スペース内のすべての一時的メッセージは完全に失われます。

TA_CURHANDLES: 0 <= *num*
: この属性は、キュー・スペースで使用される現在のハンドル数を指定します。この数は、キュー・スペースが OPEn または ACTive の場合、またはキュー・スペースを新規に作成する場合にのみ指定できます。どの条件にも該当しない場合は、-1 が返されます。

TA_CURMEMFILTERS: 0 <= *num*
: この属性は、キュー・スペースでフィルタ用に使用される現在のバイト数を指定します。この数は、キュー・スペースが OPEn または ACTive の場合、またはキュー・スペースを新規に作成する場合にのみ指定できます。どの条件にも該当しない場合は、-1 が返されます。

TA_CURMEMNONPERSIST: 0 <= *num*
: キュー・スペース内の一時的メッセージによって使用される現在のメモリ容量 (バイト)。この数は、キュー・スペースが OPEn または ACTive の場合、またはキュー・スペースを新規に作成する場合にのみ指定できます。どの条件にも該当しない場合は、-1 が返されます。

TA_CURMEMOVERFLOW: 0 <= *num*
: この属性は、キュー・スペース内のオーバフロー・メモリで使用される現在のバイト数を指定します。この数は、キュー・スペースが OPEn または ACTive の場合、またはキュー・スペースを新規に作成する場合にのみ指定できます。どの条件にも該当しない場合は、-1 が返されます。

TA_CURMSG: 0 <= *num*
: キュー・スペース内の現在のメッセージ数。この数は、キュー・スペースが OPEn または ACTive の場合、またはキュー・スペースを新規に作成する場合にのみ指定できます。どの条件にも該当しない場合は、-1 が返されます。

TA_CUROWNERS: 0 <= *num*
: この属性は、キュー・スペースで使用される現在の所有者数を指定します。この数は、キュー・スペースが OPEn または ACTive の場合、またはキュー・スペースを新規に作成する場合にのみ指定できます。どの条件にも該当しない場合は、-1 が返されます。

TA_CURPROC: 0 <= *num*

キュー・スペースに現在アクセスしているプロセスの数。

TA_CURQUEUES: 0 <= *num*

キュー・スペース内に存在しているメッセージの数。この数は、キュー・スペースが OPEn または ACTive の場合、またはキュー・スペースを新規に作成する場合にのみ指定できます。どの条件にも該当しない場合は、-1 が返されます。

TA_CURTMPQUEUES: 0 <= *num*

この属性は、キュー・スペースで使用される一時キューの現在の数を指定します。この数は、キュー・スペースが OPEn または ACTive の場合、またはキュー・スペースを新規に作成する場合にのみ指定できます。どの条件にも該当しない場合は、-1 が返されます。

TA_CURTRANS: 0 <= *num*

キュー・スペースを使用する未終了トランザクションの現在の数。

TA_ERRORQNAME: *string*[0..15]

キュー・スペースに関連付けるエラー・キューの名前。エラー・キューが存在しない場合、GET 要求で空文字列が返されます。

TA_FORCEINIT:{Y | N}

キュー・スペースに対して、新規のエクステントでディスク・ページを初期化するかどうかを指定します。デフォルト値は、「初期化しない」です。デバイス・タイプ (通常ファイルや raw スライスなど) によっては、要求しなくても初期化を行うことができます。

TA_HWACTIONS: 0 <= *num*

この属性は、キュー・スペースが最後にオープンされた後に、キュー・スペースに同時に到達するアクションの最大数を指定します。キュー・スペースの状態が CLEaning に設定されると、この値は 0 にリセットされます。

TA_HWCURSORS: 0 <= *num*

この属性は、キュー・スペースが最後にオープンされた後に、キュー・スペースで同時に作成されるカーソルの最大数を指定します。キュー・スペースの状態が CLEaning に設定されると、この値は 0 にリセットされます。

TA_HWHANDLES: 0 <= *num*

この属性は、キュー・スペースが最後にオープンされた後に、キュー・スペースで同時にオープンされるハンドルの最大数を指定します。キュー・スペースの状態が CLEaning に設定されると、この値は 0 にリセットされます。

TA_HWMEMFILTERS: 0 <= *num*

この属性は、キュー・スペースが最後にオープンされた後に、キュー・スペースでフィルタ用に使用される最大バイト数を指定します。キュー・スペースの状態が CLEaning に設定されると、この値は 0 にリセットされます。

TA_HWMEMNONPERSIST: 0 <= *num*

キュー・スペースが最後にオープンされた後に、一時的メッセージによって使用される最大メモリ容量 (バイト)。キュー・スペースの状態が CLEaning に設定されると、この値は 0 にリセットされます。

TA_HWMEMOVERFLOW: 0 <= *num*

この属性は、キュー・スペースが最後にオープンされた後に、キュー・スペースのオーバフロー・メモリで使用される最大バイト数を指定します。キュー・スペースの状態が CLEaning に設定されると、この値は 0 にリセットされます。

TA_HWMSG: 0 <= *num*

キュー・スペースが最後にオープンされてからの特定の時点でキュー・スペース内に存在する最大メッセージ数。キュー・スペースの状態が CLEaning に設定されると、この値は 0 にリセットされます。

TA_HWOWNERS: 0 <= *num*

この属性は、キュー・スペースが最後にオープンされた後に、キュー・スペースに同時に到達する所有者の最大数を指定します。キュー・スペースの状態が CLEaning に設定されると、この値は 0 にリセットされます。

TA_HWPROC: 0 <= *num*

キュー・スペースが最後にオープンされた後に、キュー・スペースに同時にアタッチされる最大プロセス数。キュー・スペースの状態が CLEaning に設定されると、この値は 0 にリセットされます。

TA_HWQUEUES: 0 <= *num*

キュー・スペースが最後にオープンされてからの特定の時点でキュー・スペース内に存在する最大キュー数。キュー・スペースの状態が CLEaning に設定されると、この値は 0 にリセットされます。

TA_HWTMPQUEUES: 0 <= *num*

この属性は、キュー・スペースが最後にオープンされた後に、キュー・スペースで同時にオープンされる一時キューの最大数を指定します。キュー・スペースの状態が CLEaning に設定されると、この値は 0 にリセットされます。

TA_HWTRANS: 0 <= *num*

キュー・スペースが最後にオープンされてからの特定の時点でキュー・スペースを使用する未処理トランザクションの最大数。キュー・スペースが複数のアプリケーションからアクセスされる場合、TUXCONFIG 環境変数で指定されるアプリケーションだけではなく、すべてのアプリケーションを含む値になります。キュー・スペースの状態が CLEaning に設定されると、この値は 0 にリセットされます。

TA_IPCKEY: 32769 <= *num* <= 262143

キュー・スペースの共用メモリにアクセスするときに使用する IPC キー。

TA_MAXACTIONS: 0 <= *num*

この属性は、BEA Tuxedo インフラストラクチャのキューイング・サービス・コンポーネントが同時に処理できる追加アクション数を指定します。ブロッキング操作の発生時に追加操作を利用できる場合、ブロッキング操作は条件を満たす状態になるまで保留されるように設定されます。ブロッキング操作が保留されると、ほかの操作要求を処理できます。ブロッキング操作が完了すると、その操作に関連する操作は続く操作でも実行できるようになります。システムでは、キュー・スペースにアタッチ可能なプロセスの数と同じ数だけ操作が予約されているため、それぞれのキュー・マネージャ・プロセスは少なくとも 1 つブロッキング操作を所有できます。システムによって予約されているブロッキング操作の数を超える場合、管理者は予約数より多くの追加ブロッキング操作に対応できるようにシステムを設定することができます。ブロッキング操作が要求された時点ですぐに条件を満たす状態にならず、利用可能な操作もない場合には、操作は失敗します。

TA_MAXCURSORS: 0 <= *num*

この属性は、BEA Tuxedo インフラストラクチャのキューイング・サービス・コンポーネントのユーザが同時に使用できるカーソル数を指定します。カーソルは、キューの操作に使用されます。カーソルを破棄すると、そのカーソル・リソースは次のカーソル作成操作に利用できるようになります。カーソルがアプリケーションによって使用される場合、管理者は同時に割り当てることができるカーソルの最大数に対応するようにシステムを設定する必要があります。ユーザがカーソルを作成する際に利用可能なカーソル・リソースがないと、操作は失敗します。BEA Tuxedo アプリケーションではこの値を調整する必要はありません。この値を調整しても、共用メモリを必要以上に浪費するだけで、BEA Tuxedo アプリケーションには何の効果もありません。

TA_MAXHANDLES: 0 <= *num*

この属性は、BEA Tuxedo インフラストラクチャのキューイング・サービス・コンポーネントのユーザが同時に使用できるハンドル数を指定します。キューイング・サービス API によって操作されるオブジェクトでは、そのオブジェクトにアクセスするためのハンドルが必要です。キューイング・サービス API の呼び出しによりオブジェクトがオープンされると、新しいハンドルが作成されてユーザに返されます。オブジェクト・ハンドルを閉じると、そのハンドルは開かれている次のオブジェクトの操作に利用できるようになります。キューイング・サービス API がアプリケーションによって使用される場合、管理者は、同時にオープンされる最大ハンドル数に対応できるようにシステムをコンフィギュレーションする必要があります。ユーザがキューイング・サービス・オブジェクトを開く際に利用可能なハンドルがないと、操作は失敗します。この値を調整しても、共用メモリを必要以上に浪費するだけで、BEA Tuxedo アプリケーションには何の効果もありません。

TA_MAXMSG: 0 <= *num*

ある時点でキュー・スペースに保存可能な最大メッセージ数。

TA_MAXOWNERS: 0 <= *num*

この属性は、キューイング・サービスのリソースを同時に使用することを許可された、BEA Tuxedo インフラストラクチャの認証済みユーザの追加数を指定します。開いているハンドルの数に関係なく、各ユーザに 1 つのオーナ・レコードがあります。開いているハンドルがない場合、次のユーザがオーナ・レコードを使用できます。システム

では、操作数と同じ数だけオーナが予約されているため、異なるオーナが各操作を開始できます。同時にキューイング・サービス・リソースを使用できるシステムによって予約されているオーナ数を超える場合、管理者は予約数より多くの追加オーナに対応できるようにシステムを設定することができます。ユーザがハンドルを開こうとした時点で開いているハンドルがなく、利用できるオーナがないと、操作は失敗します。この値を調整しても、共用メモリを必要以上に浪費するだけで、BEA Tuxedo アプリケーションには何の効果もありません。

TA_MAXPAGES: 0 <= *num*

キュー・スペース内のすべてのキューに対する最大ディスク・ページ数。TA_MAXPAGES 属性が増加するごとに、新しいエクステントが割り当てられます (TA_CUREXTENT を参照)。この属性に小さい値を指定してページ数を減少させることはできません。この場合、エラーが報告されます。

TA_MAXPROC: 0 <= *num*

キュー・スペースに追加可能な最大プロセス数。

TA_MAXQUEUES: 0 <= *num*

ある時点でキュー・スペースに保存可能な最大キュー数。

TA_MAXTMPQUEUES: 0 <= *num*

この属性は、BEA Tuxedo インフラストラクチャのキューイング・サービス・コンポーネントで同時にオープンできる一時キューの数を指定します。一時キューを使用すると、管理者はアプリケーションで使用する各キューを設定する必要がなくなります。一時キューは、動的な自己設定型アプリケーションによって使用されます。一時キューには、永続的メッセージは登録されません。一時キューへのすべてのハンドルが閉じると、一時キューのリソースは次の一時キューの作成で使用できるようになります。一時キューがアプリケーションによって使用される場合は、管理者は同時にアクティブにできる一時キューの最大数に対応できるようにシステムを設定する必要があります。
ユーザが一時キューを開く際に、利用可能な一時キューのリソースがないと、操作は失敗します。この値を調整しても、共用メモリを必要以上に浪費するだけで、BEA Tuxedo アプリケーションには何の効果もありません。

TA_MAXTRANS: 0 <= *num*

キュー・スペースで同時にアクティブにできる最大トランザクション数。

TA_MEMFILTERS: 0 <= *num*

この属性は、ユーザ定義フィルタのコンパイル表現を格納するために、共用メモリで確保するメモリ領域のサイズを指定します。メモリ・サイズはバイト単位で指定します。フィルタは、キューからのメッセージの取り出し操作やカーソル操作においてメッセージを選択する際に、BEA Tuxedo インフラストラクチャのキューイング・サービス・コンポーネントによって使用されます。いろいろな文法を使用して指定されたフィルタは、BEA Tuxedo インフラストラクチャの通常の形式にコンパイルされて、共用メモリに格納されます。フィルタは、コンパイル時に返されるハンドルによって参照されます。フィルタを破棄すると、そのフィルタが使用していたメモリを、次のコンパイル済みフィルタで利用できるようになります。フィルタがアプリケーションによって定義される場合、管理者は同時にコンパイルできるフィルタの最大数に対応できるようにシステムを設定する必要があります。ユーザが新しいフィルタを作成する際に、コンパイル後のフィルタに割り当てるための十分なメモリがないと、操作は失敗します。この値を調整しても、共用メモリを必要以上に浪費するだけで、BEA Tuxedo アプリケーションには何の効果もありません。

TA_MEMNONPERSIST: 0 <= *num* [bB]

この属性は、キュー・スペース内のすべてのキューの一時的メッセージを格納するために、共用メモリに確保する領域のサイズを指定します。メモリのサイズは、バイト (b) またはブロック (B) で指定します。ここでは、ブロックのサイズはディスク・ブロックのサイズと等しくなります。[bB] 接尾辞は指定してもしなくても構いませんが、指定しなかった場合はデフォルトでブロック (B) になります。

この属性の値をバイト (b) で指定すると、システムはその値をページあたりのバイト数で割り (ページ・サイズはディスク・ページ・サイズと等しい)、結果を最近値の整数に切り捨て、そのページ数のメモリを割り当てます。たとえば、ページ・サイズを 1024 バイト (1KB) として考えると、要求された値が 2000b の場合は 1 ページ分 (1024 バイト) のメモリ割り当てが行われ、要求された値が 2048b の場合は 2 ページ分 (2048 バイト) のメモリ割り当てが行われます。ページあたりのバイト数より小さい値を要求すると、0 ページ (0 バイト) が割り当てられます。

この属性の値をブロック (B) で指定し、1 メモリ・ブロックが 1 メモリ・ページと等しいとすると、システムは指定した値と同じページ数

を割り当てます。たとえば、要求された値が 50B の場合、50 ページ分のメモリ割り当てが行われます。

TA_MEMNONPERSIST が変更されると、指定されたキュー・スペース内のすべての一時的メッセージが永久に失われます。

キュー・スペースの TA_MEMNONPERSIST がゼロ (0) の場合、一時的メッセージ用の領域は確保されません。この場合、一時的メッセージをキューに登録しようとしても失敗します。たとえば、メッセージに対してサービスの配信の品質が指定されずに、ターゲット・キューの T_APPQ クラスの TA_DEFDELIVERYPOLICY 属性が NONPERSIST に設定された場合などです。非永続的配信では、メモリ領域のすべてが使用されている場合や分割されたメッセージをキューに登録できない場合、メッセージ用の永続ストレージが十分にあっても、メッセージのキュー登録は異常終了します。同様に、永続ストレージのすべてが使用されている場合や分割されたメッセージをキューに登録できない場合、メッセージ用の非永続ストレージが十分にあっても、メッセージの登録操作は異常終了します。

TA_MEMOVERFLOW: 0 <= *num*

この属性は、割り当て済みの共用メモリの一部または全部が使用されたピーク負荷状況に対応するために共用メモリ内に確保する領域のサイズを指定します。メモリ・サイズはバイト単位で指定します。追加オブジェクトは、先着順でこの追加メモリから割り当てられます。追加メモリで作成されたオブジェクトを閉じるか破棄すると、次に共用メモリ・リソース不足が発生するときに備えてメモリは解放されます。この追加メモリ領域では、設定数より多くのオブジェクトを生成できますが、特定のオブジェクトに対していつでも使用できるとは限りません。現在このメモリ領域を使用できるのは、アクション、ハンドル、カーソル、オーナ、一時キュー、タイマー、およびフィルタだけです。

TA_MEMSYSTEMRESERVED: 0 <= *num*

この属性は、キューイング・サービス・システムが使用するために共用メモリ内に確保するメモリ容量の合計 (バイト) を指定します。

TA_MEMTOTALALLOCATED: 0 <= *num*

この属性は、すべてのキューイング・サービス・オブジェクトに割り当てられる共用メモリ容量の合計 (バイト) を指定します。

TA_PERCENTINIT: 0 <= *num* <= 100

キュー・スペース用に初期化されるディスク領域のパーセンテージ。

T_APPQTRANS クラスの定義

概要 T_APPQTRANS クラスは、アプリケーション・キューに関連付けられるトランザクションの実行時属性を表します。

制限事項 すべてのキー・フィールドを未設定にすると、このクラスのインスタンスをすべて検索できません。反対に、1 つのアプリケーション・キュー・スペースを明示的に指定するには、適切なキー・フィールドを指定する必要があります。たとえば、tpcall() を使用して、TA_XID を除くすべてのキー・フィールドを要求で設定した場合、指定されたキュー・スペースに対応する T_APPQTRANS オブジェクトがすべて検索されます。

このクラスのオブジェクトで表現されるトランザクションは必ずしも検索対象のアプリケーションに関連付けられないので注意してください。トランザクションは実際には別のアプリケーションに属していたり、別のアプリケーションに影響を与えたりする場合があるため、トランザクションをヒューリスティックにコミットまたはアボートするときには注意が必要です。TA_XID 属性の値は、アプリケーション間で一意であるとは限りません。

属性表

表 10APPQ_MIB(5):T_APPQTRANS **クラス定義の属性表**

属性 [a]	**タイプ**	**パーミッション**	**値**	**デフォルト値**
TA_XID(k)(*)	string	R--R--R--	*string*[1..78]	N/A
TA_APPQSPACENAME(k)(*)	string	r--r--r--	*string*[1..15]	N/A
TA_QMCONFIG(k)(*)	string	r--r--r--	*string*[1..78]	N/A
TA_LMID(k)(*)	string	r--r--r--	*string*[1..30]	N/A
TA_STATE[b]	string	R-XR-XR--	GET: "{ACT | ABY | ABD | COM | REA | DEC | HAB | HCO}"	N/A
			SET: "{HAB | HCO}"	N/A

(k)—GET キー・フィールド [c]
(*)— 必須の SET キー・フィールド

a. T_APPQTRANS クラスの属性はすべてローカルです。

b. T_APPQTRANS オブジェクトのすべての操作 (GET と SET) は、関連付けられているキュー・スペースを自動的にオープンします。つまり、キュー・スペースの状態が OPEn または ACTive になっていない場合、暗黙的に OPEn に設定します。キュー・スペースが大きいと、この操作は時間がかかります。
c. 1 つのアプリケーション・キュー・スペースを明示的に指定するには、GET 操作で適切なキー・フィールドを指定する必要があります。

属性の意味

TA_XID: string[1..78]

tx_info() から返され、文字列表現にマップされるトランザクション識別子。等号比較の場合を除き、ユーザはこのフィールドのデータを直接解釈することはできません。

TA_APPQSPACENAME: *string*[1..15]

トランザクションに関連付けるアプリケーション・キュー・スペースの名前。

TA_QMCONFIG: *string*[1..78]

アプリケーション・キュー・スペースが存在するファイルまたはデバイスの絶対パス名。

TA_LMID: *string*[1..30] (no comma)

アプリケーション・キュー・スペースが存在する論理マシンの識別子。

TA_STATE:

GET: {ACTive | ABortonlY | ABorteD | COMcalled | REAdy | DECided | HAbord | HCommit}

GET 操作は、選択したトランザクションに関する実行時情報を検索します。以下に、GET 要求に対する応答で返される TA_STATE の意味を示します。すべての状態は ACTive と同等で、パーミッションのチェックに使用します。

ACTive	トランザクションはアクティブです。
ABortonlY	トランザクションはロールバックされるものと識別されています。
ABorteD	トランザクションはロールバックされるものと識別され、ロールバックが開始されました。

COMcalled	トランザクションのイニシエータが tpcommit() を呼び出し、2 フェーズ・コミットの第 1 フェーズが開始されました。
REAdy	検索サイトの参加グループすべてが 2 フェーズ・コミットの第 1 フェーズを正常に完了し、コミット可能な状態です。
DECided	2 フェーズ・コミットの第 2 フェーズが開始されました。
SUSpended	トランザクションのイニシエータがトランザクション処理を中断しました。

SET: {HABort | HCOmmit}

SET 操作は、選択したトランザクションの状態を更新します。以下に、SET 要求によって返される TA_STATE の意味を示します。これ以外の状態を設定することはできません。

HABort	トランザクションをヒューリスティックにアボートします。正常終了すると、オブジェクトの状態は HABort になります。
HCOmmit	トランザクションをヒューリスティックにコミットします。正常終了すると、オブジェクトの状態は HCOmmit になります。

APPQ_MIB(5) に関する追加情報

移植性 BEA Tuxedo システムの MIB を使用した管理作業をサポートするために必要な既存の FML32 および ATMI 関数、さらにこのリファレンス・ページに定義するヘッダ・ファイルとフィールド・テーブルは、すべてのサポート対象ネイティブ・プラットフォームとワークステーション・プラットフォームで使用可能です。

相互運用性 この MIB は、BEA Tuxedo 6.0 以降が実行されているサイト (ネイティブおよび Workstation の両方) からのみアクセスできます。

BEA Tuxedo 6.0 より前のリリースが実行されているサイトがアプリケーション内でアクティブ化された場合、この MIB による管理アクセスは次のとおり制限されます。

- SET 操作は許可されません。
- リリース 6.0 より前のサイトのローカル情報にはアクセスできません。アクセス中のクラスがグローバル情報も持っている場合、グローバル情報のみが返されます。それ以外の場合は、エラーが返されます。

リリースが異なるサイト (共にリリース 6.0 以降) を相互運用する場合、当該リリースの MIB マニュアル・ページに定義されるように、旧サイト上の情報はアクセスおよび更新可能で、以降のリリースで利用可能な情報のサブセットとなります。

使用例 以下に、アプリケーション・キュー・スペース、キュー、メッセージ、およびトランザクションに対する各種操作の実行方法を示すコードを示します。

各コードの前には、次のように、FML32 型付きバッファを割り当てるコードを追加してください。

```
rqbuf = tpalloc("FML32", NULL, 0);
```

バッファにデータを入力したら、各コードの後には、次のような、要求を送信し、応答を受信するコードを追加します。

```
flags = TPNOTRAN | TPNOCHANGE | TPSIGRSTRT;
rval = tpcall(".TMIB", rqbuf, 0, rpbuf, rplen, flags);
```

詳細については、MIB(5) を参照してください。

フィールド・テーブル

属性フィールド識別子にアクセスするには、フィールド・テーブル tpadm が必要です。そのためには、次のようにシェルで入力します。

```
$ FIELDTBLS=tpadm
$ FLDTBLDIR=${TUXDIR}/udataobj
$ export FIELDTBLS FLDTBLDIR
```

ヘッダ・ファイル

次のヘッダ・ファイルが必要です。

```
#include <atmi.h>
#include <fml32.h>
#include <tpadm.h>
```

ライブラリ

```
${TUXDIR}/lib/libtmib.a, ${TUXDIR}/lib/libqm.a,
${TUXDIR}/lib/libtmib.so.<rel>, ${TUXDIR}/lib/libqm.so.<rel>,
${TUXDIR}/lib/libqm.lib
```

buildclient を使用するときには、ライブラリを手動でリンクする必要があります。次のように使用してください。-L${TUXDIR}/lib -ltmib -lqm

アプリケーション・キュー・スペースの作成

通常、アプリケーション・キュー・スペースを作成するには 2 つの操作が必要です。最初の操作ではキュー・スペースを割り当てる BEA Tuxedo システム・デバイスを作成し、次の操作ではキュー・スペース自体を作成します。

```
/* 上記を参照してバッファを割り当てる */

/* SITE1 に新しいデバイスを作成するための要求を作成 */
Fchg32(rqbuf, TA_OPERATION, 0, "SET", 0);
Fchg32(rqbuf, TA_CLASS, 0, "T_DEVICE", 0);
Fchg32(rqbuf, TA_STATE, 0, "NEW", 0);
Fchg32(rqbuf, TA_CFGDEVICE, 0, "/dev/q/dsk001", 0);
Fchg32(rqbuf, TA_LMID, 0, "SITE1", 0);
size = 500;
Fchg32(rqbuf, TA_DEVSIZE, 0, (char *)size, 0);

/* 上記を参照して要求を作成 */

/* 再使用のために同じバッファを再初期化する */
Finit32(rqbuf, (FLDLEN) Fsizeof32(rqbuf));

/* キュー・スペースを作成するための要求を作成 */
Fchg32(rqbuf, TA_OPERATION, 0, "SET", 0);
Fchg32(rqbuf, TA_CLASS, 0, "T_APPQSPACE", 0);
Fchg32(rqbuf, TA_STATE, 0, "NEW", 0);
Fchg32(rqbuf, TA_APPQSPACENAME, 0, "QSPACE1", 0);
Fchg32(rqbuf, TA_QMCONFIG, 0, "/dev/q/dsk001", 0);
Fchg32(rqbuf, TA_LMID, 0, "SITE1", 0);
```

```
Fchg32(rqbuf, TA_ERRORQNAME, 0, "errque", 0);
ipckey = 123456;
Fchg32(rqbuf, TA_IPCKEY, 0, (char *)ipckey, 0);
maxmsg = 100;
Fchg32(rqbuf, TA_MAXMSG, 0, (char *)maxmsg, 0);
maxpages = 200;
Fchg32(rqbuf, TA_MAXPAGES, 0, (char *)maxpages, 0);
maxproc = 50;
Fchg32(rqbuf, TA_MAXPROC, 0, (char *)maxproc, 0);
maxqueues = 10;
Fchg32(rqbuf, TA_MAXQUEUES, 0, (char *)maxqueues, 0);
maxtrans = 100;
Fchg32(rqbuf, TA_MAXTRANS, 0, (char *)maxtrans, 0);

/* 上記を参照して要求を作成 */
```

アプリケーション・キュー・スペースへのキューの追加

以下のコードでは、上の例で作成したキュー・スペースに新しいキューを作成します。

```
/* 要求を作成 */
Fchg32(rqbuf, TA_OPERATION, 0, "SET", 0);
Fchg32(rqbuf, TA_CLASS, 0, "T_APPQ", 0);
Fchg32(rqbuf, TA_STATE, 0, "NEW", 0);
Fchg32(rqbuf, TA_APPQNAME, 0, "errque", 0);
Fchg32(rqbuf, TA_APPQSPACENAME, 0, "QSPACE1", 0);
Fchg32(rqbuf, TA_QMCONFIG, 0, "/dev/q/dsk001", 0);
Fchg32(rqbuf, TA_LMID, 0, "SITE1", 0);
Fchg32(rqbuf, TA_APPQORDER, 0, "PRIO", 0);

/* 上記を参照して要求を作成 */
```

アプリケーションが認識しているアプリケーション・キュー・スペースの一覧表示

アプリケーションが認識しているアプリケーション・キュー・スペースを一覧表示するには、2 段階の検索を行います。まず、/Q トランザクション・マネージャ TMS_QM を使用するグループがアプリケーション環境設定から検索され、次に各グループが参照しているキュー・スペースが検索されます。以下のコードは、キュー・スペースを使用する各 GROUP エントリに 1 つの論理マシンが対応付けられていると仮定します (つまり、サーバ移行は未使用)。

コード リスト 10-1 **アプリケーションが認識しているアプリケーション・キュー・スペースの一覧表示**

```
/* すべての TMS_QM グループを検索する要求を作成 */
Fchg32(rqbuf, TA_OPERATION, 0, "GET", 0);
Fchg32(rqbuf, TA_CLASS, 0, "T_GROUP", 0);
```

```
Fchg32(rqbuf, TA_TMSNAME, 0, "TMS_QM", 0);
fldid1 = TA_OPENINFO;
fldid2 = TA_LMID;
Fchg32(rqbuf, TA_FILTER, 0, (char *)fldid1, 0);
Fchg32(rqbuf, TA_FILTER, 0, (char *)fldid2, 1);

/* アプリケーションに参加したと見なして要求を作成 */
rval = tpcall(".TMIB", rqbuf, 0, rpbuf, rplen, flags);

/* TMS_QM グループごとに、キュー・スペースを検索する要求を作成 */
rval = Fget32(*rpbuf, TA_OCCURS, 0, (char *)occurs, NULL);
for (i = 0; i occurs; i++) {


  /* バッファを再初期化して、すべての共通属性を設定 */
  Finit32(rqbuf, (FLDLEN) Fsizeof32(rqbuf));
  Fchg32(rqbuf, TA_OPERATION, 0, "GET", 0);
  Fchg32(rqbuf, TA_CLASS, 0, "T_APPQSPACE", 0);

  /* デバイスとキュー・スペース名を調べるための OPENINFO を取得 */
  /* OPENINFO の形式は、<resource-mgr>:<qmconfig>:<appqspacename> */
  /* Windows の場合は、<resource-mgr>:<qmconfig>;<appqspacename> */
  rval = Fget32(rpbuf, TA_OPENINFO, i, openinfo, NULL);

  /* デバイスは、OPENINFO の 2 つ目のフィールド */
  qmconfig = strchr(openinfo, ':')+ 1;
  /* キュー・スペース名は、OPENINFO の 3 つ目のフィールド */

#if defined(_TMDOWN) || defined(_TM_NETWARE)
#define pathsep ";" /* PATH の区切り文字 */
#else
#define pathsep ":"/* PATH の区切り文字 */
#endif
  appqspacename = strchr(qmconfig, pathsep);
  appqspacename[0] = '\e0'; /* qmconfig をヌルで終了するように指定 */
  appqspacename++; /* ヌルに値を追加 */

  /* APPQSPACENAME と QMCONFIG キーを設定 */
  Fchg32(rqbuf, TA_APPQSPACENAME, 0, appqspacename, 0);
  Fchg32(rqbuf, TA_QMCONFIG, 0, qmconfig, 0);

  /* LMID を取得 ( このグループに対する移行はないものと見なす ) */
  rval = Fget32(rpbuf, TA_LMID, i, lmid, NULL);
  Fchg32(rqbuf, TA_LMID, 0, lmid, 0);

  /* 要求を作成 */
```

```
  rval = tpcall(".TMIB", rqbuf, 0, rpbuf2, rplen2, flags);
}
```

上記のコードでは、キュー・スペースが作成されていても、アプリケーションのコンフィギュレーションに対応する GROUP エントリがないと、キュー・スペースは検索されません。このようなキュー・スペースは、キュー・スペースのキー・フィールド (TA_APPQSPACENAME、TA_QMCONFIG、および TA_LMID) の優先順位が分かっていなければ検索できません。

アプリケーション・キュー内のメッセージの一覧表示

以下のコードでは、論理デバイス SITE1 上のデバイス /dev/q/dsk001 のキュー・スペース QSPACE1 内のキュー STRING にあるメッセージをすべて検索します。

```
/* 要求を作成 */ Fchg32(rqbuf, TA_OPERATION, 0, "GET", 0);
Fchg32(rqbuf, TA_CLASS, 0, "T_APPQMSG", 0);
Fchg32(rqbuf, TA_APPQNAME, 0, "STRING", 0);
Fchg32(rqbuf, TA_APPQSPACENAME, 0, "QSPACE1", 0);
Fchg32(rqbuf, TA_QMCONFIG, 0, "/dev/q/dsk001", 0);
Fchg32(rqbuf, TA_LMID, 0, "SITE1", 0);
/* 上記を参照して要求を作成 */
```

キュー・スペースを使用するトランザクションの一覧表示

以下のコードでは、キュー・スペース QSPACE1 の中の任意のキューを使用するトランザクションをすべて検索します。

```
/* 要求を作成 */ Fchg32(rqbuf, TA_OPERATION, 0, "GET", 0);
Fchg32(rqbuf, TA_CLASS, 0, "T_APPQTRANS", 0);
Fchg32(rqbuf, TA_APPQSPACENAME, 0, "QSPACE1", 0);
Fchg32(rqbuf, TA_QMCONFIG, 0, "/dev/q/dsk001", 0);
Fchg32(rqbuf, TA_LMID, 0, "SITE1", 0);
/* 上記を参照して要求を作成 */
```

ファイル

```
${TUXDIR}/include/tpadm.h
${TUXDIR}/udataobj/tpadm
```

関連項目

tpacall(3c)、tpadmcall(3c)、tpalloc(3c)、tpcall(3c)、tpdequeue(3c)、tpenqueue(3c)、tpgetrply(3c)、tprealloc(3c)、FML 関数の紹介、Fadd、Fadd32(3fml)、Fchg、Fchg32(3fml)、Ffind、Ffind32(3fml)、MIB(5)、TM_MIB(5)

『BEA Tuxedo アプリケーションの設定』

『BEA Tuxedo アプリケーション実行時の管理』

『C 言語を使用した BEA Tuxedo アプリケーションのプログラミング』

『FML を使用した BEA Tuxedo アプリケーションのプログラミング』

AUTHSVR(5)

名前 AUTHSVR— サーバ提供のユーザ単位の認証

形式 AUTHSVR SRVGRP="*identifier*" SRVID=*number other_parms* CLOPT="-A"

説明 AUTHSVR は、BEA Tuxedo に用意されている、認証サービスを備えたサーバです。このサーバを保護されたアプリケーションで使用することにより、クライアントがアプリケーションに参加するときにユーザ単位の認証を行うことができます。このサーバは、アプリケーションへのアクセスを要求しているクライアント・プロセスのための TPINIT 型付きバッファを含むサービス要求を受け付けます。TPINIT 型付きバッファのデータ・フィールドをユーザのパスワードとして使用し、そのパスワードを設定済みパスワードと比較することにより、要求の妥当性をチェックします。要求が妥当であると認められると、クライアントが使用するためのチケットとしてアプリケーション・キーが返されます。

アプリケーション・キーの設定には、tpreturn(3c) の rcode パラメータが使用されます。このパラメータは、妥当性検査に合格するか、パーミッションが拒否されたときに、tpinit(3c) を呼び出したコードに (tpurcode で) 返されます。

AUTHSVR の詳細については、64 ページの「AUTHSVR に関する追加情報」を参照してください。

SECURITY USER_AUTH

SECURITY が USER_AUTH に設定されている場合は、強制的にユーザ単位での認証が実行されます。UBBCONFIG ファイルの RESOURCES セクションの AUTHSVC パラメータを使用して、アプリケーションに対する認証サービスの名前を設定することができます。たとえば、次の AUTHSVC パラメータ設定では、SECURITY が USER_AUTH に設定されている場合に AUTHSVR によって宣言されるサービス (AUTHSVC) が指定されます。

```
*RESOURCES
SECURITY    USER_AUTH
AUTHSVC     AUTHSVC
```

AUTHSVC パラメータを設定しない場合、認証サービスはデフォルトとによって AUTHSVC となります。

デフォルトでは、アプリケーションの APPDIR 変数で定義される最初のパス名で参照されるディレクトリのファイル tpusr はパスワード情報の検索に使用されますが、このファイルが存在しない場合は /etc/passwd が使用されます。ただし、このファイルはシャドー・パスワード・ファイルを使用しているシステムでは正しく使用できません。ファイルは、サーバのコマンドライン・オプションの "-f *filename*" オプションでファイル名を指定することによって変更することができます (例 : CLOPT="-A -- -f /usr/tuxedo/users")。マスタ・マシンからコンフィギュレーションで指定された他のマシンへのユーザ・ファイルの複製転送は、$APPDIR/tpusr を使用した場合にのみ実行されます。

ユーザ・ファイルでは、(与えられた名前と) 一致するユーザ名とクライアント名が検索されます。ユーザ・ファイルには、4 つのタイプのエントリがあります。これらをユーザの妥当性検査を行う際の一致の優先度の順に並べると、次のようになります。

1. 正確なユーザ名 / 正確なクライアント名
2. ワイルドカード (*) を使用したユーザ名 / 正確なクライアント名
3. 正確なユーザ名 / ワイルドカード (*) を使用したクライアント名
4. ワイルドカード (*) を使用したユーザ名 / ワイルドカード (*) を使用したクライアント名

認証要求は、最初に一致するパスワード・ファイルのエントリに対してのみ認証されます。これらのセマンティクスを使用すれば、同じユーザが (通常異なるクライアント名の) 複数のエントリ名を持つことができ、ユーザ名にワイルドカードを使用できます。これらのセマンティクスを使用できるのは、tpaddusr()、tpdelusr()、および tpmodusr() を利用してユーザ・ファイルを管理する場合です。ただし、これらのセマンティクスを使用した場合、ACL と MANDATORY_ACL のセマンティクスとの互換性はなく、これらのセキュリティ・レベルへの移行は困難になります。ACL セキュリティとの互換性のため制限的なセマンティクスを得るには、tpusradd()、tpusrdel()、および tpusrmod() の各プログラムを利用してユーザ・ファイルを管理する必要があります。

注記　tpusradd()、tpusrdel()、および tpusrmod() を使用するには、ターゲット・アプリケーションの SECURITY を USER_AUTH、ACL、または MANDATORY_ACL に設定する必要があります。そのようにしない場合、これらのプログラムを使用しようとするとエラーが返されます。

認証要求を処理する際には、特殊なクライアント名の値、つまり tpsysadm (システム管理者) と tpsysop (システム・オペレータ) は AUTHSVR(5) によって特別に扱われます。これらの値は、ユーザ・ファイルのワイルドカードを利用したクライアント名と一致させることはできません。

AUTHSVR によって返されるアプリケーション・キーは、ユーザ ID です。このアプリケーション・キーは、TPSVCINFO というデータ構造の *appkey* エレメントに含まれるすべてのサービスに渡されます。

標準仕様の AUTHSVR は、システムの一部として ${TUXDIR}/bin/AUTHSVR に格納された状態で出荷され、上記で説明したセマンティクスを持っています。ソース・コードのサンプルは、${TUXDIR}/lib/AUTHSVR.c に収められています。AUTHSVR の代わりに、(Kerberos などを利用して) それぞれのアプリケーションに適した方法でユーザやユーザ・データ (パスワードは不可) の妥当性を検査するアプリケーション認証サーバを使用することができます。AUTHSVR の代わりにほかのアプリケーション認証サーバを使用する場合、このリファレンス・ページで後述する警告に特に留意してください。また、使用する認証サービスが (それぞれのサービスに渡す) アプリケーション・キーとして返す値も、それぞれのアプリケーションによって異なります。

tpsysadm と tpsysop に対応するアプリケーション・キーは、それぞれ 0x80000000 と 0xC0000000 です。

SECURITY ACL または MANDATORY_ACL

SECURITY が ACL または MANDATORY_ACL に設定されている場合、ユーザ単位の認証が強制的に実行され、サービスや、アプリケーションのキュー、イベントにアクセスするためのアクセス管理リストがサポートされます。UBBCONFIG ファイルの RESOURCES セクションの AUTHSVC パラメータを使用して、アプリケーションに対する認証サービスの名前を設定することができます。たとえば、次の AUTHSVC パラメータ設定では、SECURITY が ACL または MANDATORY_ACL に設定されている場合に AUTHSVR によって宣言されるサービス (AUTHSVC) が指定されます。

```
*RESOURCES
SECURITY    ACL
AUTHSVC     ..AUTHSVC
```

AUTHSVC パラメータを設定しない場合、認証サービスはデフォルトによって ..AUTHSVC となります。

注記　AUTHSVR は、SECURITY が USER_AUTH に設定される場合に認証サービス AUTHSVC を宣言し、SECURITY が ACL または MANDATORY_ACL に設定される場合に認証サービス ..AUTHSVC を宣言します。AUTHSVC と ..AUTHSVC は、同じ認証サービスを指します。

ユーザ・ファイルは、$APPDIR/tpusr でなければなりません。このファイルは、マスタ・マシンからコンフィギュレーションで指定された他のアクティブ・マシンに自動的に複製転送されます。マスタ・マシンでは、AUTHSVR の 1 つのインスタンスが実行されている必要があります。コンフィギュレーションで指定された別のアクティブ・マシンでは、AUTHSVR の新たなコピーを実行できます。

ユーザ・ファイルでは、(与えられた名前と) 一致するユーザ名とクライアント名が検索されます。ユーザ名は、ユーザ・ファイルのエントリと正確に一致している必要があります。クライアント名は正確に一致している必要がありますが、代替手段としてユーザ・ファイルのクライアント名の値をあらゆるクライアント名に該当するワイルドカード (*) として指定する方法も利用できます。ユーザ・ファイルのエントリは 1 人のユーザにつき 1 つだけで、ユーザ名にワイルドカードを使用することはできません。ユーザ・ファイルは、tpusradd()、tpusrdel()、および tpusrmod() の各プログラム、グラフィカル・ユーザ・インターフェイス、または管理インターフェイスを使用して管理できます。

認証要求を処理する際には、特殊なクライアント名の値、つまり tpsysadm (システム管理者) と tpsysop (システム・オペレータ) は AUTHSVR(5) によって特別に扱われます。これらの値は、ユーザ・ファイルのワイルドカードを利用したクライアント名と一致させることはできません。

AUTHSVR によって返されるアプリケーション・キーは、下位 17 ビットのユーザ ID と、それに続く 14 ビットのグループ ID で構成されます (上位ビットは管理キーとして予約されています)。tpsysadm と tpsysop に対応するアプリケーション・キーは、それぞれ 0x80000000 と 0xC0000000 です。このアプリケーション・キーは、TPSVCINFO というデータ構造の *appkey* エレメントに含まれるすべてのサービスに渡されます。

SECURITY ACL または MANDATORY_ACL の場合、システムの一部として ${TUXDIR}/bin/AUTHSVR に収められている AUTHSVR を使用する必要があります。

AUTHSVR に関する追加情報

使用法 警告 : ${TUXDIR}/lib/AUTHSVR.c は、${TUXDIR}/bin/AUTHSVR を生成するために使用されるソースではありません (この実行可能ファイルは破壊しないでください)。独自の AUTHSVR を使用する場合は、${APPDIR} にインストールしてください。

移植性 AUTHSVR は、BEA Tuxedo に付属のサービスとして非 Workstation プラットフォームでサポートされます。

使用例

```
# Using USER_AUTH
*RESOURCES
SECURITY  USER_AUTH
AUTHSVC   AUTHSVC

*SERVERS
AUTHSVR SRVGRP="AUTH" CLOPT="-A -- -f /usr/tuxedo/users" \
 SRVID=100 RESTART=Y GRACE=0 MAXGEN=2
#
#
# Using ACLs
*RESOURCES
SECURITY  ACL
AUTHSVC   ..AUTHSVC

*SERVERS
AUTHSVR SRVGRP="AUTH" SRVID=100 RESTART=Y GRACE=0 MAXGEN=2
#
#
# Using a custom authentication service
*RESOURCES
SECURITY  USER_AUTH
AUTHSVC   KERBEROS

*SERVERS
KERBEROSSVR SRVGRP="AUTH1" SRVID=100 RESTART=Y GRACE=0 MAXGEN=2
```

関連項目 tpaddusr(1)、tpusradd(1)、UBBCONFIG(5)

『BEA Tuxedo アプリケーションの設定』

『BEA Tuxedo アプリケーション実行時の管理』

『C 言語を使用した BEA Tuxedo アプリケーションのプログラミング』

compilation(5)

名前 compilation—BEA Tuxedo ATMI システムのアプリケーション・コンポーネントのコンパイル命令

機能説明 アプリケーションのクライアントとサーバ、および BEA Tuxedo システムにリンクされているサブルーチンをコンパイルする場合、プログラマは以下のことを知っておく必要があります。

- インクルードするヘッダ・ファイルとその指定順序
- 設定およびエクスポートする環境変数
- アプリケーション・モジュールのコンパイルに使用するユーティリティ

コード・モジュールの記述が済み、実行可能プログラムを構築する準備が整ったプログラマは、以下の作業を行う必要があります。

- ソース・ファイルのコンパイル
- 必要なライブラリを使用した実行可能ファイルのリンク

BEA Tuxedo システムには、この両方の操作をクライアント・モジュールとサーバ・モジュールで実行するための 2 つのコマンド、buildclient() と buildserver() が用意されています。いずれかのコマンドを実行して両方の操作を実行する場合、ファイルをリンクするためのライブラリを必ずコマンド行で指定してください。詳細については、『BEA Tuxedo コマンド・リファレンス』の buildclient(1) または buildserver(1) を参照してください。

リンクを行うには buildclient または buildserver を実行する必要がありますが、より柔軟なコンパイル方法も用意されています。必要に応じて、自分が選択したコンパイル・コマンドを使用してファイルをコンパイルし、次に buildclient または buildserver を実行してリンク編集を行うこともできます。

このリファレンス・ページの残りの部分では、各種のプログラムで必要なヘッダ・ファイルと環境変数を示します。

基本的な BEA Tuxedo システム

UNIX システムのヘッダ・ファイルは、必ず BEA Tuxedo システムのヘッダ・ファイルの前にインクルードします。一般的に使用される UNIX システムのヘッダ・ファイルは stdio.h と ctype.h です。

環境変数

以下の環境変数を設定およびエクスポートします。

TUXDIR
: BEA Tuxedo ソフトウェアが存在する最上位ディレクトリを指定します。

PATH
: $TUXDIR/bin を含むように指定します。

ULOGPFX
: 中央イベント・ログのファイル名に付ける接頭辞。デフォルトでは、ULOGPFX の値は ULOG です。

状況 . .	最初に設定およびエクスポートする環境変数 . .
次のコマンドを実行する ■ buildclient(1) ■ buildserver(1)	■ TUXDIR— サーバで常に必要。ネイティブ・クライアントでも必要 ■ CC— デフォルト以外のコンパイラを使用する場合 ■ CFLAGS— コンパイラに渡すフラグを指定する場合
デフォルトのルーチンまたは妥当性検査ルーチンが FML フィールドを参照する	■ FIELDTBLS— カンマで区切ったフィールド・テーブル・ファイルのリスト ■ FLDTBLDIR—FIELDTBLS ファイルを検索するためのコロンで区切られたディレクトリのリスト
サーバを実行する	TUXCONFIG— バイナリ・コンフィギュレーション・ファイルの絶対パス名 (デフォルトはカレント・ディレクトリ)

状況 . .	最初に設定およびエクスポートする環境変数 . .
■ アプリケーションに対してセキュリティをオンにする ■ 次のシステム提供クライアントに対して入力を間接的に (標準入力以外のソースから) 提供する tmadmin(1)、tmconfig または wtmconfig (tmconfig、wtmconfig(1) を参照)、あるいは ud または wud (ud、wud(1) を参照)	■ APP_PW— アプリケーション・パスワード ■ USR_PW— ユーザ・パスワード
ワークステーション・クライアントを実行する	■ WSENVFILE— 環境変数の設定値を収めたファイル ■ WSDEVICE— 接続に使用するネットワーク・デバイス ■ WSTYPE— ワークステーションのマシン・タイプ

注記　これらの変数の詳細については、『C 言語を使用した BEA Tuxedo アプリケーションのプログラミング』、『COBOL を使用した BEA Tuxedo アプリケーションのプログラミング』、および『BEA Tuxedo アプリケーションの設定』を参照してください。

共用ライブラリを使用してシステムが構築された場合、クライアントを実行する前に、共用ライブラリの位置を定義する環境変数を設定する必要があります。

プラットフォームの種類 . .	設定する環境変数 . .
HP-UX と AIX 以外のすべてのプラットフォーム	LD_LIBRARY_PATH=$TUXDIR/lib
HP-UX	SHLIB_PATH=$TUXDIR/lib
AIX	LIBPATH=$TUXDIR/lib

注記　サーバ用のオプションの詳細については、servopts(5) リファレンス・ページを参照してください。

FML プログラム

FML 関数を呼び出す C プログラムには、以下のヘッダ・ファイルをここに示す順序でインクルードします。

```
#include <UNIX_header_files> ( アプリケーションで必要な場合 )
#include "fml.h"
```

FML プログラムのコンパイル

FML の関数を含むプログラムをコンパイルするには、次のようにコマンドを実行します。

```
cc pgm.c -I $TUXDIR/include -L $TUXDIR/lib -lfml -lengine -o pgm
```

ここで、*pgm* は実行可能ファイルの名前です。

-L オプションがローカルでサポートされていない場合、代わりに次のコマンドを使用します。

```
cc pgm.c -I $TUXDIR/include $TUXDIR/lib/libfml.a $TUXDIR/lib/libengine.a -o pgm
```

注記　ライブラリの指定順序は重要です。上に示したとおりの順序で指定してください。

FML VIEWS のコンパイル

FML VIEW コンパイラを使用するには、次のようにコマンドを実行します。

```
viewc view_file
```

ここで *view_file* は、VIEW 用のソース記述が格納されている 1 つまたは複数のファイルです。

注記　viewc は、C コンパイラを呼び出します。使用するコンパイラを指定する場合は、環境変数 CC を使用できます。コンパイラにパラメータのセットを渡す場合は、環境変数 CFLAGS を使用できます。

FML の環境変数

FML を使用するアプリケーションを実行するときは、以下の環境変数を設定してエクスポートします。

FIELDTBLS
: カンマで区切ったフィールド・テーブル・ファイルのリスト

FLDTBLDIR
: FIELDTBLS ファイルを検索するためのコロンで区切ったディレクトリのリスト

viewc を実行するときは、以下の環境変数を設定してエクスポートします。

FIELDTBLS
: カンマで区切ったフィールド・テーブル・ファイルのリスト

FLDTBLDIR
: FIELDTBLS ファイルを検索するためのコロンで区切ったディレクトリのリスト

VIEWDIR
: VIEW ファイルが格納されているディレクトリ。デフォルトはカレント・ディレクトリです。

関連項目 buildclient(1)、buildserver(1)、viewc、viewc32(1)
UNIX システムのリファレンス・マニュアルの cc(1)、mc(1)

DMADM(5)

名前 DMADM― ドメイン管理サーバ

形式
```
DMADM SRVGRP = "identifier"
SRVID = "number"
REPLYQ = "N"
```

機能説明 Domains 管理サーバ (DMADM) は、BDMCONFIG ファイルに実行時にアクセスするための BEA Tuxedo システム提供のサーバです。

DMADM は、DMADMGRP などのグループ内で動作するサーバとして、UBBCONFIG の SERVERS セクションで記述されます。このグループ内で動作する DMADM は 1 つだけで、応答キューが存在していてはいけません (REPLYQ を N に設定する必要がある)。

SERVERS セクションでは、DMADM サーバのパラメータとして、SEQUENCE、ENVFILE、MAXGEN、GRACE、RESTART、RQPERM、および SYSTEM_ACCESS も指定できます。

BDMCONFIG 環境変数は、バイナリ形式の DMCONFIG ファイルが入っているファイルのパス名に設定する必要があります。

移植性 DMADM は、サポートされているすべてのサーバ・プラットフォームで BEA Tuxedo システム提供のサーバとしてサポートされます。

相互運用性 DMADM は、BEA Tuxedo リリース 5.0 以降にインストールする必要があります。リリース 5.0 のゲートウェイが存在するドメイン内のほかのマシンの場合は、リリース 4.1 以降でも構いません。

使用例 次の例は、UBBCONFIG ファイルで管理サーバとゲートウェイ・グループを定義する方法を示しています。この例では、GWTDOMAIN ゲートウェイ・プロセスを使用して別の BEA Tuxedo ドメインと通信します。BEA TOP END システムとの相互運用性を実現するには、GWTOPEND ゲートウェイ・プロセスを使用してください。GWTOPEND ゲートウェイ・プロセスの詳細と GWTOPEND の使用例については、GWTOPEND(5) を参照してください。

```
#
*GROUPS
```

```
DMADMGRP LMID=mach1 GRPNO=1
gwgrp    LMID=mach1 GRPNO=2
#
*SERVERS
DMADM SRVGRP="DMADMGRP" SRVID=1001 REPLYQ=N RESTART=Y GRACE=0
GWADM SRVGRP="gwgrp" SRVID=1002 REPLYQ=N RESTART=Y GRACE=0
GWTDOMAIN SRVGRP="gwgrp" SRVID=1003 RQADDR="gwgrp" REPLYQ=Y RESTART=Y MIN=1 MAX=1
```

関連項目 dmadmin(1)、tmboot(1)、DMCONFIG(5)、DMCONFIG for GWTOPEND(5)、GWADM(5)、GWTOPEND(5), servopts(5)、UBBCONFIG(5)

『BEA Tuxedo アプリケーションの設定』

『BEA Tuxedo アプリケーション実行時の管理』

『ATMI アプリケーションでの BEA Tuxedo TOP END Domain Gateway の使用』

DMCONFIG(5)

名前 DMCONFIG― テキスト形式の Domains コンフィギュレーション・ファイル

機能説明 Domains コンフィギュレーションは、BEA Tuxedo Domains コンポーネントを使用して通信およびサービス共有を行うことができる 2 つ以上のドメイン (ビジネス・アプリケーション) の集まりです。複数のドメインを接続する方法や複数のドメイン間で相互にアクセスできるサービスについては、Domains コンフィギュレーションに参加する各 BEA Tuxedo ドメインの Domains コンフィギュレーション・ファイルで定義されます。テキスト形式の Domains コンフィギュレーション・ファイルは DMCONFIG と呼ばれますが、ファイルの内容がこのリファレンス・ページで説明する形式に従っている限り、任意の名前を付けることができます。

DMCONFIG ファイルは、dmloadcf(1) ユーティリティによって構文解析され、バイナリ形式のファイル BDMCONFIG にロードされます。DMCONFIG ファイルと同様、BDMCONFIG ファイルがどのような名前であっても、実際の名前は BDMCONFIG 環境変数で指定されたデバイス・ファイル名またはシステム・ファイル名になります。BDMCONFIG ファイルは、Domains コンフィギュレーションに参加する Tuxedo ドメインごとに 1 つ必要です。

DMCONFIG ファイルと BDMCONFIG ファイルの関係は、BEA Tuxedo ドメインの定義に使用される UBBCONFIG ファイルと TUXCONFIG ファイルの関係に似ています。UBBCONFIG ファイルと TUXCONFIG ファイルについては、UBBCONFIG(5) を参照してください。

DMCONFIG ファイルの詳細 (例を含む) については、114 ページの「DMCONFIG(5) に関する追加情報」を参照してください。ATMI と CORBA の両環境向けの BEA Tuxedo Domains コンポーネントについては、『BEA Tuxedo Domains コンポーネント』を参照してください。

定義 BEA Tuxedo ドメインは、単一の TUXCONFIG ファイルに記述された環境として定義されます。BEA Tuxedo 用語では、ドメインとアプリケーション (ビジネス・アプリケーション) は同義です。

Domains コンフィギュレーションに含まれる各 BEA Tuxedo ドメインでは、1 つの Domains 管理サーバ (DMADM) プロセスが実行されます。DMADM は、特定の BEA Tuxedo ドメインで実行されるすべてのドメイン・ゲートウェイ・グループ用の管理サーバです。

ドメイン・ゲートウェイ・グループは、BEA Tuxedo システムのゲートウェイ管理サーバ (GWADM) プロセスと BEA Tuxedo システムのドメイン・ゲートウェイ・プロセスで構成されます。

BEA Tuxedo システムのドメイン・ゲートウェイ・プロセスは、特定のタイプのトランザクション処理 (TP) ドメインとの通信サービスを提供します。たとえば、GWTDOMAIN プロセスを使用すると、BEA Tuxedo アプリケーションは他の BEA Tuxedo アプリケーションと通信できます。ドメイン・ゲートウェイは、別のドメインへの要求を中継し、応答を受信します。

ローカル・ドメイン・アクセス・ポイントは、他のドメイン (リモート・ドメイン) が使用できる BEA Tuxedo ドメインの一連のサービスを表すユーザ指定の論理名です。ローカル・ドメイン・アクセス・ポイントはドメイン・ゲートウェイ・グループにマップされるため、どちらも同義語として使用されます。

リモート・ドメイン・アクセス・ポイントは、ローカル・ドメインが使用できるリモート・ドメインの一連のサービスを表すユーザ指定の論理名です。リモート・ドメインは、別の BEA Tuxedo アプリケーションまたは別の TP システムで動作するアプリケーションです。

リモート・サービスは、ローカル・ドメインがリモート・ドメイン・アクセス・ポイントとローカル・ドメイン・アクセス・ポイントを介して使用できるリモート・ドメインのサービスです。

ローカル・サービスは、リモート・ドメインがローカル・ドメイン・アクセス・ポイントを介して使用できるローカル・ドメインのサービスです。

コンフィギュレーション・ファイルの目的

DMCONFIG ファイルは、次の目的で使用します。

- リモート・ドメインのアプリケーション・クライアントがローカル・ドメインのサービスにアクセスするためのローカル・ドメイン・アクセス・ポイントを定義する。
- 各ローカル・ドメイン・アクセス・ポイントを介して使用できるローカル・サービスを定義する。

- ローカル・ドメインのアプリケーション・クライアントがリモート・ドメインのサービスにアクセスするためのリモート・ドメイン・アクセス・ポイントを定義する。
- 各リモート・ドメイン・アクセス・ポイントを介して使用できるリモート・サービスを定義する。
- ローカル・ドメイン・アクセス・ポイントとリモート・ドメイン・アクセス・ポイントを特定のドメイン・ゲートウェイ・グループとネットワーク・アドレスにマップする。

コンフィギュレーション・ファイルの形式

DMCONFIG ファイルは、次のセクションで構成されます。

- DM_LOCAL (DM_LOCAL_DOMAINS ともいう)
- DM_REMOTE (DM_REMOTE_DOMAINS ともいう)
- DM_EXPORT (DM_LOCAL_SERVICES ともいう)
- DM_IMPORT (DM_REMOTE_SERVICES ともいう)
- DM_RESOURCES
- DM_ROUTING
- DM_ACCESS_CONTROL
- DM_TDOMAIN (TDOMAIN タイプのドメイン・ゲートウェイ用のセクション)
- DM_*dom* (*dom* は他のドメイン・ゲートウェイ・タイプのセクション (TOPEND、SNACRM、SNASTACKS、SNALINKS、OSITP、OSITPX) のいずれか)

DMCONFIG ファイル内のアスタリスク (*) で始まる行は、指定セクションの開始を表します。アスタリスク (*) の直後にはセクション名が表示されます。アスタリスクは、セクション名を指定するときに必要です。DM_LOCAL セクションは、DM_REMOTE セクションの前になければなりません。

このリファレンス・ページでは、GWTDOMAIN ゲートウェイ・プロセスによってインプリメントされる TDOMAIN (TDomain ゲートウェイ) をコンフィギュレーションする方法について説明します。TOPEND ドメイン・ゲートウェイの詳細については、DMCONFIG for GWTOPEND(5) と『ATMI アプリケーション

での BEA Tuxedo TOP END Domain Gateway の使用』を参照してください。SNAX、OSITP、または OSITPX ドメイン・ゲートウェイのコンフィギュレーションについては、BEA eLink Documentation を参照してください。

パラメータは通常、*KEYWORD* = *value* という形式で指定します。等号記号 (=) の前後には空白またはタブ文字を使用できます。この形式により、*KEYWORD* が *value* に設定されます。有効なキーワードについては、以下の各セクションで説明します。

予約語の DEFAULT で始まる行にはパラメータ仕様が含まれており、セクション内の以降の該当するすべての行に対して適用されます。デフォルト仕様はすべてのセクションで使用でき、同じセクション内で複数回使用できます。これらの行のフォーマットは次のとおりです。

DEFAULT: [*KEYWORD1* = *value1* [*KEYWORD2* = *value2* [...]]]

この行で設定した値は、別の DEFAULT 行によってリセットされるか、セクションが終わるまで有効です。これらの値は、DEFAULT でない行のオプション・パラメータによって無効になる場合もあります。DEFAULT でない行におけるパラメータ設定は、その行でのみ有効です。以降の行ではデフォルト設定に戻ります。DEFAULT が行頭に表示されると、それ以前に設定されたすべてのデフォルト値はクリアされ、システムのデフォルト値に戻ります。

値が *numeric* の場合は、C の標準表記法を使用して基数を示します。つまり、基数 16 (16 進) の接頭辞は 0x、基数 8 (8 進) の接頭辞は 0、基数 10 (10 進) には接頭辞が付きません。数値パラメータで指定できる範囲については、各パラメータの項で説明します。

値が *identifier* (TYPE パラメータの TDOMAIN のように BEA Tuxedo Domains コンポーネントにとって既知の文字列値) の場合、一般的に標準 C 規則が使用されます。標準 C の *identifier* の先頭には英字またはアンダースコア (_) を使用し、以降の識別子には英数字またはアンダースコアを使用する必要があります。identifier の長さは最大 30 バイトです (最後のヌルを除く)。

識別子を二重引用符で囲む必要はありません。整数でも識別子でもない値は、二重引用符で囲む必要があります。

入力フィールドは、1 つ以上の空白 (またはタブ) 文字で区切ります。

"#" はコメントを示します。復帰改行文字でコメントを終了します。

空白行とコメントは無視されます。

コメントは任意の行の最後に自由に入力できます。

行は、復帰改行の後に最低 1 つのタブを置いて継続できます。コメントを継続することはできません。

Domains 関連の新しい用語

BEA Tuxedo のリリース 7.1 以降では、ドメイン関連の用語の一部が変更されました。Domains 用の MIB では、新しいクラスおよび属性の用語を使用して、ローカル・ドメインとリモート・ドメイン間の対話を説明しています。新しい用語は、DMCONFIG(5) リファレンス・ページ、セクション名、パラメータ名、エラー・メッセージ、および DM_MIB(5) リファレンス・ページ、クラス、エラー・メッセージに適用されます。

下位互換性のため、BEA Tuxedo 7.1 より前に使用されていた DMCONFIG 用語と Domains 用の MIB の新しい用語との間でエイリアスが提供されています。BEA Tuxedo リリース 7.1 以降の DMCONFIG では、両方のバージョンの用語を使用できます。次の表に、DMCONFIG ファイルの旧用語と新用語の対応を示します。

旧用語		新用語	
セクション名	**パラメータ名**	**セクション名**	**パラメータ名**
DM_LOCAL_DOMAINS		DM_LOCAL	
DM_REMOTE_DOMAINS		DM_REMOTE	
	DOMAINID		ACCESSPOINTID
	MAXRDOM		MAXACCESSPOINT
	MAXRDTRAN		MAXRAPTRAN
DM_LOCAL_SERVICES		DM_EXPORT	
DM_REMOTE_SERVICES		DM_IMPORT	
	LDOM		LACCESSPOINT
	RDOM		RACCESSPOINT

BEA Tuxedo リリース 7.1 以降では、dmunloadcf コマンドは、新しいドメイン用語を使用する DMCONFIG ファイルをデフォルトで生成します。旧ドメイン用語を使用する DMCONFIG ファイルを生成するには、-c オプションを使用します。次に例を示します。

プロンプト > dmunloadcf -c > dmconfig_prev

DM_LOCAL セクション

このセクション (DM_LOCAL_DOMAINS セクションともいう) では、1 つまたは複数のローカル・ドメイン・アクセス・ポイント識別子と、それらに関連付けるゲートウェイ・グループを定義します。このセクションには、UBBCONFIG ファイルで指定されたアクティブなゲートウェイ・グループごとにローカル・ドメイン・アクセス・ポイントのエントリが必要です。各エントリでは、グループで実行されるドメイン・ゲートウェイ・プロセスに必要なパラメータを指定します。

DM_LOCAL セクションのエントリの形式は次のとおりです。

LocalAccessPoint *required_parameters* [*optional_parameters*]

LocalAccessPoint は、UBBCONFIG ファイルで定義された特定のゲートウェイ・グループを表すローカル・ドメイン・アクセス・ポイント識別子 (論理名) です。LocalAccessPoint は、Domains コンフィギュレーションに含まれるローカルおよびリモート・ドメイン間で一意でなければなりません。DM_EXPORT セクションで説明するとおり、ローカル・ドメイン・アクセス・ポイントはローカル・サービスを特定のゲートウェイ・グループに関連付けるために使用します。ローカル・ドメイン・アクセス・ポイントを介して使用できるローカル・サービスは、1 つまたは複数のリモート・ドメイン内のクライアントで使用できます。

DM_LOCAL セクションの必須パラメータ

GWGRP = *identifier*

このローカル・ドメイン・アクセス・ポイントを表すドメイン・ゲートウェイ・グループの名前 (TUXCONFIG ファイルの GROUPS セクションで指定された名前) を指定します。ローカル・ドメイン・アクセス・ポイントとゲートウェイ・グループは、1 対 1 の関係です。

TYPE = *identifier*

このローカル・ドメイン・アクセス・ポイントに関連付けるドメイン・ゲートウェイのタイプを指定します。TYPE は、TDOMAIN、TOPEND、SNAX、OSITP、または OSITPX に設定できます。

TDOMAIN は、このローカル・ドメイン・アクセス・ポイントが GWTDOMAIN ゲートウェイ・インスタンスに関連付けられ、これによって別の BEA Tuxedo アプリケーションと通信できることを示します。

TOPEND は、このローカル・ドメイン・アクセス・ポイントが GWTOPEND ドメイン・ゲートウェイ・インスタンスに関連付けられ、これによって BEA TOP END システムと通信できることを示します。

SNAX は、このローカル・ドメイン・アクセス・ポイントが GWSNAX ゲートウェイ・インスタンスに関連付けられ、これによって別の TP ドメインに SNA プロトコルを介して通信できることを示します。

OSITP または OSITPX は、このローカル・ドメイン・アクセス・ポイントが GWOSITP ゲートウェイ・インスタンスに関連付けられ、これによって別の TP ドメインに OSI TP プロトコルを介して通信できることを示します。OSITP は OSI TP 1.3 プロトコルを使用することを示し、OSITPX は OSI TP 4.0 以降のプロトコルを使用することを示します。OSITPX は、BEA Tuxedo 8.0 以降のソフトウェアでのみサポートされます。

ドメイン・タイプは、DMTYPE ファイルで定義する必要があります。このファイルの場所は、Windows の場合は %TUXDIR%\udataobj\DMTYPE、UNIX の場合は $TUXDIR/udataobj/DMTYPE です。

ACCESSPOINTID (DOMAINID ともいう) = *string*[1..30]

リモート・ドメインへの接続を設定するときのセキュリティのため、このローカル・ドメイン・アクセス・ポイントに関連付けられているドメイン・ゲートウェイ・グループを識別するために使用します。ACCESSPOINTID は、すべてのローカルおよびリモート・ドメイン・アクセス・ポイント間で一意でなければなりません。

string の値は、一連の文字 ("BA.CENTRAL01" など) か、または 0x で始まる 16 進数 ("0x0002FF98C0000B9D6" など) です。ACCESSPOINTID は、30 バイト以下で指定する必要があります。文字列を指定する場合は、30 文字以内で指定する必要があります (最後のヌルを含む)。

DM_LOCAL セクションのオプション・パラメータ

以下に示す DM_LOCAL セクションのオプション・パラメータでは、ドメイン・ゲートウェイの操作で使用するリソースと制限を指定します。

AUDITLOG = *string*[1..256] (BEA Tuxedo 8.0 以前では最大 78 バイト)

このローカル・ドメイン・アクセス・ポイントに対する監査ログ・ファイルの名前を指定します。監査ログ機能は dmadmin(1) コマンドによって起動し、このローカル・ドメイン・アクセス・ポイントで行われるすべての動作を記録します。監査ログ機能がオンになっており、このパラメータが指定されていないと、環境変数 $APPDIR によって指定されたディレクトリまたは TUXCONFIG ファイルの MACHINES セクションの APPDIR パラメータで指定されるディレクトリに、DM*mmddyy*.LOG (*mm*= 月、*dd*= 日、*yy*= 年) というファイルが作成されます。

BLOCKTIME = *numeric*

このローカル・ドメイン・アクセス・ポイントに対するブロッキング・コールの最大待ち時間を指定します。この値は、TUXCONFIG ファイルの RESOURCES セクションの SCANUNIT パラメータの乗数です。SCANUNIT * BLOCKTIME の値は、SCANUNIT 以上 32,768 秒未満でなければなりません。このパラメータを指定しないと、TUXCONFIG ファイルの RESOURCES セクションに指定された BLOCKTIME パラメータの値がデフォルトとして使用されます。ブロッキング・タイムアウト状態は、関連する要求が失敗したことを示します。

ドメイン間トランザクションでは、トランザクション期間が BLOCKTIME を超過すると ブロッキング・タイムアウト状態が生成されます。つまり、ドメイン間トランザクションでは、BLOCKTIME 値が TUXCONFIG ファイルの SERVICES セクションで指定された TRANTIME タイムアウト値未満の場合、またはトランザクションを開始するための tpbegin() 呼び出しで渡されたタイムアウト値未満の場合、トランザクションのタイムアウトは BLOCKTIME 値まで減らされます。一方、ドメイン内トランザクション (単一の BEA Tuxedo ドメイン内で処理されるトランザクション) の場合は、TUXCONFIG ファイルの RESOURCES セクションで指定された BLOCKTIME 値は、ドメイン内トランザクションのタイムアウトに何の影響も与えません。

CONNECTION_POLICY = {ON_DEMAND | ON_STARTUP | INCOMING_ONLY}

このローカル・ドメイン・アクセス・ポイントに関連付けられているドメイン・ゲートウェイがリモート・ドメインへの接続を確立するときの条件を指定します。有効な値は、ON_DEMAND、ON_STARTUP、または INCOMING_ONLY です。このパラメータは、TDOMAIN または TOPEND タイプのドメイン・ゲートウェイにのみ適用されます。

接続ポリシーが ON_DEMAND の場合、クライアントがリモート・サービスを要求したとき、または dmadmin(1) connect コマンドが実行されたときにのみ、ドメイン・ゲートウェイはリモート・ドメインへの接続を試行します。CONNECTION_POLICY のデフォルトは ON_DEMAND です。接続ポリシーが ON_DEMAND の場合、再接続は行われません。

接続ポリシーが ON_STARTUP の場合、ドメイン・ゲートウェイはゲートウェイ・サーバの初期化時にリモート・ドメインへの接続を試行します。CONNECTION_POLICY を ON_STARTUP に設定した場合、リモート・ドメインへの接続が確立された場合にのみそのリモート・サービス (ドメイン・ゲートウェイによって宣言されたサービス) が宣言されますつまり、リモート・ドメインとの接続が確立されていないと、リモート・サービスは中断されます。デフォルトでは、失敗した接続が 60 秒おきに再試行されるよう設定されています。再接続の間隔は、RETRY_INTERVAL パラメータで変更できます。MAXRETRY パラメータも参照してください。

接続ポリシーが INCOMING_ONLY の場合、ドメイン・ゲートウェイは起動時にリモート・ドメインへの接続を試みません。このため、リモート・サービスは最初は中断されています。ドメイン・ゲートウェイは、リモート・ドメインからの接続を受信したときに利用可能になります。リモート・サービスは、ドメイン・ゲートウェイが接続を受信したときか、dmadmin(1) connect コマンドで管理接続が確立されたときに宣言されます。接続ポリシーが INCOMING_ONLY の場合、再接続は行われません。

注記　BEA Tuxedo 8.1 以降のソフトウェアが実行されている TDOMAIN ドメイン・ゲートウェイの場合、DM_TDOMAIN セクションにリモート・ドメインごとに CONNECTION_POLICY を指定できます。

MAXRETRY = {*numeric* | MAXLONG}

このローカル・ドメイン・アクセス・ポイントに関連付けられているドメイン・ゲートウェイがリモート・ドメインへの接続を試行する回数を指定します。このパラメータは、TDOMAIN または TOPEND タイプのドメイン・ゲートウェイにのみ適用され、このローカル・ドメイン・アクセス・ポイントの CONNECTION_POLICY パラメータが ON_STARTUP に設定されている場合にのみ有効です。それ以外の接続ポリシーの場合、自動再試行は無効になります。

MAXRETRY の最小値は 0 で、最大値は MAXLONG (2147483647) です。MAXLONG (デフォルト) の場合、再接続処理が無限に繰り返されるか、または接続が確立されるまで繰り返されます。MAXRETRY=0 に設定すると、自動再接続は行われません。

RETRY_INTERVAL = *numeric*

このローカル・ドメイン・アクセス・ポイントに関連付けられているドメイン・ゲートウェイがリモート・ドメインへの接続を自動的に試行する間隔を秒単位で指定します。このパラメータは、TDOMAIN または TOPEND タイプのドメイン・ゲートウェイにのみ適用され、このローカル・ドメイン・アクセス・ポイントの CONNECTION_POLICY パラメータが ON_STARTUP に設定されている場合にのみ有効です。それ以外の接続ポリシーの場合、自動再試行は無効になります。

RETRY_INTERVAL の最小値は 0 、最大値は 2147483647 です。デフォルトは 60 です。MAXRETRY を 0 に設定すると、RETRY_INTERVAL は設定できません。

CONNECTION_PRINCIPAL_NAME = *string*[0..511]

接続プリンシパル名の識別子を指定します。これは、ローカル・ドメイン・アクセス・ポイントに関連付けられているドメイン・ゲートウェイがリモート・ドメインに接続するときに、そのドメイン・ゲートウェイの ID を検証するためのプリンシパル名です。このパラメータは、BEA Tuxedo 7.1 以降のソフトウェアが実行されている TDOMAIN タイプのドメイン・ゲートウェイにのみ適用されます。

CONNECTION_PRINCIPAL_NAME パラメータには最大 511 文字を指定できます (最後のヌル文字を除く)。このパラメータを指定しないと、接続プリンシパル名はデフォルトでこのローカル・ドメイン・アクセス・ポイントの ACCESSPOINTID 文字列になります。

デフォルトの認証プラグインの場合、このローカル・ドメイン・アクセス・ポイントの CONNECTION_PRINCIPAL_NAME パラメータに値を割り当てる場合、その値は、このローカル・ドメイン・アクセス・ポイントの ACCESSPOINTID パラメータの値と同じでなければなりません。これらの値が一致しないと、ローカル・TDomain ゲートウェイ・プロセスが起動せず、次の userlog(3c) メッセージが生成されます。
ERROR: 証明書を取得できません。

DMTLOGDEV = *string*[1..256] (BEA Tuxedo 8.0 以前では最大 78 バイト)

このローカル・ドメイン・アクセス・ポイントの Domains トランザクション・ログ (TLOG) を含む BEA Tuxedo ファイルシステムを指定します。TLOG は、BEA Tuxedo システムの VTOC テーブルとしてデバイスに格納されています。このパラメータを指定しない場合、このローカル・ドメイン・アクセス・ポイントに関連付けられているドメイン・ゲートウェイ・グループは要求をトランザクション・モードで処理できません。同じマシンのローカル・ドメイン・アクセス・ポイント間で同じ BEA Tuxedo ファイルシステムを共有することはできますが、各ローカル・ドメイン・アクセス・ポイントは DMTLOGNAME パラメータで指定された名前のログ (DMTLOGDEV 内のテーブル) を保持する必要があります。

DMTLOGNAME = *string*[1..30]

このローカル・ドメイン・アクセス・ポイント用の TLOG の名前を指定します。複数のローカル・ドメイン・アクセス・ポイント間で同じ BEA Tuxedo ファイルシステム (DMTLOGDEV で指定) を共有する場合は、一意な名前を指定する必要があります。このパラメータを指定しない場合、デフォルトは DMTLOG となります。名前は 30 文字以内で指定する必要があります。

DMTLOGSIZE = *numeric*

このローカル・ドメイン・アクセス・ポイント用の TLOG のサイズをページ数で指定します。この値は、0 より大きく、BEA Tuxedo ファイルシステムで使用可能な容量より小さくする必要があります。このパラメータを指定しないと、デフォルトの 100 ページが設定されます。

MAXRAPTRAN (MAXRDTRAN ともいう) = *numeric*

このローカル・ドメイン・アクセス・ポイントのトランザクションに含めることのできるドメインの最大数を指定します。この値は 0 より大きく、32,768 未満でなければなりません。デフォルトは 16 です。

MAXTRAN = *numeric*

このローカル・ドメイン・アクセス・ポイントで同時に実行できるグローバル・トランザクションの最大数を指定します。この値は、0 以上で、TUXCONFIG ファイルの RESOURCES セクションに定義されている MAXGTT パラメータ以下でなければなりません。MAXTRAN を指定しない場合、デフォルトは MAXGTT です。

MTYPE = *string*[1..15]

ドメインをグループ化して、このローカル・ドメイン・アクセス・ポイントに関連付けられているマシンとリモート・ドメイン・アクセス・ポイントに関連付けられているマシン間のメッセージの符号化と復号化を省略するために使用します。このパラメータは、TDOMAIN または TOPEND タイプのドメイン・ゲートウェイにのみ適用されます。

MTYPE を指定しない場合、デフォルトで符号化または復号化が実行されます。MTYPE フィールドに設定した値が DMCONFIG ファイルの DM_LOCAL セクションと DM_REMOTE セクションで共通している場合、データの符号化と復号化が省略されます。MTYPE には、15 文字までの任意の文字列値を指定できます。この値は比較のためだけに使用します。

SECURITY = {NONE | APP_PW | DM_PW}

このローカル・ドメイン・アクセス・ポイント用に使用するアプリケーション・セキュリティの種類を指定します。TDOMAIN ドメイン・ゲートウェイ用の SECURITY パラメータの有効値は、現時点では NONE、APP_PW、DM_PW の 3 つです。NONE (デフォルト) の場合、セキュリティは使用されません。APP_PW を指定すると、リモート・ドメインからの接続の確立時にアプリケーション・パスワード・セキュリティが使用されます。アプリケーション・パスワードは、TUXCONFIG ファイルで定義しておく必要があります。DM_PW を指定すると、リモート・ドメインからの接続の確立時に、ドメイン・パスワード・セキュリティが使用されます。ドメイン・パスワードは、dmadmin(1) コマンドで定義しておく必要があります。

SECURITY パラメータは、OSITP ドメイン・ゲートウェイには適用されません。OSITPX ゲートウェイの場合、NONE または DM_PW を使用できます。SNAX ゲートウェイの場合、NONE または DM_USER_PW を使用できます。TOPEND ゲートウェイの場合、NONE、CLEAR、SAFE、または PRIVATE を使用できます。

DM_LOCAL セクションの 非 TDomain パラメータ

以下の DM_LOCAL セクション・パラメータは補完的に示したもので、TDOMAIN ドメイン・ゲートウェイには適用されません。

- BLOB_SHM_SIZE = *numeric* — SNAX ドメイン・ゲートウェイに適用可能です。
- MAXACCESSPOINT (MAXRDOM ともいう) = *numeric* — OSITP ドメイン・ゲートウェイに適用可能です。
- MAXDATALEN = *numeric* — OSITP ドメイン・ゲートウェイに適用可能です。

SNAX および OSITP パラメータの詳細については、BEA eLink Documentation を参照してください。

DM_REMOTE セクション

このセクション (DM_REMOTE_DOMAINS セクションともいう) では、1 つまたは複数のリモート・ドメイン・アクセス・ポイント識別子とそれらの特性を定義します。

DM_REMOTE セクションのエントリの形式は次のとおりです。

RemoteAccessPoint required_parameters [*optional_parameters*]

RemoteAccessPoint は、ローカル BEA Tuxedo アプリケーションにとって既知の各リモート・ドメインを識別するために選択するリモート・ドメイン・アクセス・ポイント識別子 (論理名) です。*RemoteAccessPoint* は、Domains コンフィギュレーションに含まれるローカルおよびリモート・ドメイン間で一意でなければなりません。DM_IMPORT セクションで説明するとおり、リモート・ドメイン・アクセス・ポイントはリモート・サービスを特定のリモート・ドメインに関連付けるために使用します。リモート・ドメイン・アクセス・ポイントを介して使用できるリモート・サービスは、リモート・ドメイン・アクセス・ポイントとローカル・ドメイン・アクセス・ポイントを介してローカル・ドメイン内のクライアントで使用できます。

DM_REMOTE セクションの必須パラメータ

TYPE = *identifier*

このリモート・ドメイン・アクセス・ポイントに関連付けられるリモート・ドメインとの通信に必要なローカル・ドメイン・ゲートウェイのタイプを指定します。TYPE は、TDOMAIN、TOPEND、SNAX、OSITP、または OSITPX に設定できます。

TDOMAIN は、GWTDOMAIN プロセスのローカル・インスタンスがリモート BEA Tuxedo アプリケーションと通信することを示します。

TOPEND は、GWTOPEND プロセスのローカル・インスタンスがリモート BEA TOP END システムと通信することを示します。

SNAX は、GWSNAX プロセスのローカル・インスタンスが SNA プロトコルを介してリモート TP ドメインと通信することを示します。

OSITP は、GWOSITP プロセスのローカル・インスタンスが OSI TP 1.3 プロトコルを介してリモート TP ドメインと通信することを示します。

OSITPX は、GWOSITP プロセスのローカル・インスタンスが OSI TP 4.0 以降のプロトコルを介してリモート TP ドメインと通信することを示します。OSITPX は、BEA Tuxedo 8.0 以降のソフトウェアでのみサポートされます。

ACCESSPOINTID (DOMAINID ともいう) = *string*[1..30]
リモート・ドメインへの接続を設定するときのセキュリティのため、このリモート・ドメイン・アクセス・ポイントに関連付けられているリモート・ドメインを識別するために使用します。TDOMAIN ローカル・ドメイン・ゲートウェイの場合、この値は、このリモート・ドメイン・アクセス・ポイント接続から受信した要求のユーザ ID として TDomain ゲートウェイ (GWTDOMAIN プロセスのローカル・インスタンス) によって使用される場合があります。ACCESSPOINTID は、ローカルおよびリモート・ドメイン・アクセス・ポイント間で一意でなければなりません。

ACCESSPOINTID は、30 バイト以下で指定する必要があります。文字列を指定する場合は、30 文字以内で指定する必要があります (最後のヌルを含む)。*string* の値は、一連の文字か、または 0x で始まる 16 進数です。

DM_REMOTE セクションのオプション・パラメータ

以下に示す DM_REMOTE セクションのオプション・パラメータでは、ドメイン・ゲートウェイの操作で使用するリソースと制限を指定します。

ACL_POLICY = {LOCAL | GLOBAL}
このリモート・ドメイン・アクセス・ポイント用のアクセス制御リスト (ACL) 方針を指定します。このパラメータは、BEA Tuxedo 7.1 以降のソフトウェアが実行されている TDOMAIN タイプのドメイン・ゲートウェイ、および BEA Tuxedo 8.0 以降のソフトウェアが実行されている OSITPX タイプのドメイン・ゲートウェイにのみに適用されます。

LOCAL を指定すると、ローカル・ドメインは、リモート・ドメインから受け取ったサービス要求のクリデンシャル (ID) を、このリモート・ドメイン・アクセス・ポイントに対する LOCAL_PRINCIPAL_NAME パラメータで指定されたプリンシパル名に置き換えます。GLOBAL を指定すると、ローカル・ドメインは、リモート・サービス要求で受け取ったクリデンシャルを置き換えません。リモート・サービス要求でクリデ

ンシャルを受け取らなかった場合、ローカル・ドメインはそのサービス要求をそのままの状態でローカル・サービスに転送します (通常これは失敗する)。このパラメータを指定しない場合、デフォルトは LOCAL です。

ACL_POLICY パラメータは、ローカル・ドメインがリモート・ドメインから受信したサービス要求のクリデンシャルを LOCAL_PRINCIPAL_NAME パラメータに指定されているプリンシパル名に置き換えるかどうかを制御します。CREDENTIAL_POLICY はこのパラメータに関連するパラメータで、ローカル・ドメインがリモート・ドメインにローカル・サービス要求を送信する前にその要求からクリデンシャルを削除するかどうかを制御します。

LOCAL_PRINCIPAL_NAME = *string*[0..511]

ローカル・プリンシパル名の識別子 (クリデンシャル) を指定します。これは、このリモート・ドメイン・アクセス・ポイントの ACL_POLICY パラメータが LOCAL (デフォルト) に設定されている場合、このリモート・ドメインから受け取ったサービス要求に対してローカル・ドメインが割り当てる ID です。このパラメータは、BEA Tuxedo 7.1 以降のソフトウェアが実行されている TDOMAIN タイプのドメイン・ゲートウェイ、および BEA Tuxedo 8.0 以降が実行されている OSITPX タイプのドメイン・ゲートウェイにのみに適用されます。

LOCAL_PRINCIPAL_NAME パラメータには最大 511 文字を指定できます (最後のヌル文字を除く)。このパラメータを指定しないと、ローカル・プリンシパル名はデフォルトでこのリモート・ドメイン・アクセス・ポイントの ACCESSPOINTID 文字列になります。

CONNECTION_PRINCIPAL_NAME = *string*[0..511]

接続プリンシパル名の識別子を指定します。これは、ローカル・ドメインに接続するこのリモート・ドメイン・アクセス・ポイントの ID を検証するためのプリンシパル名です。このパラメータは、BEA Tuxedo 7.1 以降のソフトウェアが実行されている TDOMAIN タイプのドメイン・ゲートウェイにのみ適用されます。

CONNECTION_PRINCIPAL_NAME パラメータには最大 511 文字を指定できます (最後のヌル文字を除く)。このパラメータを指定しないと、接続プリンシパル名はデフォルトでこのリモート・ドメイン・アクセス・ポイントの ACCESSPOINTID 文字列になります。

デフォルトの認証プラグインの場合、このリモート・ドメイン・アクセス・ポイントの CONNECTION_PRINCIPAL_NAME パラメータに値を割り当てる場合、その値は、このリモート・ドメイン・アクセス・ポイントの ACCESSPOINTID パラメータの値と同じでなければなりません。これらの値が一致しないと、ローカル TDomain ゲートウェイとリモート TDomain ゲートウェイの接続は失敗し、次の userlog(3c) メッセージが生成されます。ERROR: Unable to initialize administration key for domain *domain_name*.

CREDENTIAL_POLICY = {LOCAL | GLOBAL}

このリモート・ドメイン・アクセス・ポイント用のクリデンシャル方針を指定します。このパラメータは、BEA Tuxedo 8.0 以降のソフトウェアが実行されている TDOMAIN タイプのドメイン・ゲートウェイにのみ適用されます。

LOCAL を指定すると、ローカル・ドメインは、このリモート・ドメイン・アクセス・ポイントへのローカル・サービス要求からクリデンシャル (ID) を削除します。GLOBAL を指定すると、ローカル・ドメインは、このリモート・ドメイン・アクセス・ポイントへのローカル・サービス要求からクリデンシャルを削除しません。このパラメータを指定しない場合、デフォルトは LOCAL です。

CREDENTIAL_POLICY パラメータは、ローカル・ドメインがリモート・ドメインにローカル・サービス要求を送信する前にその要求からクリデンシャルを削除するかどうかを制御します。ACL_POLICY はこのパラメータに関連するパラメータで、ローカル・ドメインがリモート・ドメインから受信したサービス要求のクリデンシャルを LOCAL_PRINCIPAL_NAME パラメータに指定されているプリンシパル名に置き換えるかどうかを制御します。

MTYPE = *string*[1..15]

ドメインをグループ化して、このリモート・ドメイン・アクセス・ポイントに関連付けられているマシンとローカル・ドメイン・アクセス・ポイントに関連付けられているマシン間のメッセージの符号化と復号化を省略するために使用します。このパラメータは、TDOMAIN タイプのドメイン・ゲートウェイにのみ適用されます。

MTYPE を指定しない場合、デフォルトで符号化または復号化が実行されます。MTYPE フィールドに設定した値が DMCONFIG ファイルの DM_LOCAL セクションと DM_REMOTE セクションで共通している場合、

データの符号化と復号化が省略されます。MTYPE には、15 文字までの任意の文字列値を指定できます。この値は比較のためだけに使用します。

PRIORITY_TYPE = {LOCAL_RELATIVE | LOCAL_ABSOLUTE | GLOBAL}

INPRIORITY = *numeric*

PRIORITY_TYPE パラメータと INPRIORITY パラメータでは、このリモート・ドメイン・アクセス・ポイントのメッセージの優先順位に関する処理を指定します。これらのパラメータは、BEA Tuxedo 8.0 以降のソフトウェアでサポートされます。

PRIORITY_TYPE パラメータの場合、LOCAL_RELATIVE と LOCAL_ABSOLUTE はすべてのリモート・ドメイン・タイプに対して有効ですが、GLOBAL は TDOMAIN のリモート・ドメイン・タイプに対してのみ有効です。PRIORITY_TYPE パラメータを設定しない場合、デフォルトは LOCAL_RELATIVE です。

PRIORITY_TYPE=LOCAL_RELATIVE は、tpsprio 呼び出しなどによるリモート・ドメイン・アクセス・ポイントからの要求の優先順位がローカル・ドメインによって使用されないことを意味します。代わりに、リモート・ドメイン・アクセス・ポイントから受信する要求の優先順位は INPRIORITY の値を基準に設定されます。この値は -99 (最低の優先順位) ～ +99 (最高の優先順位) です。デフォルトは 0 です。INPRIORITY の設定によって、サービスのデフォルトの優先順位は、設定値の符号に応じて最大 100、最小 1 まで増減します。最高の優先順位は 100 です。リモート・ドメイン・アクセス・ポイントへの要求の場合、要求に関連付けられている優先順位も一緒にリモート・ドメイン・アクセス・ポイントに送信されます。

PRIORITY_TYPE=LOCAL_ABSOLUTE は、このリモート・ドメイン・アクセス・ポイントからの要求の優先順位がローカル・ドメインによって使用されないことを意味します。代わりに、リモート・ドメイン・アクセス・ポイントから受信する要求の優先順位は INPRIORITY の値を基準に設定されます。この値は 1 (最低の優先順位) ～ 100 (最高の優先順位) です。デフォルトは 50 です。INPRIORITY の設定によって、サービスのデフォルトの優先順位は、設定値の符号に応じて最大 100、最小 1 まで増減します。最高の優先順位は 100 です。リモート・ドメイン・アクセス・ポイントへの要求の場合、要求に関連付けられてい

る優先順位も一緒にリモート・ドメイン・アクセス・ポイントに送信されます。

PRIORITY_TYPE=LOCAL_GLOBAL は、このリモート・ドメイン・アクセス・ポイントからの要求の優先順位がローカル・ドメインによって調整されることを意味します。リモート・ドメイン・アクセス・ポイントから受信する要求の優先順位は INPRIORITY の値を基準に調整されます。この値は -99 (最低の優先順位) 〜 +99 (最高の優先順位) です。デフォルトは 0 です。INPRIORITY を設定した場合、受信した要求に関連付けられている優先順位は INPRIORITY の値に加算され、その要求の優先順位の絶対値が設定されます。INPRIORITY を設定しない場合、受信する要求の優先順位がそのままローカル・ドメインによって使用されます。リモート・ドメイン・アクセス・ポイントへの要求の場合、要求に関連付けられている優先順位も一緒にリモート・ドメイン・アクセス・ポイントに送信されます。

DM_REMOTE セクションの 非 TDomain パラメータ

以下の DM_REMOTE セクション・パラメータは補完的に示したもので、TDOMAIN ドメイン・ゲートウェイには適用されません。

CODEPAGE = *string* — SNAX および OSITPX ドメイン・ゲートウェイに適用可能です。

SNAX および OSITPX パラメータの詳細については、BEA eLink Documentation を参照してください。

DM_EXPORT セクション

このセクション (DM_LOCAL_SERVICES セクションともいう) では、各ローカル・ドメイン・アクセス・ポイントによってエクスポートされるサービスに関する情報を指定します。このセクションを指定しない場合、DM_LOCAL セクションで定義したすべてのローカル・ドメイン・アクセス・ポイントは、ローカル BEA Tuxedo アプリケーションによって宣言されるすべてのサービスに対するリモート要求を受け付けます。このセクションを定義することにより、リモート・ドメインから要求できるローカル・サービスのセットが制限されます。

ローカル・サービスは、1 つまたは複数のリモート・ドメインがローカル・ドメイン・アクセス・ポイントを介して使用できるサービスです。

DM_EXPORT セクションのエントリの形式は次のとおりです。

service [*optional_parameters*]

service は、特定のローカル・サービスの識別子の名前 (15 文字以内) です。この名前は、ローカルの BEA Tuxedo アプリケーション内で実行される 1 つまたは複数のサーバによって宣言された名前です。

1 つまたは複数のリモート・ドメインで使用できるローカル・サービスは、そのプロパティの多くを、TUXCONFIG ファイルの SERVICES セクションか、またはそれらのデフォルトから継承します。継承される特性として、LOAD、PRIO、AUTOTRAN、ROUTING、BUFTYPE、TRANTIME があります。

DM_EXPORT セクションのオプション・パラメータ

LACCESSPOINT (LDOM ともいう) = *identifier*
: このサービスをエクスポートするローカル・ドメイン・アクセス・ポイントの名前を指定します。このパラメータを指定しない場合、DM_LOCAL セクションで定義したすべてのローカル・ドメイン・アクセス・ポイントは、このローカル・サービスに対するリモート要求を受け付けます。

ACL = *identifier*
: アクセス制御リスト (ACL) の名前を指定します。ローカル・ドメイン・アクセス・ポイントは、このリストを使用してリモート・ドメイ

ンからのこのサービスへの要求を制限します。ACL の名前は、DM_ACCESS_CONTROL セクションで定義します。

CONV = {Y | N}
このローカル・サービスが会話型サービスであるか (Y) 否か (N) を指定します。デフォルトは N です。

RNAME = *string*[1..30]
リモート・ドメインに対するこのローカル・サービスの名前の代わりとなる識別子 (エイリアス) を指定します。リモート・ドメインは、この名前を使用してこのサービスを要求します。このパラメータを指定しないと、リモート・ドメインはローカル・サービスの実際の名前 (*service* 識別子) を使用してサービスを要求します。

DM_EXPORT セクションの 非 TDomain パラメータ

以下の DM_EXPORT セクション・パラメータは補完的に示したもので、TDOMAIN ドメイン・ゲートウェイには適用されません。

- TYPE = {SERVICE | QSPACE | QNAME} — TOPEND ドメイン・ゲートウェイに適用可能です。
- TE_PRODUCT = *string* — TOPEND ドメイン・ゲートウェイに適用可能です。
- TE_FUNCTION = *string* — TOPEND ドメイン・ゲートウェイに適用可能です。
- TE_TARGET = *string* — TOPEND ドメイン・ゲートウェイに適用可能です。
- TE_QUALIFIER = *numeric* — TOPEND ドメイン・ゲートウェイに適用可能です。
- TE_RTQGROUP = *string* — TOPEND ドメイン・ゲートウェイに適用可能です。
- TE_RTQNAME = *string* — TOPEND ドメイン・ゲートウェイに適用可能です。
- INBUFTYPE = *string* — TOPEND、SNAX、OSITP、および OSITPX ドメイン・ゲートウェイに適用可能です。

- OUTBUFTYPE = *string* — TOPEND、SNAX、OSITP、および OSITPX ドメイン・ゲートウェイに適用可能です。
- COUPLING = {TIGHT | LOOSE} — OSITPX ドメイン・ゲートウェイに適用可能です。
- INRECTYPE = *string* — OSITPX ドメイン・ゲートウェイに適用可能です。
- OUTRECTYPE = *string* — OSITPX ドメイン・ゲートウェイに適用可能です。

TOPEND パラメータの詳細については、DMCONFIG for GWTOPEND(5) を参照してください。SNAX、OSITP、および OSITPX パラメータの詳細については、BEA eLink Documentation を参照してください。

DM_IMPORT セクション

このセクション (DM_REMOTE_SERVICES セクションともいう) では、DM_REMOTE セクションで定義されたリモート・ドメイン・アクセス・ポイントを介してローカル・ドメインにインポートおよび提供されるサービスに関する情報を指定します。DM_IMPORT セクションが存在しない場合、または存在しても空の場合、リモート・サービスはローカル・ドメインで使用できません。

リモート・サービスは、ローカル・ドメインがリモート・ドメイン・アクセス・ポイントとローカル・ドメイン・アクセス・ポイントを介して使用できるサービスです。

DM_IMPORT セクションのエントリの形式は次のとおりです。

service [*optional_parameters*]

service は、特定のリモート・サービスに対してローカル BEA Tuxedo によって宣言される識別子の名前 (15 文字以内) です。リモート・サービスは、1 つまたは複数のリモート・ドメインからインポートされます。

ローカル・ドメインで使用できるリモート BEA Tuxedo サービスは、そのプロパティの多くを、TUXCONFIG ファイルの SERVICES セクションか、またはそれらのデフォルトから継承します。継承される特性として、LOAD、PRIO、AUTOTRAN、ROUTING、BUFTYPE、TRANTIME があります。

DM_IMPORT セクションのオプション・パラメータ

RACCESSPOINT (RDOM ともいう) =
identifier1[,*identifier2*][,*identifier3*]

このサービスをインポートするためのリモート・ドメイン・アクセス・ポイントの名前を指定します。このサービスのリモート・ドメイン・アクセス・ポイントを指定し、かつ、このサービスのローカル・ドメイン・アクセス・ポイントを (LACCESSPOINT パラメータで) 指定した場合、指定したローカル・ドメイン・アクセス・ポイントだけが、指定したリモート・ドメイン・アクセス・ポイントを介してこのリモート・サービスにローカル要求を送信できます。

このサービスのリモート・ドメイン・アクセス・ポイントを指定し、ローカル・ドメイン・アクセス・ポイントを指定しなかった場合、DM_LOCAL セクションで定義され、リモート・ドメイン・アクセス・

ポイントと同じゲートウェイ・タイプ (TDOMAIN など) を持つローカル・ドメイン・アクセス・ポイントが、指定したリモート・ドメイン・アクセス・ポイントを介してこのリモート・サービスにローカル要求を送信できます。

このサービスのリモート・ドメイン・アクセス・ポイントとローカル・ドメイン・アクセス・ポイントをいずれも指定しなかった場合、DM_LOCAL セクションで定義された任意のローカル・ドメイン・アクセス・ポイントが、DM_REMOTE セクションで定義された任意のリモート・ドメイン・アクセス・ポイントを介してこのリモート・サービスにローカル要求を送信できます。

identifier2 および *identifier3* 引数を指定して代替リモート・ドメイン・アクセス・ポイントをコンフィギュレーションする場合、DM_LOCAL セクションの CONNECTION_POLICY パラメータの値として ON_STARTUP を指定する必要があります。BEA Tuxedo 8.1 以降のアプリケーションの場合、DM_TDOMAIN セクションでも CONNECTION_POLICY を指定できます。*identifier2* を設定した場合、それはフェイルオーバー用に使用されます。*identifier1* に関連付けられているリモート・ドメインが使用できなくなった場合、*identifier2* に関連付けられているリモート・ドメインが使用されます。同様に、*identifier3* を設定した場合、それはフェイルオーバー用に使用されます。*identifier1* と *identifier2* に関連付けられているリモート・ドメインが使用できなくなった場合、*identifier3* に関連付けられているリモート・ドメインが使用されます。

LACCESSPOINT (LDOM ともいう) = *identifier*

このリモート・サービスに要求を送信できるローカル・ドメイン・アクセス・ポイントの名前を指定します。このローカル・ドメイン・アクセス・ポイントに関連付けられたゲートウェイ・グループは、リモート・サービスの名前 (*service* 識別子) を BEA Tuxedo システムの掲示板で宣言します。

CONV = {Y | N}

このリモート・サービスが会話型サービスであるか (Y) 否か (N) を指定します。デフォルトは N です。

LOAD = *numeric*

このリモート・サービスのサービス負荷を指定します。この値は、1 以上 32767 以下でなければなりません。デフォルトは 50 です。ロード・バランシングのためにインターフェイス負荷が使用されます。つまり、キュー登録の負荷が高いキューほど、新しい要求の処理に使用されません。

RNAME = *string*[1..30]

ローカル・ドメインに対するこのリモート・サービスの名前の代わりとなる識別子 (エイリアス) を指定します。ローカル・ドメインは、この名前を使用してこのサービスを要求します。このパラメータを指定しないと、ローカル・ドメインはこのリモート・サービスの実際の名前 (*service* 識別子) を使用してこのサービスを要求します。

ROUTING = *identifier*

このリモート・サービスのデータ依存型ルーティングを行うために使用するルーティング基準テーブルの名前を指定します。複数のリモート・ドメイン・アクセス・ポイントが同じサービスを提供するとき、このパラメータを指定してあれば、ローカル・ドメイン・アクセス・ポイントはデータ依存型ルーティングを実行できます。このパラメータを指定しないと、このサービスに対してデータ依存型ルーティングは使用されません。

identifier は、DM_ROUTING セクションで定義された *ROUTING_CRITERIA_NAME* です。*identifier* の値は 15 文字以下でなければなりません。同じサービス名の複数のエントリが異なるリモート・ドメイン・アクセス・ポイントに含まれている場合 (RACCESSPOINT パラメータで指定)、ROUTING パラメータの値はこれらのエントリすべてに対して同じにする必要があります。

DM_IMPORT セクションの 非 TDomain パラメータ

以下の DM_IMPORT セクション・パラメータは補完的に示したもので、TDOMAIN ドメイン・ゲートウェイには適用されません。

- TE_PRODUCT = *string* — TOPEND ドメイン・ゲートウェイに適用可能です。
- TE_FUNCTION = *string* — TOPEND ドメイン・ゲートウェイに適用可能です。

- TE_TARGET = *string* — TOPEND ドメイン・ゲートウェイに適用可能です。
- TE_QUALIFIER = *numeric* — TOPEND ドメイン・ゲートウェイに適用可能です。
- TE_RTQGROUP = *string* — TOPEND ドメイン・ゲートウェイに適用可能です。
- TE_RTQNAME = *string* — TOPEND ドメイン・ゲートウェイに適用可能です。
- INBUFTYPE = *string* — TOPEND、SNAX、OSITP、および OSITPX ドメイン・ゲートウェイに適用可能です。
- OUTBUFTYPE = *string* — TOPEND、SNAX、OSITP、および OSITPX ドメイン・ゲートウェイに適用可能です。
- AUTOPREPARE = {Y | N} — OSITPX ドメイン・ゲートウェイに適用可能です。
- INRECTYPE = *string* — OSITPX ドメイン・ゲートウェイに適用可能です。
- OUTRECTYPE = *string* — OSITPX ドメイン・ゲートウェイに適用可能です。
- TPSUT_TYPE = {INTEGER | PRINTABLESTRING} — OSITPX ドメイン・ゲートウェイに適用可能です。
- REM_TPSUT = *string* — OSITPX ドメイン・ゲートウェイに適用可能です。

TOPEND パラメータの詳細については、DMCONFIG for GWTOPEND(5) を参照してください。SNAX、OSITP、および OSITPX パラメータの詳細については、BEA eLink Documentation を参照してください。

DM_RESOURCES

このオプション・セクションでは、グローバル Domains コンフィギュレーション情報、特にユーザ指定のコンフィギュレーション・バージョン文字列を定義します。このフィールドはソフトウェアによってチェックされません。

DM_RESOURCES セクションのパラメータは次の 1 つだけです。

VERSION = *string*

string は、ユーザが現在の DMCONFIG コンフィギュレーション・ファイルのバージョン番号を入力するためのフィールドです。

DM_ROUTING セクション

このセクションでは、型付きバッファである FML、FML32、VIEW、VIEW32、X_C_TYPE、X_COMMON、または XML を使用したローカル・サービス要求のデータ依存型ルーティングに関する情報を指定します。

DM_ROUTING セクションのエントリの形式は次のとおりです。

ROUTING_CRITERIA_NAME required_parameters

ROUTING_CRITERIA_NAME は、DM_IMPORT セクションの特定のサービス・エントリの ROUTING パラメータに割り当てられる *identifier* の名前です。*ROUTING_CRITERIA_NAME* は 15 文字以下でなければなりません。

DM_ROUTING セクションの必須パラメータ

FIELD = *identifier*

ルーティング・フィールドの名前を指定します。名前は 30 文字以内でなければなりません。*identifier* の値には次のいずれかを指定できます。FML フィールド・テーブル (FML および FML32 バッファの場合) で識別されたフィールド名、XML の要素あるいは要素属性 (XML バッファの場合)、または FML VIEW テーブル (VIEW、X_C_TYPE、または X_COMMON バッファの場合) で識別されたフィールド名です。FML フィールド・テーブルを検索するには、2 つの環境変数、FLDTBLDIR および FIELDTBLS、または FLDTBLDIR32 および FIELDTBLS32 を使用します。同様に、FML VIEW テーブルを検索するには、2 つの環境変数、VIEWDIR および VIEWFILES、または VIEWDIR32 および VIEWFILES32 を使用します。FML または FML32 バッファ内のフィールドがルーティングに使用される場合は、フィールド番号は 8191 以下でなければなりません。

UTF-8 で符号化された XML 要素の内容をルーティングに使用できます。ルーティングに使用する場合、この要素の内容に文字リファレンス、エンティティ・リファレンス、および CDATA セクションを含めることはできません。UTF-8 で符号化された XML 要素の属性は、この属性が属する要素が定義されている場合にルーティングに使用できます。

XML 文書が要素の内容または属性に基づいてルーティングされる場合、FIELD パラメータは次の構文で定義される必要があります。

FIELD = "*root_element*[/*child_element*][/*child_element*][/. . .][/@*attribute_name*]"

FIELD の値には、ルーティングの要素または要素の属性名を指定します。*root_element* の値には、XML ドキュメントまたはデータグラムの要素のタイプ (または名前) あるいは要素の属性名を指定できます。この情報は、ドキュメントまたはデータグラム送信時に、データ依存型ルーティングで要素の内容または属性を識別するために使用されます。要素名と属性名を組み合わせて、最大 30 文字まで指定できます。インデックスはサポートされないので、BEA Tuxedo システムは、データ依存型ルーティングで XML バッファを処理する際に、与えられた要素タイプの最初のオカレンスだけを認識します。

XML は、属性名に使用できる文字セットを厳密に定義しています。属性名は、単一の文字、アンダースコア (_)、またはコロン (:) を含む文字列で、その後に 1 つ以上の名前文字が続きます。要素名と属性名はいずれも、大文字小文字が区別されます。

XML の詳細については、World Wide Web Consortium の Web サイト http://www.w3c.org/XML を参照してください。

FIELDTYPE = *type*

FIELD パラメータに指定されたルーティング・フィールドのタイプを指定します。このパラメータは、XML バッファをルーティングする場合にのみ使用されます。値 *type* は、CHAR、SHORT、LONG、FLOAT、DOUBLE、STRING のいずれかに設定できます。ルーティング・フィールドのデフォルトのタイプは STRING です。

UTF-8 で符号化された XML 要素の内容と属性を FIELDTYPE パラメータで指定されたデータ型に変換できる場合、それらをルーティングに使用できます。

RANGES = "*string*[1..4096]"

ルーティング・フィールドの範囲および関連するリモート・ドメイン・アクセス・ポイント名を指定します。*string* は二重引用符で囲みます。*string* はカンマで区切ったペアのリストで、各ペアはコロン (:) で区切られた範囲とリモート・ドメイン・アクセス・ポイントで構

成されます (
`RANGES = "MIN-1000:b01,1001-3000:b02,*:b03"` など)。

範囲は、単一の値 (符号付き数値または一重引用符で囲んだ文字列)、または *lower* - *upper* の形式で表します。*lower* と *upper* は、いずれも符号付き数値または一重引用符で囲んだ文字列です。*lower* の値は、*upper* の値より小さくなければなりません。

文字列値に一重引用符を埋め込むには (例 : `O'Brien`)、一重引用符の前にバックスラッシュを 2 つ入れます (例 : `O\\'Brien`)。

関連する `FIELD` のデータ型の最小値を示すには、値 `MIN` を使用します。文字列と carray の最小値にはヌル文字列を指定します。文字フィールドの最小値には 0 を指定します。数値の場合、これはフィールドに格納できる最小値です。

関連する `FIELD` のデータ型の最大値を示すには、値 `MAX` を使用します。文字列と carray の最大値には、8 進数値の 255 文字の無限文字列を指定します。文字フィールドの最大値には、単一の 8 進数値の 255 文字を指定します。数値の場合は、数値としてフィールドに格納できる最大値です。したがって、`"MIN - -5"` は -5 以下のすべての数値を指し、`"6 - MAX"` は、6 以上のすべての数値を指すことになります。範囲内のメタキャラクタ `*` (ワイルドカード) は、既にエントリとして指定した範囲では使用されなかった任意の値を示します。各エントリでは、1 つのワイルドカードによる範囲指定だけが可能です。1 つのエントリで使用できるワイルドカード範囲は 1 つだけで、最後になければなりません (後続の範囲は無視される)。

数値ルーティング・フィールドには数値で範囲を指定し、文字列ルーティング・フィールドには、文字列で範囲を指定する必要があります。文字列で範囲を設定する場合は、文字列、carray、および文字フィールド型の値を一重引用符で囲みます。先頭に符号を付けることはできません。short 型および long 型の整数値は数字の文字列であり、オプションで先頭に正の符号または負の符号を付けることができます。浮動小数点数は、C コンパイラまたは `atof(3)` で受け入れられる形式で指定します。つまり、符号 (オプション)、数字の文字列 (オプションで小数点を追加)、`e` または `E` (オプション)、符号またはスペース (オプション)、整数という形式で指定します。

フィールド値が範囲と一致する場合、関連付けられているリモート・ドメイン・アクセス・ポイントは、要求がルーティングされるリモート・ドメインを示します。リモート・ドメイン・アクセス・ポイントの値に "*" を指定すると、ゲートウェイ・グループが認識する任意のリモート・ドメインに要求が送られます。

BUFTYPE = "*type1*[:*subtype1*[, *subtype2* . . .]][;*type2*[:*subtype3*[, . . .]]] . . ."

このルーティング・エントリで有効なデータ・バッファのタイプとサブタイプのリストを指定します。タイプは、FML、FML32、VIEW、VIEW32、X_C_TYPE、X_COMMON、または XML に制限されています。FML、FML32、または XML に対してはサブタイプを指定できず、VIEW、VIEW32、X_C_TYPE、および X_COMMON ではサブタイプを指定する必要があります ("*" は使用できません)。タイプとサブタイプのペアのうち、重複するものは同じルーティング基準名として指定できません。タイプとサブタイプのペアが一意の場合、複数のルーティング・エントリは同じ基準名を持つことができます。このパラメータは必須です。単一のルーティング・エントリに複数のバッファ・タイプが指定される場合、各バッファ・タイプに対するルーティング・フィールドのデータ型は同じでなければなりません。

フィールド値が設定されていないか (FML または FML32 バッファの場合)、または特定の範囲と一致しておらず、ワイルドカードの範囲が指定されていない場合、リモート・サービスの実行を要求したアプリケーション・プロセスに対してエラーが返されます。

DM_ACCESS_CONTROL セクション

このセクションでは、1 つまたは複数のアクセス制御リスト (ACL) の名前を指定し、各 ACL 名に 1 つまたは複数のリモート・ドメイン・アクセス・ポイントを関連付けます。ACL=*ACL_NAME* に設定すると、DM_EXPORT セクションの ACL パラメータを使用して、特定のローカル・ドメイン・アクセス・ポイントから *ACL_NAME* に関連付けられているリモート・ドメイン・アクセス・ポイントにエクスポートされるローカル・サービスへのアクセスを制限できます。

DM_ACCESS_CONTROL セクションのエントリの形式は次のとおりです。

ACL_NAME required_parameters

ACL_NAME はアクセス制御リストを指定するための識別子です。長さは 15 文字までです。

DM_ACCESS_CONTROL セクションの必須パラメータは次の 1 つだけです。

ACLIST = *identifier*[,*identifier*]

ACLIST には、1 つまたは複数のリモート・ドメイン・アクセス・ポイント名をカンマで区切って指定します。ワイルドカード文字 (*) を使用すると、DM_REMOTE セクションで定義したすべてのリモート・ドメイン・アクセス・ポイントが特定のローカル・ドメイン・アクセス・ポイントからエクスポートされるローカル・サービスにアクセスできます。

DM_TDOMAIN セクション

このセクションでは、TDomain ゲートウェイのネットワーク固有の情報を定義します。リモート・ドメインからローカル・サービスへの要求がローカル・ドメイン・アクセス・ポイントで受け付けられる場合、DM_TDOMAIN セクションには、ローカル・ドメインごとに 1 つのエントリがなければなりません。また、ローカル・ドメインからリモート・サービスへの要求がリモート・ドメイン・アクセス・ポイントで受け付けられる場合、そのアクセス・ポイントごとに 1 つのエントリがなければなりません。

DM_TDOMAIN セクションは、アクセス・ポイント・エントリの次のネットワーク・プロパティをコンフィギュレーションするために使用します。

- ローカル・ドメイン・アクセス・ポイント・エントリの場合、受信する接続指示を受け付けるためのネットワーク・アドレス。
- リモート・ドメイン・アクセス・ポイント・エントリの場合、そのアクセス・ポイントに関連付けられているリモート・ドメインに接続するために使用するネットワーク・アドレス。
- ローカルまたはリモート・ドメイン・アクセス・ポイント・エントリの場合、TDomain ゲートウェイが接続を試行するための条件。このオプションのコンフィギュレーションは、BEA Tuxedo 8.1 以降のアプリケーションでのみ使用できます。
- ローカルまたはリモート・ドメイン・アクセス・ポイント・エントリの場合、TDomain ゲートウェイがリモート・ドメインへの接続でキープアライブ・メッセージを送信するかどうか。このオプションのコンフィギュレーションは、BEA Tuxedo 8.1 以降のアプリケーションでのみ使用できます。

DM_TDOMAIN セクションのエントリの形式は次のとおりです。

AccessPoint required_parameters [*optional_parameters*]

AccessPoint は、ローカル・ドメイン・アクセス・ポイントまたはリモート・ドメイン・アクセス・ポイントの識別子の値です。*AccessPoint* 識別子は、DM_LOCAL セクションに定義されているローカル・ドメイン・アクセス・ポイントか、または DM_REMOTE セクションに定義されているリモート・ドメイン・アクセス・ポイントと一致する必要があります。

DM_TDOMAIN セクションの必須パラメータ

NWADDR = *string*[1..256] (BEA Tuxedo 8.0 以前では最大 78 バイト)

ローカルまたはリモート・ドメイン・アクセス・ポイントに関連付けるネットワーク・アドレスを指定します。ローカル・ドメイン・アクセス・ポイントの場合、ほかの BEA Tuxedo アプリケーションからの接続指示を受け付けるためのアドレスを指定します。リモート・ドメイン・アクセス・ポイント・エントリの場合、リモート・ドメイン・アクセス・ポイントに関連付けられている BEA Tuxedo アプリケーションに接続するときに使用するアドレスを指定します。このパラメータの値は、すべての DM_TDOMAIN エントリ間で一意でなければなりません。

string の形式が "0x*hex-digits*" または "\\x*hex-digits*" の場合、偶数の有効な 16 進数桁を含める必要があります。これらの形式は、TCP/IP アドレスを含む文字配列に内部変換されます。*string* の値は次のいずれかの形式で指定します。

"//*hostname*:*port_number*"
"//*#.#.#.#*:*port_number*"

最初の形式では、gethostbyname(3c) を介してアクセスされたローカル設定の名前解決機能を使ってアドレスが結合されるときに、*hostname* は TCP/IP ホスト・アドレスに解決されます。*#.#.#.#* はドットで区切った 10 進数の形式で、各 *#* は 0 から 255 までの 10 進数です。

Port_number は、0 ～ 65535 の 10 進数です。

注記　一部のポート番号は、お使いのシステムで使用されるトランスポート・プロトコル (TCP/IP など) のために予約されている場合があります。予約されているポート番号を確認するには、トランスポート・プロトコルのマニュアルを調べてください。

DM_TDOMAIN セクションのオプション・パラメータ

NWDEVICE = *string*[1..78]

このローカルまたはリモート・ドメイン・アクセス・ポイントのネットワーク・アドレスにバインディングするときに使用するネットワーク・デバイスを指定します。ローカル・ドメイン・アクセス・ポイント・エントリの場合、この属性は接続指示を受け付けるために使用す

るデバイスを指定します。リモート・ドメイン・アクセス・ポイントの場合、リモート・ドメイン・アクセス・ポイントに接続するために使用するデバイスを指定します。

NWDEVICE パラメータの指定は必須ではありません。以前のバージョンでは、TLI 対応のネットワーキング機能に対しては、デバイス名に絶対パス名を指定する必要があります。

CMPLIMIT = *numeric*

このリモート・ドメイン・アクセス・ポイントにデータを送信するときに使用する圧縮しきい値を指定します。このパラメータは、リモート・ドメイン・アクセス・ポイントにのみ適用されます。最小値は 0、最大値は 2147483647 で、デフォルトは 2147483647 です。CMPLIMIT の値より大きいアプリケーション・バッファは圧縮されます。

MINENCRYPTBITS = {0 | 40 | 56 | 128}

このリモート・ドメイン・アクセス・ポイントに関連付けられているリモート・ドメインへのネットワーク・リンクを確立するときに必要な最小暗号化レベルを指定します。このパラメータは、リモート・ドメイン・アクセス・ポイントにのみ適用されます。

0 は暗号化が行われないことを意味し、40、56、または 128 は暗号化キーの長さをビット単位で指定します。40 ビットの値は、下位互換性のために用意されています。デフォルトは 0 です。ここで指定する最小レベルの暗号化が満たされない場合、リンクの確立は失敗します。

MAXENCRYPTBITS = {0 | 40 | 56 | 128}

このリモート・ドメイン・アクセス・ポイントに関連付けられているリモート・ドメインへのネットワーク・リンクを確立するときに必要な最大暗号化レベルを指定します。このパラメータは、リモート・ドメイン・アクセス・ポイントにのみ適用されます。

0 は暗号化が行われないことを意味し、40、56、または 128 は暗号化キーの長さをビット単位で指定します。40 ビットの値は、下位互換性のために用意されています。デフォルトは 128 です。

CONNECTION_POLICY = {LOCAL | ON_DEMAND | ON_STARTUP | INCOMING_ONLY}

このローカルまたはリモート・ドメイン・アクセス・ポイントに関連付けられている TDomain ゲートウェイが接続を確立するための条件を指定します。有効な値は、LOCAL、ON_DEMAND、ON_STARTUP、または

INCOMING_ONLY です。LOCAL は、リモート・ドメイン・アクセス・ポイントにのみ適用されます。

BEA Tuxedo 8.1 以降のソフトウェアを実行する場合、CONNECTION_POLICY パラメータは DM_TDOMAIN セクションでも指定できます。特定のローカルまたはリモート・ドメイン・アクセス・ポイントの DM_TDOMAIN セクションの値は、DM_LOCAL セクションのグローバル値に優先します。グローバル接続ポリシーを無効にできるので、リモート・ドメイン単位で接続ポリシーをコンフィギュレーションできます。

ローカル・ドメイン・アクセス・ポイントの接続ポリシーを指定しない場合、デフォルトとして DM_LOCAL セクションに指定されるグローバル接続ポリシーが使用されます。DM_TDOMAIN セクションにグローバル接続ポリシーを指定する場合、DM_LOCAL セクションにグローバル接続ポリシーを指定しないでください。

接続ポリシーが LOCAL の場合、リモート・ドメイン・アクセス・ポイントは DM_LOCAL セクションに指定されるグローバル接続ポリシーを受け入れます。LOCAL は、リモート・ドメイン・アクセス・ポイントに対するデフォルトの接続ポリシーです。LOCAL を除き、リモート・ドメイン・アクセス・ポイントに対する接続ポリシーは、ローカル・ドメイン・アクセス・ポイントに対する接続ポリシーに優先します。

接続ポリシーが ON_DEMAND の場合、クライアントがリモート・サービスを要求したとき、または dmadmin(1) connect コマンドが実行されたときにのみ、TDomain ゲートウェイは接続を試行します。接続ポリシーが ON_DEMAND の場合、再接続は行われません。

接続ポリシーが ON_STARTUP の場合、TDomain ゲートウェイはゲートウェイ・サーバの初期化時に接続を試行します。ON_STARTUP に設定した場合、リモート・ドメインへの接続が確立された場合にのみそのリモート・サービス (TDomain ゲートウェイによって宣言されたサービス) が宣言されますつまり、リモート・ドメインとの接続が確立されていないと、リモート・サービスは中断されます。デフォルトでは、失敗した接続が 60 秒おきに再試行されるよう設定されています。再接続の間隔は、DM_TDOMAIN セクションの RETRY_INTERVAL パラメータで変更できます。このセクションの MAXRETRY パラメータも参照してください。

接続ポリシーが INCOMING_ONLY の場合、TDomain ゲートウェイは起動時にリモート・ドメインへの接続を試みません。このため、リモート・サービスは最初は中断されています。TDomain ゲートウェイは、リモート・ドメインからの接続を受信したときに利用可能になります。リモート・サービスは、ドメイン・ゲートウェイが接続を受信したときか、dmadmin(1) connect コマンドで管理接続が確立されたときに宣言されます。接続ポリシーが INCOMING_ONLY の場合、再接続は行われません。

MAXRETRY = {*numeric* | MAXLONG}

このローカルまたはリモート・ドメイン・アクセス・ポイントに関連付けられている TDomain ゲートウェイが接続を試行する回数を指定します。このパラメータは、BEA Tuxedo 8.1 以降のソフトウェアが実行されているときに TDOMAIN セクションで使用でき、このアクセス・ポイントの CONNECTION_POLICY パラメータが ON_STARTUP に設定されている場合に有効です。それ以外の接続ポリシーの場合、自動再試行は無効になります。

MAXRETRY の最小値は 0 で、最大値は MAXLONG (2147483647) です。MAXLONG (デフォルト) の場合、再接続処理が無限に繰り返されるか、または接続が確立されるまで繰り返されます。

RETRY_INTERVAL = *numeric*

このローカルまたはリモート・ドメイン・アクセス・ポイントに関連付けられている TDomain ゲートウェイが接続を自動的に試行する間隔を指定します。このパラメータは、BEA Tuxedo 8.1 以降のソフトウェアが実行されているときに TDOMAIN セクションで使用でき、このアクセス・ポイントの CONNECTION_POLICY パラメータが ON_STARTUP に設定されている場合に有効です。それ以外の接続ポリシーの場合、自動再試行は無効になります。

RETRY_INTERVAL の最小値は 0 、最大値は 2147483647 です。デフォルトは 60 です。MAXRETRY を 0 に設定すると、RETRY_INTERVAL は設定できません。

TCPKEEPALIVE = {LOCAL | NO | YES}

ローカルまたはリモート・ドメイン・アクセス・ポイントの TCP レベル・キープアライブを有効にします。有効な値は、LOCAL, N (NO)、または Y (YES) です。LOCAL は、リモート・ドメイン・アクセス・ポイントにのみ適用されます。

TCPKEEPALIVE パラメータは、BEA Tuxedo 8.1 以降のソフトウェアが実行されている TDOMAIN タイプのドメイン・ゲートウェイにのみ適用されます。リモート・ドメイン・アクセス・ポイントに対するこの値は、ローカル・ドメイン・アクセス・ポイントに対する値に優先します。ローカル・ドメイン・アクセス・ポイント値を無効にできるので、リモート・ドメイン単位で TCP レベル・キープアライブをコンフィギュレーションできます。

LOCAL を指定すると、リモート・ドメイン・アクセス・ポイントは、ローカル・ドメイン・アクセス・ポイントに対して定義されている TCP レベル・キープアライブ値を受け入れます。LOCAL は、リモート・ドメイン・アクセス・ポイントに対するデフォルトの TCP レベル・キープアライブ値です。

NO を指定すると、このアクセス・ポイントに対する TCP レベル・キープアライブが無効になります。N は、ローカル・ドメイン・アクセス・ポイントに対するデフォルトの TCP レベル・キープアライブ値です。

YES を指定すると、このアクセス・ポイントに対する TCP レベル・キープアライブが有効になります。接続の TCP レベル・キープアライブが有効になった場合、その接続のキープアライブ間隔は、オペレーティング・システムの TCP キープアライブ・タイマ用にコンフィギュレーションされているシステム・レベル値です。この間隔は、TDomain ゲートウェイが接続でトラフィックを受信せずに待機する最長時間です。この最長時間を超えると、ゲートウェイは TCP レベル・キープアライブ要求メッセージを送信します。接続がまだオープンしており、リモート TDomain ゲートウェイが正常に動作している場合、リモート・ゲートウェイは肯定応答を返信します。ローカル TDomain ゲートウェイは、要求メッセージを送信してから一定時間内に肯定応答を受信しなかった場合、接続が切断されたと見なして、その接続に関連するすべてのリソースを解放します。

TCP レベル・キープアライブを使用すると、BEA Tuxedo のドメイン間接続を非アクティブな期間にわたってオープンにできるだけでなく、TDomain ゲートウェイが接続の障害を迅速に検出できるようになります。

注記　TCPKEEPALIVE と DMKEEPALIVE は、相互に排他的ではありません。つまり、両方のパラメータを使用してドメイン間接続をコンフィギュレーションできます。

DMKEEPALIVE = *numeric*

ローカルまたはリモート・ドメイン・アクセス・ポイントのアプリケーション・レベル・キープアライブを制御します。この値は、-1 以上 2147483647 以下でなければなりません。値 -1 は、リモート・ドメイン・アクセス・ポイントにのみ適用されます。

DMKEEPALIVE パラメータは、BEA Tuxedo 8.1 以降のソフトウェアが実行されている TDOMAIN タイプのドメイン・ゲートウェイにのみ適用されます。リモート・ドメイン・アクセス・ポイントに対するこの値は、ローカル・ドメイン・アクセス・ポイントに対する値に優先します。ローカル・ドメイン・アクセス・ポイント値を無効にできるので、リモート・ドメイン単位で アプリケーション・レベル・キープアライブをコンフィギュレーションできます。

-1 を指定すると、リモート・ドメイン・アクセス・ポイントは、ローカル・ドメイン・アクセス・ポイントに対して定義されているアプリケーション・レベル・キープアライブ値を受け入れます。-1 は、リモート・ドメイン・アクセス・ポイントに対するデフォルトのアプリケーション・レベル・キープアライブ値です。

0 を指定すると、このアクセス・ポイントに対するアプリケーション・レベル・キープアライブが無効になります。0 は、ローカル・ドメイン・アクセス・ポイントに対するデフォルトのアプリケーション・レベル・キープアライブ値です。

1 以上 2147483647 以下の値 (単位はミリ秒で、Domains ソフトウェアによって最も近い秒数に切り上げられる) を指定すると、このアクセス・ポイントに対するアプリケーション・レベル・キープアライブが有効になります。指定した時間は、TDomain ゲートウェイが接続でトラフィックを受信せずに待機する最長時間です。この最長時間を超えると、ゲートウェイはアプリケーション・レベル・キープアライブ要求メッセージを送信します。接続がまだオープンしており、リモート TDomain ゲートウェイが正常に動作している場合、リモート・ゲートウェイは肯定応答を返信します。ローカル TDomain ゲートウェイは、要求メッセージを送信してから指定の時間内 (DMKEEPALIVEWAIT パラ

メータを参照) に肯定応答を受信しなかった場合、接続が切断されたと見なして、その接続に関連するすべてのリソースを解放します。

アプリケーション・レベル・キープアライブを使用すると、BEA Tuxedo のドメイン間接続を非アクティブな期間にわたってオープンにできるだけでなく、TDomain ゲートウェイが接続の障害を迅速に検出できるようになります。

注記　DMKEEPALIVE と TCPKEEPALIVE は、相互に排他的ではありません。つまり、両方のパラメータを使用してドメイン間接続をコンフィギュレーションできます。

DMKEEPALIVEWAIT = *numeric*

このローカルまたはリモート・ドメイン・アクセス・ポイントに関連付けられている TDomain ゲートウェイが送信したキープアライブ・メッセージに対する肯定応答を受信するまでの待ち時間を指定します。この値は、0 以上 2147483647 以下でなければなりません (単位はミリ秒で、Domains ソフトウェアによって最も近い秒数に切り上げられる)。デフォルトは 0 です。このパラメータは、BEA Tuxedo 8.1 以降のソフトウェアが実行されている TDOMAIN タイプのドメイン・ゲートウェイにのみ適用されます。

このアクセス・ポイントに対する DMKEEPALIVE が 0 (キープアライブが無効) の場合、DMKEEPALIVEWAIT の設定は無効です。

このアクセス・ポイントに対する DMKEEPALIVE を有効にし、DMKEEPALIVEWAIT を DMKEEPALIVE より大きい値に設定した場合、ローカル TDomain ゲートウェイは DMKEEPALIVEWAIT タイマが期限切れになるまでに複数のアプリケーション・レベル・キープアライブ・メッセージを送信します。このような設定の組み合わせも可能です。

このアクセス・ポイントに対する DMKEEPALIVE を有効にし、DMKEEPALIVEWAIT を 0 に設定した場合、送信されたキープライブ・メッセージに対する肯定応答は意味を持ちません。こうした肯定応答は、TDomain ゲートウェイによってすべて無視されます。ゲートウェイは、DMKEEPALIVE タイマがタイム・アウトするたびにキープライブ・メッセージを送信します。この設定の組み合わせは、ファイアウォールを介したアイドル接続を保持するために使用します。

DM_TDOMAIN セクション内の同じアクセス・ポイントに対する複数のエントリ

この DM_TDOMAIN エントリが (DM_LOCAL セクションで指定された) ローカル・ドメイン・アクセス・ポイントの場合、その NWADDR は接続指示を受け付けるためのネットワーク・アドレスです。DM_TDOMAIN セクションでは、ローカル・ドメイン・アクセス・ポイントに関連付けられたエントリを複数回指定して、ローカル・ドメイン・アクセス・ポイントに関連付けられているサービスを BEA Tuxedo ドメイン内の別のマシンに移行できます。

リモート・ドメイン・アクセス・ポイント (DM_REMOTE セクションで指定) に関連付けられたエントリも、DM_TDOMAIN セクションで複数回指定できます。最初のエントリは、一次アドレスと見なされます。つまり、その NWADDR は、リモート・ドメイン・アクセス・ポイントへの接続が最初に試行されるときのネットワーク・アドレスです。2 番目のエントリは、二次アドレスと見なされます。つまり、その NWADDR は、一次アドレスを使用して接続を確立できないときの次のネットワーク・アドレスです。

この DM_TDOMAIN エントリがリモート・ドメイン・アクセス・ポイントの別のオカレンスの場合、このエントリは、一次リモート・ゲートウェイが存在する BEA Tuxedo ドメインとは別の BEA Tuxedo ドメインに存在する必要がある二次リモート・ゲートウェイを指し示します。二次および一次リモート・ゲートウェイの ACCESSPOINTID は、それぞれに対応する DMCONFIG ファイルの DM_LOCAL セクションで同じ値でなければなりません。この仕組みはミラー・ゲートウェイと呼ばれます。この機能は、トランザクションや会話の際に使用しないようにしてください。また、一次リモート・ゲートウェイが使用できるときには、ミラー・ゲートウェイの使用はお勧めできません。

注記　DM_TDOMAIN セクションのローカルまたはリモート・ドメイン・アクセス・ポイントの複数のエントリの場合、NWADDR パラメータの複数のインスタンスだけが Domains ソフトウェアによって読み取られます。他のパラメータの複数のインスタンスの場合、パラメータの最初のインスタンスだけが Domains ソフトウェアによって読み取られ、それ以外のインスタンスはすべて無視されます。

DMCONFIG(5) に関する追加情報

ファイル　BDMCONFIG 環境変数は、BDMCONFIG コンフィギュレーション・ファイルを検索するために使用します。

例 1　以下は、5 つのサイトの Domains コンフィギュレーションを定義するコンフィギュレーション・ファイルの例です。この例は、Central Bank Branch と通信する 4 つの銀行支店ドメインを示しています。3 つの銀行支店は、ほかの BEA Tuxedo ドメイン内で動作しています。4 つ目の支店は、別の TP ドメインの制御下で動作しています。そのドメインと Central Bank との通信には OSI TP が使用されています。この例は、Central Bank から見た Domains コンフィギュレーション・ファイルを示しています。

```
# Central Bank 用の BEA Tuxedo Domains コンフィギュレーション・ファイル
#
#
*DM_LOCAL
#
DEFAULT: SECURITY = NONE

c01  GWGRP = bankg1
     TYPE = TDOMAIN
     ACCESSPOINTID = "BA.CENTRAL01"
     DMTLOGDEV = "/usr/apps/bank/DMTLOG"
     DMTLOGNAME = "DMTLG_C01"

c02  GWGRP = bankg2
     TYPE = OSITP
     ACCESSPOINTID = "BA.CENTRAL02"
     DMTLOGDEV = "/usr/apps/bank/DMTLOG"
     DMTLOGNAME = "DMTLG_C02"

#
*DM_REMOTE
#
b01  TYPE = TDOMAIN
     ACCESSPOINTID = "BA.BANK01"

b02  TYPE = TDOMAIN
     ACCESSPOINTID = "BA.BANK02"

b03  TYPE = TDOMAIN
     ACCESSPOINTID = "BA.BANK03"

b04  TYPE = OSITP
```

```
     ACCESSPOINTID = "BA.BANK04"

*DM_TDOMAIN
#
# ローカル・ネットワーク・アドレス
c01  NWADDR = "//newyork.acme.com:65432"  NWDEVICE ="/dev/tcp"

# リモート・ネットワーク・アドレス
b01  NWADDR = "//192.11.109.5:1025"  NWDEVICE = "/dev/tcp"
b02  NWADDR = "//dallas.acme.com:65432"  NWDEVICE = "/dev/tcp"
b03  NWADDR = "//192.11.109.156:4244"  NWDEVICE = "/dev/tcp"

*DM_OSITP
#
c02  APT = "BA.CENTRAL01"
     AEQ = "TUXEDO.R.4.2.1"
     AET = "{1.3.15.0.3},{1}"
     ACN = "XATMI"
b04  APT = "BA.BANK04"
     AEQ = "TUXEDO.R.4.2.1"
     AET = "{1.3.15.0.4},{1}"
     ACN = "XATMI"

*DM_EXPORT
#
open_act ACL = branch
close_act ACL = branch
credit
debit
balance
loan   LACCESSPOINT = c02  ACL = loans

*DM_IMPORT
#
tlr_add LACCESSPOINT = c01  ROUTING = ACCOUNT
tlr_bal LACCESSPOINT = c01  ROUTING = ACCOUNT
tlr_add RACCESSPOINT = b04  LACCESSPOINT = c02  RNAME ="TPSU002"
tlr_bal RACCESSPOINT = b04  LACCESSPOINT = c02  RNAME ="TPSU003"
tlr_bal RACCESSPOINT = b02,b03"  LACCESSPOINT = c02

*DM_ROUTING
#
ACCOUNT FIELD = branchid BUFTYPE = "VIEW:account"
   RANGES = "MIN-1000:b01,1001-3000:b02,*:b03"

*DM_ACCESS_CONTROL
#
branch ACLIST = "b01,b02,b03"
loans  ACLIST = b04
```

例 2　この例は、1 つの Bank Branches (BANK01) の BEA Tuxedo Domains コンフィギュレーション・ファイルを示しています。

```
#
#Bank Branch 用の BEA Tuxedo Domains コンフィギュレーション・ファイル
#
#
*DM_LOCAL
#
b01  GWGRP = auth
     TYPE = TDOMAIN
     ACCESSPOINTID = "BA.BANK01"
     DMTLOGDEV = "/usr/apps/bank/DMTLOG"

*DM_REMOTE
#
c01  TYPE = TDOMAIN
     ACCESSPOINTID = "BA.CENTRAL01"

*DM_TDOMAIN
#
b01  NWADDR = "//192.11.109.156:4244"  NWDEVICE = "/dev/tcp"
c01  NWADDR = "//newyork.acme.com:65432"  NWDEVICE ="/dev/tcp"
*DM_EXPORT
#
tlr_add   ACL = central
tlr_bal   ACL = central

*DM_IMPORT
#

OPA001   RNAME = "open_act"
CLA001   RNAME = "close_act"
CRD001   RNAME = "credit"
DBT001   RNAME = "debit"
BAL001   RNAME = "balance"

*DM_ACCESS_CONTROL
#
central   ACLIST = c01
```

ネットワーク・アドレス　TDomain を実行するローカル・マシンが TCP/IP アドレッシングを使用していて、アドレスは 155.2.193.18 で、backus.company.com という名前になっているとします。さらに、TDomain が要求を受け取るポート番号は 2334 であるとします。このポート番号 2334 は、bankapp-tuxwsvr という名前のネットワーク・サービス・データベースに追加されているとします。この場合、アドレスは次のように表現されます。

```
//155.2.193.18:bankapp-gwtaddr

//155.2.193.18:2334

//backus.company.com:bankapp-gwtaddr

//backus.company.com:2334

0x0002091E9B02C112
```

最後の表現は 16 進形式です。0002 は TCP/IP アドレスの先頭部分、091E は 16 進数に変換されたポート番号 2334 です。その後、IP アドレス 155.2.193.12 の各要素は 16 進数に変換されています。つまり、155 は 9B、2 は 02 というようになります。

関連項目 dmadmin(1)、dmloadcf(1)、dmunloadcf(1)、tmboot(1), tmshutdown(1)、DMADM(5)、DMCONFIG for GWTOPEND(5)、GWADM(5)、GWTDOMAIN(5)

『BEA Tuxedo アプリケーションの設定』

『BEA Tuxedo アプリケーション実行時の管理』

『BEA Tuxedo Domains コンポーネント』

『C 言語を使用した BEA Tuxedo アプリケーションのプログラミング』

DMCONFIG for GWTOPEND(5)

名前 DMCONFIG for GWTOPEND—BEA TOP END ドメイン・ゲートウェイのテキスト形式のドメイン・コンフィギュレーション・ファイル

機能説明 Domains コンフィギュレーションは、BEA Tuxedo Domains コンポーネントを使用して通信およびサービス共有を行うことができる 2 つ以上のドメイン (ビジネス・アプリケーション) の集まりです。複数のドメインを接続する方法や複数のドメイン間で相互にアクセスできるサービスについては、Domains コンフィギュレーションに参加する各 BEA Tuxedo ドメインの Domains コンフィギュレーション・ファイルで定義されます。テキスト形式の Domains コンフィギュレーション・ファイルは DMCONFIG と呼ばれますが、ファイルの内容がこのリファレンス・ページで説明する形式に従っている限り、任意の名前を付けることができます。

DMCONFIG ファイルは、dmloadcf(1) ユーティリティによって構文解析され、バイナリ形式のファイル BDMCONFIG にロードされます。DMCONFIG ファイルと同様、BDMCONFIG ファイルがどのような名前であっても、実際の名前は BDMCONFIG 環境変数で指定されたデバイス・ファイル名またはシステム・ファイル名になります。BDMCONFIG ファイルは、Domains コンフィギュレーションに参加する Tuxedo ドメインごとに 1 つ必要です。

DMCONFIG ファイルと BDMCONFIG ファイルの関係は、BEA Tuxedo ドメインの定義に使用される UBBCONFIG ファイルと TUXCONFIG ファイルの関係に似ています。UBBCONFIG ファイルと TUXCONFIG ファイルについては、UBBCONFIG(5) を参照してください。

GWTOPEND ファイル用の DMCONFIG ファイルの詳細 (例を含む) については、160 ページの「DMCONFIG for GWTOPEND(5) に関する追加情報」を参照してください。BEA TOP END ドメイン・ゲートウェイの詳細については、『ATMI アプリケーションでの BEA Tuxedo TOP END Domain Gateway の使用』を参照してください。

定義 BEA Tuxedo ドメインは、単一の TUXCONFIG ファイルに記述された環境として定義されます。BEA Tuxedo 用語では、ドメインとアプリケーション (ビジネス・アプリケーション) は同義です。

Domains コンフィギュレーションに含まれる各 BEA Tuxedo ドメインでは、1 つの Domains 管理サーバ (DMADM) プロセスが実行されます。DMADM は、特定の BEA Tuxedo ドメインで実行されるすべてのドメイン・ゲートウェイ・グループ用の管理サーバです。

ドメイン・ゲートウェイ・グループは、BEA Tuxedo システムのゲートウェイ管理サーバ (GWADM) プロセスと BEA Tuxedo システムのドメイン・ゲートウェイ・プロセスで構成されます。

BEA Tuxedo システムのドメイン・ゲートウェイ・プロセスは、特定のタイプのトランザクション処理 (TP) ドメインとの通信サービスを提供します。たとえば、GWTOPEND プロセスを使用すると、BEA Tuxedo アプリケーションは BEA TOP END アプリケーションと通信できます。ドメイン・ゲートウェイは、別のドメインへの要求を中継し、応答を受信します。

ローカル・ドメイン・アクセス・ポイントは、他のドメイン (リモート・ドメイン) が使用できる BEA Tuxedo ドメインの一連のサービスを表すユーザ指定の論理名です。ローカル・ドメイン・アクセス・ポイントはドメイン・ゲートウェイ・グループにマップされるため、どちらも同義語として使用されます。

リモート・ドメイン・アクセス・ポイントは、ローカル・ドメインが使用できるリモート・ドメインの一連のサービスを表すユーザ指定の論理名です。リモート・ドメインは、別の BEA Tuxedo アプリケーションまたは BEA TOP END などの別の TP システムで動作するアプリケーションです。

コンフィギュレーション・ファイルの目的

DMCONFIG ファイルは、次の目的で使用します。

- リモート・ドメインのアプリケーション・クライアントがローカル・ドメインのサービスまたはキューにアクセスするためのローカル・ドメイン・アクセス・ポイントを定義する。
- 各ローカル・ドメイン・アクセス・ポイントを介して使用できるローカル・サービスまたはキューを定義する。
- ローカル・ドメインのアプリケーション・クライアントがリモート・ドメインのサービスにアクセスするためのリモート・ドメイン・アクセス・ポイントを定義する。
- 各リモート・ドメイン・アクセス・ポイントを介して使用できるリモート・サービスを定義する。

- ローカル・ドメイン・アクセス・ポイントとリモート・ドメイン・アクセス・ポイントを特定のドメイン・ゲートウェイ・グループとネットワーク・アドレスにマップする。

コンフィギュレーション・ファイルの形式

DMCONFIG ファイルは、次のセクションで構成されます。

- DM_LOCAL (DM_LOCAL_DOMAINS ともいう)
- DM_REMOTE (DM_REMOTE_DOMAINS ともいう)
- DM_EXPORT (DM_LOCAL_SERVICES ともいう)
- DM_IMPORT (DM_REMOTE_SERVICES ともいう)
- DM_RESOURCES
- DM_ROUTING
- DM_ACCESS_CONTROL
- DM_TOPEND (TOPEND タイプのドメイン・ゲートウェイ用のセクション)
- DM_*dom* (*dom* は他のドメイン・ゲートウェイ・タイプのセクション (TDOMAIN、SNACRM、SNASTACKS、SNALINKS、OSITP、OSITPX) のいずれか)

DMCONFIG ファイル内のアスタリスク (*) で始まる行は、指定セクションの開始を表します。アスタリスク (*) の直後にはセクション名が表示されます。アスタリスクは、セクション名を指定するときに必要です。DM_LOCAL セクションは、DM_REMOTE セクションの前になければなりません。

このリファレンス・ページでは、GWTOPEND ゲートウェイ・プロセスによってインプリメントされる TOPEND (TEDG ゲートウェイ) をコンフィギュレーションする方法について説明します。TDOMAIN ドメイン・ゲートウェイのコンフィギュレーション方法については、DMCONFIG(5) を参照してください。SNAX、OSITP、または OSITPX ドメイン・ゲートウェイのコンフィギュレーションについては、BEA eLink Documentation を参照してください。

パラメータは通常、*KEYWORD* = *value* という形式で指定します。等号 (=) の前後には空白またはタブ文字を使用できます。この形式により、*KEYWORD* が *value* に設定されます。有効なキーワードについては、以下の各セクションで説明します。

予約語の DEFAULT で始まる行にはパラメータ仕様が含まれており、セクション内の以降の該当するすべての行に対して適用されます。デフォルト仕様はすべてのセクションで使用でき、同じセクション内で複数回使用できます。これらの行のフォーマットは次のとおりです。

```
DEFAULT: [KEYWORD1 = value1 [KEYWORD2 = value2 [...]]]
```

この行で設定した値は、別の DEFAULT 行によってリセットされるか、セクションが終わるまで有効です。これらの値は、DEFAULT でない行のオプション・パラメータによって無効になる場合もあります。DEFAULT でない行におけるパラメータ設定は、その行でのみ有効です。以降の行ではデフォルト設定に戻ります。DEFAULT が行頭に表示されると、それ以前に設定されたすべてのデフォルト値はクリアされ、システムのデフォルト値に戻ります。

値が *numeric* の場合は、C の標準表記法を使用して基数を示します。つまり、基数 16 (16 進) の接頭辞は 0x、基数 8 (8 進) の接頭辞は 0、基数 10 (10 進) には接頭辞が付きません。数値パラメータで指定できる範囲については、各パラメータの項で説明します。

値が *identifier* (TYPE パラメータの TOPEND のように BEA Tuxedo Domains コンポーネントにとって既知の文字列値) の場合、一般的に標準 C 規則が使用されます。標準 C の *identifier* の先頭には英字またはアンダースコア (_) を使用し、以降の識別子には英数字またはアンダースコアを使用する必要があります。identifier の長さは最大 30 バイトです (最後のヌルを除く)。

識別子を二重引用符で囲む必要はありません。整数でも識別子でもない値は、二重引用符で囲む必要があります。

入力フィールドは、1 つ以上の空白 (またはタブ) 文字で区切ります。

"#" はコメントを示します。復帰改行文字でコメントを終了します。

空白行とコメントは無視されます。

コメントは任意の行の最後に自由に入力できます。

行は、復帰改行の後に最低 1 つのタブを置いて継続できます。コメントを継続することはできません。

Domains 関連の新しい用語

BEA Tuxedo のリリース 7.1 以降では、ドメイン関連の用語の一部が変更されました。Domains 用の MIB では、新しいクラスおよび属性の用語を使用して、ローカル・ドメインとリモート・ドメイン間の対話を説明していま

す。新しい用語は、DMCONFIG(5) および DMCONFIG for GWTOPEND(5) リファレンス・ページ、セクション名、パラメータ名、エラー・メッセージ、および DM_MIB(5) リファレンス・ページ、クラス、エラー・メッセージに適用されます。

下位互換性のため、BEA Tuxedo 7.1 より前に使用されていた DMCONFIG 用語と Domains 用の MIB の新しい用語との間でエイリアスが提供されています。BEA Tuxedo リリース 7.1 以降の DMCONFIG では、両方のバージョンの用語を使用できます。次の表に、DMCONFIG ファイルの旧用語と新用語の対応を示します。

旧用語		新用語	
セクション名	パラメータ名	セクション名	パラメータ名
DM_LOCAL_DOMAINS		DM_LOCAL	
DM_REMOTE_DOMAINS		DM_REMOTE	
	DOMAINID		ACCESSPOINTID
	MAXRDOM		MAXACCESSPOINT
	MAXRDTRAN		MAXRAPTRAN
DM_LOCAL_SERVICES		DM_EXPORT	
DM_REMOTE_SERVICES		DM_IMPORT	
	LDOM		LACCESSPOINT
	RDOM		RACCESSPOINT

BEA Tuxedo リリース 7.1 以降では、dmunloadcf コマンドは、新しいドメイン用語を使用する DMCONFIG ファイルをデフォルトで生成します。旧ドメイン用語を使用する DMCONFIG ファイルを生成するには、-c オプションを使用します。次に例を示します。

```
prompt> dmunloadcf -c > dmconfig_prev
```

DM_LOCAL セクション

このセクション (DM_LOCAL_DOMAINS セクションともいう) では、1 つまたは複数のローカル・ドメイン・アクセス・ポイント識別子と、それらに関連付けるゲートウェイ・グループを定義します。このセクションには、UBBCONFIG ファイルで指定されたアクティブなゲートウェイ・グループごとにローカル・ドメイン・アクセス・ポイントのエントリが必要です。各エントリでは、グループで実行されるドメイン・ゲートウェイ・プロセスに必要なパラメータを指定します。

このエントリでは、1 つのリモート BEA TOP END システムに関連付けられる GWTOPEND ドメイン・ゲートウェイ・インスタンス (および対応する GWADM) を定義します。ローカル BEA Tuxedo アプリケーションは、同じ BEA TOP END システムの一部である TOPEND タイプのリモート・ドメインと通信します。BEA TOP END システムの名前は DM_TOPEND セクションで定義します。

DM_LOCAL セクションのエントリの形式は次のとおりです。

LocalAccessPoint *required_parameters* [*optional_parameters*]

LocalAccessPoint は、UBBCONFIG ファイルで定義された特定のゲートウェイ・グループを表すローカル・ドメイン・アクセス・ポイント識別子 (論理名) です。LocalAccessPoint は、Domains コンフィギュレーションに含まれるローカルおよびリモート・ドメイン間で一意でなければなりません。DM_EXPORT セクションで説明するとおり、ローカル・ドメイン・アクセス・ポイントはローカル・リソース (サービス、キュー) をゲートウェイ・グループに関連付けるために使用します。ローカル・ドメイン・アクセス・ポイントを介して使用できるローカル・リソースは、1 つまたは複数のリモート・ドメイン内のクライアントで使用できます。

DM_LOCAL セクションの必須 TEDG パラメータ

GWGRP = *identifier*

このローカル・ドメイン・アクセス・ポイントを表すドメイン・ゲートウェイ・グループの名前 (TUXCONFIG ファイルの GROUPS セクションで指定された名前) を指定します。ローカル・ドメイン・アクセス・ポイントとゲートウェイ・グループは、1 対 1 の関係です。

TYPE = *identifier*

このローカル・ドメイン・アクセス・ポイントに関連付けるドメイン・ゲートウェイのタイプを指定します。TYPE は、TOPEND、TDOMAIN、SNAX、OSITP、または OSITPX に設定できます。

TOPEND は、このローカル・ドメイン・アクセス・ポイントが GWTOPEND ドメイン・ゲートウェイ・インスタンスに関連付けられ、これによって BEA TOP END システムと通信できることを示します。

TDOMAIN は、このローカル・ドメイン・アクセス・ポイントが GWTDOMAIN ゲートウェイ・インスタンスに関連付けられ、これによって別の BEA Tuxedo アプリケーションと通信できることを示します。

SNAX は、このローカル・ドメイン・アクセス・ポイントが GWSNAX ゲートウェイ・インスタンスに関連付けられ、これによって別の TP ドメインに SNA プロトコルを介して通信できることを示します。

OSITP または OSITPX は、このローカル・ドメイン・アクセス・ポイントが GWOSITP ゲートウェイ・インスタンスに関連付けられ、これによって別の TP ドメインに OSI TP プロトコルを介して通信できることを示します。OSITP は OSI TP 1.3 プロトコルを使用することを示し、OSITPX は OSI TP 4.0 以降のプロトコルを使用することを示します。OSITPX は、BEA Tuxedo 8.0 以降のソフトウェアでのみサポートされます。

ドメイン・タイプは、DMTYPE ファイルで定義する必要があります。このファイルの場所は、Windows の場合は %TUXDIR%\udataobj\DMTYPE、UNIX の場合は $TUXDIR/udataobj/DMTYPE です。

ACCESSPOINTID (DOMAINID ともいう) = *string*[1..30]

リモート・ドメインへの接続を設定するときのセキュリティのため、このローカル・ドメイン・アクセス・ポイントに関連付けられているドメイン・ゲートウェイ・グループを識別するために使用します。TOPEND ドメイン・ゲートウェイの場合、この値は、BEA TOP END システムへの要求の BEA TOP END ユーザ ID として TEDG (GWTOPEND プロセスのローカル・インスタンス) によって使用される場合があります。

ACCESSPOINTID は、30 バイト以下で指定する必要があります。BEA TOP END ユーザ ID の長さは 1 ～ 12 文字です (最後のヌルを除く)。ASCII 文字 " " (32) から "~" (126) は、この文字列に使用できます。ただし、"/" (47) を除きます。ACCESSPOINTID 値に関連付けるパスワードは、dmadmin(1) サブコマンドの topendpasswd を使用して入力できます。

DM_LOCAL セクションのオプション TEDG パラメータ

AUDITLOG = *string*[1..256] (up to 78 bytes for BEA Tuxedo 8.0 or earlier)

このローカル・ドメイン・アクセス・ポイントに対する監査ログ・ファイルの名前を指定します。監査ログ機能は dmadmin(1) コマンドによって起動し、このローカル・ドメイン・アクセス・ポイントで行われるすべての動作を記録します。監査ログ機能がオンになっており、このパラメータが指定されていないと、環境変数 $APPDIR によって指定されたディレクトリまたは TUXCONFIG ファイルの MACHINES セクションの APPDIR パラメータで指定されるディレクトリに、*DMmmddyy*.LOG (ただし *mm*= 月、*dd*= 日、*yy*= 年) というファイルが作成されます。

BLOCKTIME = *numeric*

このローカル・ドメイン・アクセス・ポイントに対するブロッキング・コールの最大待ち時間を指定します。この値は、TUXCONFIG ファイルの RESOURCES セクションの SCANUNIT パラメータの乗数です。SCANUNIT * BLOCKTIME の値は、SCANUNIT 以上 32,768 秒未満でなければなりません。このパラメータを指定しないと、TUXCONFIG ファイルの RESOURCES セクションに指定された BLOCKTIME パラメータの値がデフォルトとして使用されます。ブロッキング・タイムアウト状態は、関連する要求が失敗したことを示します。

ドメイン間トランザクションでは、トランザクション期間が BLOCKTIME を超過すると ブロッキング・タイムアウト状態が生成されます。つまり、ドメイン間トランザクションでは、BLOCKTIME 値が TUXCONFIG ファイルの SERVICES セクションで指定された TRANTIME タイムアウト値未満の場合、またはトランザクションを開始するための tpbegin() 呼び出しで渡されたタイムアウト値未満の場合、トランザクションのタイムアウトは BLOCKTIME 値まで減らされます。一方、ドメイン内トランザクション (単一の BEA Tuxedo ドメイン内で処理さ

れるトランザクション) の場合は、TUXCONFIG ファイルの RESOURCES セクションで指定された BLOCKTIME 値は、ドメイン内トランザクションのタイムアウトに何の影響も与えません。

CONNECTION_POLICY = {ON_DEMAND | ON_STARTUP | INCOMING_ONLY}

このローカル・ドメイン・アクセス・ポイントに関連付けられているドメイン・ゲートウェイがリモート・ドメインへの接続を確立するときの条件を指定します。有効な値は、ON_DEMAND、ON_STARTUP、および INCOMING_ONLY です。このパラメータは、TOPEND または TDOMAIN タイプのドメイン・ゲートウェイにのみ適用されます。

接続ポリシーが ON_DEMAND の場合、クライアントがリモート・サービスを要求したとき、または dmadmin(1) connect コマンドが実行されたときにのみ、ドメイン・ゲートウェイはリモート・ドメインへの接続を試行します。CONNECTION_POLICY のデフォルトは ON_DEMAND です。リモート・ドメインへの接続で複数のネットワーク・アドレスが順に試行されるように設定したい場合、リモート・ドメインの複数のエントリ (複数のリモート・ドメイン・アクセス・ポイント) を DM_TOPEND セクションで指定できます。接続ポリシーが ON_DEMAND の場合、再接続は行われません。

接続ポリシーが ON_STARTUP の場合、ドメイン・ゲートウェイはゲートウェイ・サーバの初期化時にリモート・ドメインへの接続を試行します。リモート・ドメインへの接続で複数のネットワーク・アドレスが順に試行されるように設定したい場合、リモート・ドメインの複数のエントリ (複数のリモート・ドメイン・アクセス・ポイント) を DM_TOPEND セクションで指定できます。CONNECTION_POLICY を ON_STARTUP に設定した場合、リモート・ドメインへの接続が確立された場合にのみそのリモート・サービス (ドメイン・ゲートウェイによって宣言されたサービス) が宣言されますつまり、リモート・ドメインとの接続が確立されていないと、リモート・サービスは中断されます。デフォルトでは、失敗した接続が 60 秒おきに再試行されるよう設定されています。再接続の間隔は、RETRY_INTERVAL パラメータで変更できます。MAXRETRY パラメータも参照してください。

接続ポリシーが INCOMING_ONLY の場合、ドメイン・ゲートウェイは起動時にリモート・ドメインへの接続を試みません。このため、リモート・サービスは最初は中断されています。ドメイン・ゲートウェイは、リモート・ドメインからの接続を受信したときに利用可能になります。リモート・サービスは、ドメイン・ゲートウェイが接続を受信

したときか、dmadmin(1) connect コマンドで管理接続が確立されたときに宣言されます。リモート・ドメインへの管理接続でのみ複数のネットワーク・アドレスが順に試行されるように設定したい場合、リモート・ドメインの複数のエントリ (複数のリモート・ドメイン・アクセス・ポイント) を DM_TOPEND セクションで指定できます。接続ポリシーが INCOMING_ONLY の場合、再接続は行われません。

MAXRETRY = {*numeric* | MAXLONG}

このローカル・ドメイン・アクセス・ポイントに関連付けられているドメイン・ゲートウェイがリモート・ドメインへの接続を試行する回数を指定します。このパラメータは、TOPEND または TDOMAIN タイプのドメイン・ゲートウェイにのみ適用され、このローカル・ドメイン・アクセス・ポイントの CONNECTION_POLICY パラメータが ON_STARTUP に設定されている場合にのみ有効です。それ以外の接続ポリシーの場合、自動再試行は無効になります。

MAXRETRY の最小値は 0 で、最大値は MAXLONG (2147483647) です。MAXLONG (デフォルト) の場合、再接続処理が無限に繰り返されるか、または接続が確立されるまで繰り返されます。MAXRETRY=0 に設定すると、自動再接続は行われません。

RETRY_INTERVAL = *numeric*

このローカル・ドメイン・アクセス・ポイントに関連付けられているドメイン・ゲートウェイがリモート・ドメインへの接続を自動的に試行する間隔を秒単位で指定します。このパラメータは、TOPEND または TDOMAIN タイプのドメイン・ゲートウェイにのみ適用され、このローカル・ドメイン・アクセス・ポイントの CONNECTION_POLICY パラメータが ON_STARTUP に設定されている場合にのみ有効です。それ以外の接続ポリシーの場合、自動再試行は無効になります。

RETRY_INTERVAL の最小値は 0 、最大値は 2147483647 です。デフォルトは 60 です。MAXRETRY を 0 に設定すると、RETRY_INTERVAL は設定できません。

DMTLOGDEV = *string*[1..256] (BEA Tuxedo 8.0 以前では最大 78 バイト)

このローカル・ドメイン・アクセス・ポイントの Domains トランザクション・ログ (TLOG) を含む BEA Tuxedo ファイルシステムを指定します。TLOG は、BEA Tuxedo システムの VTOC テーブルとしてデバイスに格納されています。このパラメータを指定しない場合、このローカル・ドメイン・アクセス・ポイントに関連付けられているドメイン・

ゲートウェイ・グループは要求をトランザクション・モードで処理できません。同じマシンのローカル・ドメイン・アクセス・ポイント間で同じ BEA Tuxedo ファイルシステムを共有することはできますが、各ローカル・ドメイン・アクセス・ポイントは DMTLOGNAME パラメータで指定された名前のログ (DMTLOGDEV 内のテーブル) を保持する必要があります。

DMTLOGNAME = *string*[1..30]

このローカル・ドメイン・アクセス・ポイント用の TLOG の名前を指定します。複数のローカル・ドメイン・アクセス・ポイント間で同じ BEA Tuxedo ファイルシステム (DMTLOGDEV で指定) を共有する場合は、一意な名前を指定する必要があります。このパラメータを指定しない場合、デフォルトは DMTLOG となります。名前は 30 文字以内で指定する必要があります。

DMTLOGSIZE = *numeric*

このローカル・ドメイン・アクセス・ポイント用の TLOG のサイズをページ数で指定します。この値は、0 より大きく、BEA Tuxedo ファイルシステムで使用可能な容量より小さくする必要があります。このパラメータを指定しないと、デフォルトの 100 ページが設定されます。

MAXRAPTRAN (MAXRDTRAN ともいう) = *numeric*

このローカル・ドメイン・アクセス・ポイントのトランザクションに含めることのできるドメインの最大数を指定します。この値は 0 より大きく、32,768 未満でなければなりません。デフォルトは 16 です。

MAXTRAN = *numeric*

このローカル・ドメイン・アクセス・ポイントで同時に実行できるグローバル・トランザクションの最大数を指定します。この値は、0 以上で、TUXCONFIG ファイルの RESOURCES セクションに定義されている MAXGTT パラメータ以下でなければなりません。MAXTRAN を指定しない場合、デフォルトは MAXGTT です。

SECURITY = {NONE | CLEAR | SAFE | PRIVATE}

このローカル・ドメイン・アクセス・ポイント用に使用するアプリケーション・セキュリティの種類を指定します。TOPEND ドメイン・ゲートウェイ用の SECURITY パラメータの有効値は、現時点では NONE、CLEAR、SAFE、PRIVATE の 4 つです。NONE (デフォルト) の場合、セキュリティは使用されません。NONE 以外の値を指定した場合、BEA

TOP END 認証および認可が BEA TOP END システムとゲートウェイによって使用されます。また、CLEAR を指定した場合、ノード間メッセージに対する保護が不要になります。SAFE を指定した場合、Kerberos SAFE メッセージ・チェックサムを使用してメッセージを送信する必要があります。PRIVATE を指定した場合、DES の Kerberos 4 インプリメンテーションを使用してメッセージを暗号化する必要があります。SECURITY は、各 BEA TOP END ノードの nm_config (4T) ファイルの対応する BEA TOP END ノード・マネージャのコンフィギュレーションと一致する必要があります。これは、リモート BEA TOP END ノードとの接続の確立時に検証されます。

DM_REMOTE セクション

このセクション (DM_REMOTE_DOMAINS セクションともいう) では、1 つまたは複数のリモート・ドメイン・アクセス・ポイント識別子とそれらの特性を定義します。TOP END ドメイン・ゲートウェイ (TEDG) の定義について、このセクションではリモート BEA TOP END システム・ノード上のネットワーク・インターフェイス・コンポーネントへの接続を定義します。

DM_REMOTE セクションのエントリの形式は次のとおりです。

RemoteAccessPoint required_parameters [*optional_parameters*]

RemoteAccessPoint は、ローカル BEA Tuxedo アプリケーションにとって既知の各リモート・ドメインを識別するために選択するリモート・ドメイン・アクセス・ポイント識別子 (論理名) です。RemoteAccessPoint は、Domains コンフィギュレーションに含まれるローカルおよびリモート・ドメイン間で一意でなければなりません。

各リモート・ドメイン・アクセス・ポイントは、ローカル・ドメイン・アクセス・ポイントに関連付けられている TEDG が接続可能な BEA TOP END システム・ノードを定義します。ローカル・ドメインの TEDG は、ローカル・ドメインと同じ BEA TOP END システムの一部である TOPEND タイプのリモート・ドメインと通信します。BEA TOP END システムの名前は DM_TOPEND セクションで定義します。隣接ノードに対する BEA TOP END のルーティング・トポロジにより、BEA TOP END システムのサービスが複数のノードに分散されている場合があります。これらのサービスが属する各 BEA TOP END ノードへの接続を定義するため、TEDG ローカル・ドメイン・アクセス・ポイントには複数のリモート・ドメイン・アクセス・ポイントのエントリが必要になる場合があります。

DM_REMOTE セクションの必須 TEDG パラメータ

TYPE = *identifier*

このリモート・ドメイン・アクセス・ポイントに関連付けられるリモート・ドメインとの通信に必要なローカル・ドメイン・ゲートウェイのタイプを指定します。TYPE は、TOPEND、TDOMAIN、SNAX、OSITP、または OSITPX に設定できます。

TOPEND は、GWTOPEND プロセスのローカル・インスタンスがリモート BEA TOP END システムと通信することを示します。

TDOMAIN は、GWTDOMAIN プロセスのローカル・インスタンスがリモート BEA Tuxedo アプリケーションと通信することを示します。

SNAX は、GWSNAX プロセスのローカル・インスタンスが SNA プロトコルを介してリモート TP ドメインと通信することを示します。

OSITP は、GWOSITP プロセスのローカル・インスタンスが OSI TP 1.3 プロトコルを介してリモート TP ドメインと通信することを示します。

OSITPX は、GWOSITP プロセスのローカル・インスタンスが OSI TP 4.0 以降のプロトコルを介してリモート TP ドメインと通信することを示します。OSITPX は、BEA Tuxedo 8.0 以降のソフトウェアでのみサポートされます。

ACCESSPOINTID (DOMAINID ともいう) = *string*[1..30]

リモート・ドメインへの接続を設定するときのセキュリティのため、このリモート・ドメイン・アクセス・ポイントに関連付けられているリモート・ドメインを識別するために使用します。TOPEND ローカル・ドメイン・ゲートウェイの場合、この値は、このリモート・ドメイン・アクセス・ポイント接続で BEA TOP END システムから受信した要求の BEA Tuxedo ユーザ ID として TEDG (GWTOPEND プロセスのローカル・インスタンス) によって使用されます。ACCESSPOINTID は、ローカルおよびリモート・ドメイン・アクセス・ポイント間で一意でなければなりません。

ACCESSPOINTID は、30 バイト以下で指定する必要があります。文字列を指定する場合は、30 文字以内で指定する必要があります (最後のヌルを含む)。*string* の値は、一連の文字か、または 0x で始まる 16 進数です。

DM_REMOTE セクションのオプション TEDG パラメータ

PRIORITY_TYPE = {LOCAL_RELATIVE | LOCAL_ABSOLUTE | GLOBAL}

INPRIORITY = *numeric*

PRIORITY_TYPE パラメータと INPRIORITY パラメータでは、このリモート・ドメイン・アクセス・ポイントのメッセージの優先順位に関する処理を指定します。これらのパラメータは、BEA Tuxedo 8.0 以降のソフトウェアでサポートされます。

PRIORITY_TYPE パラメータの場合、LOCAL_RELATIVE と LOCAL_ABSOLUTE はすべてのリモート・ドメイン・タイプに対して有効ですが、GLOBAL は TDOMAIN のリモート・ドメイン・タイプに対してのみ有効です。PRIORITY_TYPE パラメータを設定しない場合、デフォルトは LOCAL_RELATIVE です。

PRIORITY_TYPE=LOCAL_RELATIVE は、tpsprio 呼び出しなどによるリモート・ドメイン・アクセス・ポイントからの要求の優先順位がローカル・ドメインによって使用されないことを意味します。代わりに、リモート・ドメイン・アクセス・ポイントから受信する要求の優先順位は INPRIORITY の値を基準に設定されます。この値は -99 (最低の優先順位) 〜 +99 (最高の優先順位) です。デフォルトは 0 です。INPRIORITY の設定によって、サービスのデフォルトの優先順位は、設定値の符号に応じて最大 100、最小 1 まで増減します。最高の優先順位は 100 です。リモート・ドメイン・アクセス・ポイントへの要求の場合、要求に関連付けられている優先順位も一緒にリモート・ドメイン・アクセス・ポイントに送信されます。

PRIORITY_TYPE=LOCAL_ABSOLUTE は、このリモート・ドメイン・アクセス・ポイントからの要求の優先順位がローカル・ドメインによって使用されないことを意味します。代わりに、リモート・ドメイン・アクセス・ポイントから受信する要求の優先順位は INPRIORITY の値を基準に設定されます。この値は 1 (最低の優先順位) 〜 100 (最高の優先順位) です。デフォルトは 50 です。INPRIORITY の設定によって、サービスのデフォルトの優先順位は、設定値の符号に応じて最大 100、最小 1 まで増減します。最高の優先順位は 100 です。リモート・ドメイン・アクセス・ポイントへの要求の場合、要求に関連付けられている優先順位も一緒にリモート・ドメイン・アクセス・ポイントに送信されます。

PRIORITY_TYPE=LOCAL_GLOBAL は、このリモート・ドメイン・アクセス・ポイントからの要求の優先順位がローカル・ドメインによって調整されることを意味します。リモート・ドメイン・アクセス・ポイントから受信する要求の優先順位は INPRIORITY の値を基準に調整されます。この値は -99 (最低の優先順位) 〜 +99 (最高の優先順位) です。デフォルトは 0 です。INPRIORITY を設定した場合、受信した要求に関連付けられている優先順位は INPRIORITY の値に加算され、その要求の優先順位の絶対値が設定されます。INPRIORITY を設定しない場合、受信する要求の優先順位がそのままローカル・ドメインによって使用

されます。リモート・ドメイン・アクセス・ポイントへの要求の場合、要求に関連付けられている優先順位も一緒にリモート・ドメイン・アクセス・ポイントに送信されます。

DM_EXPORT セクション

このセクション (DM_LOCAL_SERVICES セクションとうもいう) では、BEA Tuxedo サービスおよび /Q キュー・スペースを BEA TOP END システムが使用するために必要なマッピング情報を定義します。このセクションは、TOP END Domain Gateway で必要となります。

TEDG 用に記述された DMCONFIG ファイルの場合、DM_EXPORT セクションでは次のエントリを定義します。

- BEA Tuxedo と BEA TOP END 間の要求 / 応答および会話サービスのマッピングを定義するエントリ
- BEA Tuxedo と BEA TOP END 間のキュー・スペースのマッピングを定義するエントリ
- BEA Tuxedo と BEA TOP END 間のキュー名のマッピングを定義するエントリ

DM_EXPORT セクションのエントリの形式は次のいずれかです。

service [TYPE=SERVICE]*required_parameters* [*optional_parameters*]

qspace TYPE=QSPACE *required_parameters* [*optional_parameters*]

qname TYPE=QNAME *required_parameters* [*optional_parameters*]

service はエクスポートされた BEA Tuxedo サービスの名前、*qspace* はエクスポートされた BEA Tuxedo キュー・スペースの名前、*qname* は BEA Tuxedo キュー・スペース内で定義されたキューの名前です。それぞれの名前は 15 文字以内で指定します。エントリのタイプは、TYPE パラメータの設定値 (SERVICE、QSPACE、QNAME) によって決まります。

DM_EXPORT セクションの SERVICE エントリでは、TEDG によって BEA TOP END システムに対して宣言される BEA Tuxedo サービスを定義します。BEA TOP END システムに対して宣言される BEA Tuxedo サービスのエントリには、BEA Tuxedo サービス名から BEA TOP END サービス識別子 (製品、関数、ターゲット、修飾子) へのマッピングが指定されます。これらのサービス識別子は、BEA TOP END の tp_client_send(3T) および tp_client_signon(3T) ルーチン呼び出しで使用されます。

DM_EXPORT セクションの QSPACE エントリでは、BEA TOP END が RTQ キューとして使用できる BEA Tuxedo キュー・スペースを定義します (制限あり)。RTQ キューは、RTQ グループ名、RTQ キュー名、およびターゲット名を BEA TOP END サービス名として宣言することにより、BEA TOP END で使用可能になります。BEA TOP END ゲートウェイは、その RTQ キュー名に送信された tp_rtq_put(3T) 要求を、RTQ サーバと同じような方法で処理します。次に、各要求はこの QSPACE エントリで識別される BEA Tuxedo キュー・スペースにマッピングされます。メッセージのキュー処理には QSPACE エントリと QNAME エントリの両方が必要です。

DM_EXPORT セクションの QNAME エントリでは、RTQ を通して要求を BEA Tuxedo システムのキューに入れるための BEA TOP END サービス要求の BEA Tuxedo キュー名へのマッピングを定義します。QNAME エントリは、BEA TOP END システムにサービスとして宣言されません。QSPACE エントリと QNAME エントリは互いに独立しています。QSPACE および QNAME 識別子は、関連する BEA TOP END 識別子を tp_rtq_put(3T) ルーチン呼び出しで指定することにより、アプリケーションで任意に組み合わせて使用できます。組み合わせがローカル BEA Tuxedo ドメインに存在しないと、実行時エラーが発生します。。

QNAME エントリは、その製品、関数、ターゲット、および修飾子の組み合わせに関して、LACCESSPOINT パラメータで指定される特定のローカル・ドメイン・アクセス・ポイントで一意である必要があります。同じ組み合わせのエントリが複数設定された場合、TEDG は最初のエントリしか使用しません。

TE_PRODUCT パラメータを含む任意の SERVICE または QNAME エントリ、または TE_RTQGROUP パラメータを含む任意の QSPACE エントリは、そのエントリが LACCESSPOINT パラメータで特定のローカル・ドメイン・アクセス・ポイントに対して設定されていない場合、すべての TOPEND ローカル・ドメイン・アクセス・ポイントに適用できます。特定のローカル・ドメイン・アクセス・ポイントに対して設定されたエントリは、そのドメイン・ゲートウェイだけに適用されます。

SERVICE エントリと QSPACE エントリでは、BEA TOP END サービスとして宣言される BEA TOP END サービス識別子をコンフィギュレーションするので、特定のローカル・ドメイン・アクセス・ポイントに対するこれらの識別子は重複してはいけません。SERVICE エントリの場合、TE_PRODUCT、TE_FUNCTION、および TE_TARGET が宣言されます。QSPACE エントリの場合、

TE_RTQGROUP、TE_RTQNAME、および TE_TARGET が、製品、関数、およびターゲット識別子として宣言されます。このため、SERVICE エントリの製品、関数、およびターゲットが QSPACE エントリの RTQ グループ、RTQ キュー名、およびターゲットと一致する場合、TEDG は要求をルーティングできません。BEA TOP END システムの場合と同様に、ターゲットのデフォルト値は短縮されたノード名です。

DMCONFIG ファイルに複数の BEA TOP END システムのローカル・ドメイン・アクセス・ポイントを含む場合、または複数のタイプのドメイン・ゲートウェイを含む場合は、DM_EXPORT セクションでローカル・リソース・エントリに対する LACCESSPOINT パラメータを指定する必要があります。ローカル・ドメイン・アクセス・ポイントを指定せずに混在型のコンフィギュレーションを作成しないでください。ゲートウェイが正しく初期化しない場合があります。判断に迷う場合は、LACCESSPOINT を明示的に設定します。

次の表に、DM_EXPORT セクションの各 TEDG エントリの必須パラメータとオプション・パラメータを示します。

エントリ・タイプ	必須パラメータ	オプション・パラメータ
SERVICE	TE_PRODUCT, TE_FUNCTION	TYPE、LACCESSPOINT、TE_TARGET、TE_QUALIFIER、INBUFTYPE、OUTBUFTYPE、ACL、CONV
QSPACE	TYPE、TE_RTQGROUP、TE_RTQNAME	LACCESSPOINT、TE_TARGET
QNAME	TYPE、TE_PRODUCT、TE_FUNCTION	LACCESSPOINT、TE_TARGET、TE_QUALIFIER、INBUFTYPE、ACL

DM_EXPORT セクションの必須およびオプション TEDG パラメータ

LACCESSPOINT = *identifier*

このローカル・リソース (サービスまたはキュー・スペース) をエクスポートするローカル・ドメイン、またはこのキュー名が適用されるローカル・ドメイン・アクセス・ポイントを指定します。このパラメータを指定しない場合、ローカル・リソースは、DM_LOCAL セクションで定義した TOPEND タイプのすべてのローカル・ドメイン・アクセス・ポイントに適用できます。

TYPE = {SERVICE | QSPACE | QNAME}

このローカル・リソースが SERVICE、QSPACE、または QNAME エントリのどれであるかを指定します。値 SERVICE を指定した場合、このローカル・リソースのエントリは、BEA TOP END システムにエクスポートされるローカル BEA Tuxedo サービスに適用可能なマッピング・パラメータを定義します。値 QSPACE を指定した場合、このローカル・リソースのエントリは、BEA TOP END システムに RTQ キューとして提供されるローカル BEA Tuxedo キュー・スペースに適用可能なマッピング・パラメータを定義します。値 QNAME を指定した場合、このローカル・リソースのエントリは、RTQ を通して要求を BEA Tuxedo システムのキューに入れるための BEA TOP END サービス名

の BEA Tuxedo キュー名へのマッピングに適用可能なパラメータを定義します。デフォルトは "SERVICE" です。

TE_PRODUCT = *string*[1..32]
このローカル・リソースの BEA TOP END 製品名を指定します。長さは、最後のヌルを除いて 32 文字までです。有効な文字は、a-z、A-Z、0-9、_、-、および . (ピリオド) だけです。

TE_PRODUCT パラメータは、TYPE=SERVICE または QNAME の場合に指定する必要があります。このパラメータは、TYPE=QSPACE の場合は使用できません。

TE_FUNCTION = *string*[1..8]
このローカル・リソースの BEA TOP END 関数名を指定します。長さは、最後のヌルを除いて 8 文字までです。有効な文字は、a-z、A-Z、0-9、_、-、および . (ピリオド) だけです。

TE_FUNCTION パラメータは、TYPE=SERVICE または QNAME の場合に指定する必要があります。このパラメータは、TYPE=QSPACE の場合は使用できません。

TE_TARGET = *string*[1..8]
このローカル・リソースの BEA TOP END メッセージ・センシティブ・ルーティング (MSR) のターゲットを指定します。長さは、最後のヌルを除いて 8 文字までです。有効な文字は、a-z、A-Z、0-9、_、-、および . (ピリオド) だけです。SERVICE および QSPACE スペース・エントリの場合、最後の非空白文字としてアスタリスクを使用できます。TE_TARGET パラメータのデフォルト値は空白で、値が設定されていないことを示します SERVICE エントリと QSPACE エントリでは、このパラメータの値は、実行時にデフォルトで TEDG の短縮ノード名に変更されます。これらの値は、BEA TOP END システムがデフォルトのターゲット名に対して従う規則と一致します。

TE_TARGET パラメータは、TYPE=SERVICE、QSPACE、または QNAME の場合に指定できます。

TE_QUALIFIER = *numeric*
このローカル・リソースの BEA TOP END 関数修飾子を指定します。有効値は 0 から MAXLONG (2147483647) です。デフォルト値は 0 です。

TE_QUALIFIER パラメータは、TYPE=SERVICE または QNAME の場合に指定できます。このパラメータは、TYPE=QSPACE の場合は使用できません。

TE_RTQGROUP = *string*[1..32]
このローカル・リソースの BEA TOP END RTQ グループ名を指定します。グループ名の長さは、最後のヌルを除いて 32 文字までです。有効な文字は、a-z、A-Z、0-9、_、-、および . (ピリオド) だけです。

TE_RTQGROUP パラメータは、TYPE=QSPACE の場合に指定する必要があります。このパラメータは、TYPE=SERVICE または QNAME の場合は使用できません。

TE_RTQNAME = *string*[1..8]
このローカル・リソースの BEA TOP END RTQ キュー名を指定します。キュー名の長さは、最後のヌルを除いて 8 文字までです。有効な文字は、a-z、A-Z、0-9、_、-、および . (ピリオド) だけです。

TE_RTQNAME パラメータは、TYPE=QSPACE の場合に指定する必要があります。このパラメータは、TYPE=SERVICE または QNAME の場合は使用できません。

INBUFTYPE = *string*[0..513]
type[:*subtype*]— このローカル・リソースの入力バッファ・タイプと、オプションでサブタイプを指定します。BEA TOP END サービスおよびキュー名エントリでは、*type* の有効値は FML32、CARRAY、および X_OCTET です。

INBUFTYPE パラメータは、TYPE=SERVICE または QNAME の場合に指定できます。このパラメータは、TYPE=QSPACE の場合は使用できません。

OUTBUFTYPE = *string*[0..513]
type[:*subtype*]— このローカル・リソースの出力バッファ・タイプと、オプションでサブタイプを指定します。BEA TOP END サービス・エントリでは、*type* の有効値は FML32、CARRAY、および X_OCTET です。

OUTBUFTYPE パラメータは、TYPE=SERVICE の場合に指定できます。このパラメータは、TYPE=QSPACE または QNAME の場合は使用できません。

ACL = *identifier*

アクセス制御リスト (ACL) の名前を指定します。TEDG は、このリストを使用して BEA TOP END システムによるこのローカル・リソースへの要求を制限します。ACL は、DM_ACCESS_CONTROL セクションで定義します。

ACL パラメータは、TYPE=SERVICE または QNAME の場合に指定できます。このパラメータは、TYPE=QSPACE の場合は使用できません。

CONV = {Y | N}

このリモート・リソースが会話型サービスであるか (Y) 否か (N) を指定します。デフォルトは N です。CONV 属性は、TYPE=SERVICE に適用されます。TYPE=QSPACE または QNAME の場合は N に設定する必要があります。

DM_IMPORT セクション

このセクション (DM_REMOTE_SERVICES セクションともいう) では、BEA TOP END サービス、RTQ キュー、および RTQ を介してアクセスされるサービスを BEA Tuxedo アプリケーションで使用するために必要なマッピング情報を定義します。このセクションは、TOP END Domain Gateway で必要となります。

TEDG 用に記述された DMCONFIG ファイルの場合、DM_IMPORT セクションでは次のエントリを定義します。

- BEA Tuxedo と BEA TOP END 間の要求 / 応答および会話サービスのマッピングを定義するエントリ
- BEA Tuxedo と BEA TOP END 間のキュー・スペースのマッピングを定義するエントリ
- BEA Tuxedo と BEA TOP END 間のキュー名のマッピングを定義するエントリ

DM_IMPORT セクションのエントリの形式は次のいずれかです。

service [TYPE=SERVICE]*required_parameters* [*optional_parameters*]

qspace TYPE=QSPACE *required_parameters* [*optional_parameters*]

qname TYPE=QNAME *required_parameters* [*optional_parameters*]

service は BEA TOP END サービスに割り当てられた BEA Tuxedo サービスの名前、*qspace* は RTQ キューに割り当てられた BEA Tuxedo キュー・スペースの名前、*qname* は RTQ を介してアクセスされる BEA TOP END サービスに割り当てられた BEA Tuxedo キューの名前です。それぞれの名前は 15 文字以内で指定します。エントリのタイプは、TYPE パラメータの設定値 (SERVICE、QSPACE、QNAME) によって決まります。

DM_IMPORT セクションの SERVICE エントリでは、TEDG によってローカル BEA Tuxedo アプリケーションに対して宣言される BEA TOP END サービスを定義します。BEA Tuxedo アプリケーションに対して宣言される BEA TOP END サービスのエントリでは、BEA TOP END サービス識別子 (製品、関数、ターゲット、修飾子) から BEA Tuxedo サービス名へのマッピングが指定されます。これらのサービス名は、XATMI tpcall(3c) および tpacall(3c) 関数で使用されます。

DM_IMPORT セクションの QSPACE エントリでは、TEDG によってローカル BEA Tuxedo アプリケーションで BEA Tuxedo キュー・スペースと同じように使用できるようになる BEA TOP END RTQ キューを定義します (制限あり)。キュー・スペースは、キュー・スペース名を BEA Tuxedo サービス名として宣言することにより、BEA Tuxedo アプリケーションで使用できるようになります。TEDG は、そのキュー・スペース名に送信された tpenqueue(3c) 要求を、TMQUEUE(5) サーバと同じような方法で処理します。次に、各要求はこの qspace エントリで識別される RTQ キューにマップされます。メッセージのキュー処理には、QSPACE エントリと QNAME エントリの両方が必要です。

DM_IMPORT セクションの QNAME エントリでは、要求を BEA TOP END システムのキューに入れるための BEA Tuxedo キュー名の BEA TOP END サービス名へのマッピングを定義します。QNAME エントリは、ローカル BEA Tuxedo アプリケーションにサービスとして宣言されません。QSPACE と QNAME は独立したエントリです。QSPACE 識別子と QNAME 識別子は、tpenqueue(3c) 関数を使用してアプリケーションで任意に組み合わせて使用できます。

QNAME エントリは、キュー名識別子に関して、特定の LACCESSPOINT で一意でなければなりません。同じキュー名識別子のエントリが複数設定された場合、TEDG は最初のエントリしか使用しません。

TE_PRODUCT パラメータを含む任意の SERVICE または QNAME エントリ、または TE_RTQGROUP パラメータを含む任意の QSPACE エントリは、そのエントリが LACCESSPOINT パラメータで特定のローカル・ドメイン・アクセス・ポイントに対して設定されていない場合、TOPEND タイプの各ローカル・ドメイン・アクセス・ポイントに適用できます。特定の LACCESSPOINT に対してコンフィギュレーションされるエントリは、そのローカル・ドメイン・アクセス・ポイントのゲートウェイ (TEDG) に対してのみ適用できます。

SERVICE エントリと QSPACE エントリでは、BEA Tuxedo サービスとして宣言されるサービス識別子とキュー・スペース識別子をコンフィギュレーションするので、特定のローカル・ドメイン・アクセス・ポイントに対するこれらの識別子は重複してはいけません。ただし、ロード・バランシングに関しては、同じタイプおよび識別子で複数のエントリを使用できます。同じサービス識別子のすべてのエントリには、同じ CONV パラメータ値を指定する必要があります。

DMCONFIG ファイルに複数の BEA TOP END システムのローカル・ドメイン・アクセス・ポイントを含む場合、または複数のタイプのドメイン・ゲートウェイを含む場合は、DM_IMPORT セクションでリモート・リソース・エントリに対する LACCESSPOINT パラメータを指定する必要があります。リモート・ドメイン・アクセス・ポイントをリモート・サービス・エントリ、または参照ルーティング・エントリで指定する場合、その値はローカル・ドメイン・アクセス・ポイントのタイプ (TOPEND) と TP_SYSTEM の値に一致する必要があります。ローカル・ドメイン・アクセス・ポイントを指定せずに、あるいはタイプまたは TP_SYSTEM が混在するリモート・ドメイン・アクセス・ポイントを参照して、混在型のコンフィギュレーションを作成してはなりません。判断に迷う場合は、LACCESSPOINT および RACCESSPOINT パラメータを明示的に設定します。リモート・ドメインに対するワイルドカードの指定は、単一のゲートウェイ・タイプを定義する場合に限られます。

次の表に、DM_IMPORT セクションの各 TEDG エントリの必須パラメータとオプション・パラメータを示します。

エントリ・タイプ	必須パラメータ	オプション・パラメータ
SERVICE	TE_PRODUCT, TE_FUNCTION	TYPE、RACCESSPOINT、LACCESSPOINT、TE_TARGET、TE_QUALIFIER、INBUFTYPE、OUTBUFTYPE、CONV、LOAD、ROUTING
QSPACE	TYPE, TE_RTQGROUP, TE_RTQNAME	RACCESSPOINT、LACCESSPOINT、TE_TARGET、LOAD
QNAME	TYPE、TE_PRODUCT、TE_FUNCTION	LACCESSPOINT、TE_TARGET、TE_QUALIFIER、INBUFTYPE、LOAD、ROUTING

DM_IMPORT セクションの必須およびオプション TEDG パラメータ

RACCESSPOINT (RDOM ともいう) =
identifier1[,*identifier2*][,*identifier3*]

このリソース (サービスまたは RTQ キュー) をインポートするためのリモート・ドメイン・アクセス・ポイントを指定します。リモート・ドメイン・アクセス・ポイントは TOPEND タイプで、このエントリが適用されるローカル・ドメイン・アクセス・ポイント (LACCESSPOINT パラメータで指定する) と同じ TP_SYSTEM の一部でなければなりません。

RACCESSPOINT を指定しない場合、TOPEND ローカル・ドメイン・アクセス・ポイントに関連付けられているドメイン・ゲートウェイは、TP_SYSTEM の値がローカル・ドメインと同じである TOPEND リモート・ドメイン・アクセス・ポイントはすべてこのリソースを提供するものと見なします。

identifier2 および *identifier3* 引数を指定して代替リモート・ドメイン・アクセス・ポイントをコンフィギュレーションする場合、DM_LOCAL セクションの CONNECTION_POLICY パラメータの値として ON_STARTUP を指定する必要があります。*identifier2* を設定した場合、それはフェイルオーバー用に使用されます。*identifier1* に関連付けられているリモート・ドメインが使用できなくなった場合、*identifier2* に関連付けられているリモート・ドメインが使用されます。同様に、*identifier3* を設定した場合、それはフェイルオーバー用に使用されます。*identifier1* と *identifier2* に関連付けられているリモート・ドメインが使用できなくなった場合、*identifier3* に関連付けられているリモート・ドメインが使用されます。リモート・サービスでは、ロード・バランシング (複数のリモート・サービス・エントリ) と Domains フェイルオーバー (このパラメータで指定される代替リモート・ドメイン・アクセス・ポイントを使用) の両方を使用できます。

LACCESSPOINT (LDOM ともいう) = *identifier*

このリモート・リソース (サービスまたは RTQ スペース) をインポートするローカル・ドメイン、またはこのキュー名が適用されるローカル・ドメイン・アクセス・ポイントを指定します。このパラメータを指定しない場合、リモート・リソースは、DM_LOCAL セクションで定義した TOPEND タイプのすべてのローカル・ドメイン・アクセス・ポイントに適用できます。

TYPE = {SERVICE | QSPACE | QNAME}

このリモート・リソースが SERVICE、QSPACE、または QNAME エントリのどれであるかを指定します。値 SERVICE を指定した場合、このリモート・リソースのエントリは、BEA TOP END サービスをローカル BEA Tuxedo サービスとして使用できるようにするためのマッピング・パラメータを定義します。値 QSPACE を指定した場合、このリモート・リソースのエントリは、BEA TOP END RTQ キューをローカル BEA Tuxedo キュー・スペースとして使用できるようにするためのマッピング・パラメータを定義します。値 QNAME を指定した場合、このリモート・リソースのエントリは、/Q を通して要求を BEA TOP END システムのキューに入れるための BEA Tuxedo キュー名の BEA TOP END サービス名へのマッピングに必要なパラメータを定義します。デフォルトは SERVICE です。

TE_PRODUCT = *string*[1..32]

このリモート・リソースの BEA TOP END 製品名を指定します。長さは、最後のヌルを除いて 32 文字までです。有効な文字は、a-z、A-Z、0-9、_、-、および . (ピリオド) だけです。

TE_PRODUCT パラメータは、TYPE=SERVICE または QNAME の場合に指定する必要があります。このパラメータは、TYPE=QSPACE の場合は使用できません。

TE_FUNCTION = *string*[1..8]

このリモート・リソースの BEA TOP END 関数名を指定します。長さは、最後のヌルを除いて 8 文字までです。有効な文字は、a-z、A-Z、0-9、_、-、および . (ピリオド) だけです。

TE_FUNCTION パラメータは、TYPE=SERVICE または QNAME の場合に指定する必要があります。このパラメータは、TYPE=QSPACE の場合は使用できません。

TE_TARGET = *string*[1..8]

このリモート・リソースの BEA TOP END メッセージ・センシティブ・ルーティング (MSR) のターゲットを指定します。長さは、最後のヌルを除いて 8 文字までです。有効な文字は、a-z、A-Z、0-9、_、-、および . (ピリオド) だけです。デフォルトはスペースです。

TE_TARGET パラメータは、TYPE=SERVICE、QSPACE、または QNAME の場合に指定できます。

TE_QUALIFIER = *numeric*

このリモート・リソースの BEA TOP END 関数修飾子を指定します。有効値は 0 から MAXLONG (2147483647) です。デフォルト値は 0 です。

TE_QUALIFIER パラメータは、TYPE=SERVICE または QNAME の場合に指定できます。このパラメータは、TYPE=QSPACE の場合は使用できません。

TE_RTQGROUP = *string*[1..32]

このリモート・リソースの BEA TOP END RTQ グループ名を指定します。グループ名の長さは、最後のヌルを除いて 32 文字までです。有効な文字は、a-z、A-Z、0-9、_、-、および . (ピリオド) だけです。

TE_RTQGROUP パラメータは、TYPE=QSPACE の場合に指定する必要があります。このパラメータは、TYPE=SERVICE または QNAME の場合は使用できません。

TE_RTQNAME = *string*[1..8]
このリモート・リソースの BEA TOP END RTQ キュー名を指定します。キュー名の長さは、最後のヌルを除いて 8 文字までです。有効な文字は、a-z、A-Z、0-9、_、-、および . (ピリオド) だけです。

TE_RTQNAME パラメータは、TYPE=QSPACE の場合に指定する必要があります。このパラメータは、TYPE=SERVICE または QNAME の場合は使用できません。

INBUFTYPE = *string*[1..256]
type[:*subtype*]— このリモート・リソースの入力バッファ・タイプと、オプションでサブタイプを指定します。BEA TOP END サービスおよびキュー名エントリでは、*type* の有効値は FML32、CARRAY、および X_OCTET です。

INBUFTYPE パラメータは、TYPE=SERVICE または QNAME の場合に指定できます。このパラメータは、TYPE=QSPACE の場合は使用できません。

OUTBUFTYPE = *string*[1..256]
type[:*subtype*]— このリモート・リソースの出力バッファ・タイプと、オプションでサブタイプを指定します。BEA TOP END サービス・エントリでは、*type* の有効値は FML32、CARRAY、および X_OCTET です。

OUTBUFTYPE パラメータは、TYPE=SERVICE の場合に指定できます。このパラメータは、TYPE=QSPACE または QNAME の場合は使用できません。

CONV = {Y | N}
このリモート・リソースが会話型サービスであるか (Y) 否か (N) を指定します。デフォルトは N です。CONV 属性は、TYPE=SERVICE に適用されます。TYPE=QSPACE または QNAME の場合は N に設定する必要があります。

BEA TOP END サーバ・アプリケーションで擬似会話を管理する場合、このリモート・サービスの CONV パラメータは Y に設定する必要があります。アプリケーション・コンテキストは維持される場合と維

持されない場合があります。CONV を Y に設定した場合、会話の XATMI 関数 (tpconnect、tpsend、および tpdiscon) を使用する必要があります。CONV を N に設定した場合、このサービスでは要求 / 応答 XATMI 関数 (tpcall、tpacall) を使用する必要があります。

LOAD = *numeric*

このリモート・リソースのサービス負荷を指定します。この値は、1 以上 32767 以下でなければなりません。デフォルトは 50 です。ロード・バランシングのためにインターフェイス負荷が使用されます。つまり、キュー登録の負荷が高いキューほど、新しい要求の処理に使用されません。

The LOAD パラメータは、TYPE=SERVICE、QSPACE、または QNAME の場合に指定できます。

ROUTING = *identifier*

このリモート・リソースのデータ依存型ルーティングを行うために使用するルーティング基準の名前を指定します。複数のリモート・ドメイン・アクセス・ポイントが同じリソースを提供するとき、このパラメータを指定してあれば、ローカル・ドメイン・アクセス・ポイントはデータ依存型ルーティングを実行できます。このパラメータを指定しないと、このリソースに対してデータ依存型ルーティングは使用されません。

identifier は、DM_ROUTING セクションで定義された *ROUTING_CRITERIA_NAME* です。*identifier* の値は 15 文字以下でなければなりません。同じサービス名またはキュー・スペース名の複数のエントリが異なるリモート・ドメイン・アクセス・ポイントに含まれている場合 (RACCESSPOINT パラメータで指定)、ROUTING パラメータの値はこれらのエントリすべてに対して同じにする必要があります。さらに、参照されるルーティング基準でコンフィギュレーションされたリモート・ドメインは、このリモート・リソース・エントリが適用されるローカル・ドメイン・アクセス・ポイントと同じ TP_SYSTEM の一部でなければなりません。

ROUTING パラメータは、TYPE=SERVICE または QNAME の場合に指定できます。このパラメータは、TYPE=QSPACE の場合は使用できません。

DM_RESOURCES

このオプション・セクションでは、グローバル Domains コンフィギュレーション情報、特にユーザ指定のコンフィギュレーション・バージョン文字列を定義します。このフィールドはソフトウェアによってチェックされません。

DM_RESOURCES セクションのパラメータは次の 1 つだけです。

VERSION = *string*

string は、ユーザが現在の DMCONFIG コンフィギュレーション・ファイルのバージョン番号を入力するためのフィールドです。

DM_ROUTING セクション

このセクションでは、型付きバッファ FML32 を使用して、同じリソースを提供する複数のリモート TOP END システムの 1 つにローカル要求をデータ依存型ルーティングするための情報を指定します。この説明は、TOPEND タイプのドメイン・ゲートウェイにのみ適用されます。

DM_ROUTING セクションのエントリの形式は次のとおりです。

ROUTING_CRITERIA_NAME required_parameters

ROUTING_CRITERIA_NAME は、DM_IMPORT セクションの特定のリソース・エントリの ROUTING パラメータに割り当てられる *identifier* の名前です。*ROUTING_CRITERIA_NAME* は 15 文字以下でなければなりません。

DM_ROUTING セクションの必須 TEDG パラメータ

FIELD = *identifier*

ルーティング・フィールドの名前を 30 文字以内で指定します。このパラメータの値は、FML フィールド・テーブル (FML32 バッファの場合) で識別されたフィールド名であると想定されます。FML フィールド・テーブルを検索するには、FLDTBLDIR32 および FIELDTBLS32 環境変数を使用します。FML32 バッファ内のフィールドがルーティングに使用される場合は、8191 以下の数値をフィールド番号として指定します。

RANGES = "*string*[1..4096]"

ルーティング・フィールドの範囲および関連するリモート・ドメイン・アクセス・ポイント名を指定します。*string* は二重引用符で囲みます。*string* はカンマで区切ったペアのリストで、各ペアはコロン (:) で区切られた範囲とリモート・ドメイン・アクセス・ポイントで構成されます (
RANGES = "MIN-1000:b01,1001-3000:b02,*:b03" など)。

範囲は、単一の値 (符号付き数値または一重引用符で囲んだ文字列)、または *lower* - *upper* の形式で表します。*lower* と *upper* は、いずれも符号付き数値または一重引用符で囲んだ文字列です。*lower* の値は、*upper* の値より小さくなければなりません。

文字列値に一重引用符を埋め込むには (例 : `O'Brien`)、一重引用符の前にバックスラッシュを 2 つ入れます (例 : `O\\'Brien`)。

関連する `FIELD` のデータ型の最小値を示すには、`MIN` を使用します。文字列と carray の最小値にはヌル文字列を指定します。文字フィールドの最小値には 0 を指定します。数値の場合、これはフィールドに格納できる最小値です。

関連する `FIELD` のデータ型の最大値を示すには、`MAX` を使用します。文字列と carray の最大値には、8 進数値の 255 文字の無限文字列を指定します。文字フィールドの最大値には、単一の 8 進数値の 255 文字を指定します。数値の場合は、数値としてフィールドに格納できる最大値です。したがって、文字列値 "`MIN - -5`" は -5 以下のすべての数値を指し、"`6 - MAX`" は、6 以上のすべての数値を指すことになります。範囲内のメタキャラクタ `*` (ワイルドカード) は、既にエントリとして指定した範囲では使用されなかった任意の値を示します。各エントリでは 1 つのワイルドカードによる範囲指定だけが可能です。`*` は最後に指定します。続けて範囲を指定すると無視されます。

ルーティング・フィールドには、FML でサポートされている任意のデータ型を指定できます。数値ルーティング・フィールドには数値で範囲を指定し、文字列ルーティング・フィールドには、文字列で範囲を指定する必要があります。文字列で範囲を設定する場合は、文字列、carray、および文字フィールド型の値を一重引用符で囲みます。先頭に符号を付けることはできません。short 型および long 型の整数値は数字の文字列であり、オプションで先頭に正の符号または負の符号を付けることができます。浮動小数点数は、C コンパイラまたは `atof(3)` で受け入れられる形式で指定します。つまり、符号 (オプション)、数字の文字列 (オプションで小数点を追加)、`e` または `E` (オプション)、符号またはスペース (オプション)、整数という形式で指定します。

フィールド値が範囲と一致する場合、関連付けられているリモート・ドメイン・アクセス・ポイントは、要求がルーティングされるリモート・ドメインを示します。リモート・ドメイン・アクセス・ポイントの値に "`*`" を指定すると、ゲートウェイ・グループが認識する任意のリモート・ドメインに要求が送られます。

BUFTYPE = "*type1*[:*subtype1*[,*subtype2* . . .]][;*type2*[:*subtype3*[, . . .]]] . . ."

このルーティング・エントリが有効なデータ・バッファのタイプとサブタイプを示すリストです。TOP END Domain Gateway の場合、タイプは FML32 に制限されます。FML32 ではサブタイプを指定できません。このパラメータは必須です。

フィールド値が設定されていないか (FML32 バッファの場合)、または特定の範囲と一致しておらず、ワイルドカードの範囲が指定されていない場合、リモート・リソースの実行を要求したアプリケーション・プロセスに対してエラーが返されます。

DM_ACCESS_CONTROL セクション

このセクションでは、1 つまたは複数のアクセス制御リスト (ACL) の名前を指定し、各 ACL 名に 1 つまたは複数のリモート・ドメイン・アクセス・ポイントを関連付けます。ACL=*ACL_NAME* に設定すると、DM_EXPORT セクションの ACL パラメータを使用して、特定のローカル・ドメイン・アクセス・ポイントから *ACL_NAME* に関連付けられているリモート・ドメイン・アクセス・ポイントにエクスポートされるローカル・リソースへのアクセスを制限できます。

DM_ACCESS_CONTROL セクションのエントリの形式は次のとおりです。

ACL_NAME required_parameters

ACL_NAME はアクセス制御リストを指定するための識別子です。長さは 15 文字までです。

以下は、必須パラメータです。

ACLIST = *identifier*[,*identifier*]

ACLIST には、1 つまたは複数のリモート・ドメイン・アクセス・ポイント名をカンマで区切って指定します。ワイルドカード文字 (*) を使用すると、DM_REMOTE セクションで定義したすべてのリモート・ドメイン・アクセス・ポイントが特定のローカル・ドメイン・アクセス・ポイントからエクスポートされる特定のローカル・リソースにアクセスできます。

DM_TOPEND セクション

このセクションでは、BEA TOP END Domain Gateway のネットワーク固有の情報を定義します。リモート・ドメインからローカル・リソースへの要求がローカル・ドメイン・アクセス・ポイントで受け付けられる場合、DM_TOPEND セクションには、ローカル・ドメインごとに 1 つのエントリがなければなりません。また、ローカル・ドメインからリモート・リソースへの要求がリモート・ドメイン・アクセス・ポイントで受け付けられる場合、そのアクセス・ポイントごとに 1 つのエントリがなければなりません。

DM_TOPEND セクションのエントリの形式は次のとおりです。

AccessPoint required_parameters [*optional_parameters*]

AccessPoint は、ローカル・ドメイン・アクセス・ポイントまたはリモート・ドメイン・アクセス・ポイントの識別子の値です。*AccessPoint* 識別子は、DM_LOCAL セクションに定義されているローカル・ドメイン・アクセス・ポイントか、または DM_REMOTE セクションに定義されているリモート・ドメイン・アクセス・ポイントと一致する必要があります。

ローカルおよびリモート・ドメイン・アクセス・ポイント、およびそれらのネットワーク・アドレスは、実行時に特定の TP_SYSTEM 名に対して、BEA TOP END ノードへの BEA TOP END ゲートウェイ接続が 1 つだけアクティブになるようにコンフィギュレーションする必要があります。BEA TOP END のネットワーク・インターフェイス・プロトコルは、このような複数ゲートウェイ接続をサポートしていません。複数の接続をアクティブにしようとすると、TEDG または BEA TOP END ノードで実行時エラーが発生し、受け付けられる接続は 1 つだけです。ネットワーク・アドレスは変更可能であるため、このようなコンフィギュレーションを DMCONFIG ファイルで完全に検証することはできません。

DM_TOPEND セクションの必須パラメータ

NWADDR = *string*[1..78]

ローカルまたはリモート・ドメイン・アクセス・ポイントに関連付けるネットワーク・アドレスを指定します。ローカル・ドメイン・アクセス・ポイントの場合、このパラメータには BEA TOP END システムからの接続指示を受け付けるためのアドレスを指定します。接続指示受け付けアドレスは、BEA TOP END システムのネットワーク・イン

ターフェイス・コンポーネントから BEA Tuxedo アプリケーションへの通信手段となります。リモート・ドメイン・アクセス・ポイント・エントリの場合、このパラメータにはリモート・ドメイン・アクセス・ポイントに関連付けられている BEA TOP END システムに接続するときに使用するアドレスを指定します。このパラメータの値は、すべての DM_TOPEND エントリ間で一意でなければなりません。

string の形式が "0x*hex-digits*" または "\\x*hex-digits*" の場合、偶数の有効な 16 進数桁を含める必要があります。これらの形式は、TCP/IP アドレスを含む文字配列に内部変換されます。*string* の値は次のいずれかの形式で指定します。

"//*hostname*:*port_number*"
"//*#.#.#.#*:*port_number*"

最初の形式では、gethostbyname(3c) を介してアクセスされたローカル設定の名前解決機能を使ってアドレスが結合されるときに、*hostname* は TCP/IP ホスト・アドレスに解決されます。*#.#.#.#* はドットで区切った 10 進数の形式で、各 *#* は 0 から 255 までの 10 進数です。

Port_number は、0 ～ 65535 の 10 進数です。

注記　一部のポート番号は、お使いのシステムで使用されるトランスポート・プロトコル (TCP/IP など) のために予約されている場合があります。予約されているポート番号を確認するには、トランスポート・プロトコルのマニュアルを調べてください。

管理者が、BEA TOP END ネットワーク・インターフェイスで使用されるような接続指示受け付けアドレス (ローカル・ドメイン・アクセス・ポイント) として INADDR_ANY を設定する場合、"//0.0.0.0:*port_number*" という形式で指定します。アドレスがこの形式で指定されると、TEDG (GWTOPEND) プロセスは、ローカル BEA Tuxedo アプリケーションが実行されているマシン上のすべての有効な IP アドレスの *port_number* で接続指示を受け付けることができます。

NWADDR パラメータのホスト・アドレス部分を指定する際は注意が必要です。BEA TOP END の NI は、TEDG から発行された接続要求を受け取ると、TEDG のネットワーク・アドレスを名前に解決します。解

決された名前は、TEDG で定義済みのホスト名と一致している必要があります。一致しない場合 (大文字と小文字の区別も含む)、NI 接続は失敗します。このような失敗は、GWTOPEND ログ・ファイルにもリモートの BEA TOP END NI ログ・ファイルにも記録されない場合があります。基本的に、ホスト名の定義を DMCONFIG ファイル、TOP END NI コンフィギュレーション・ファイル、TOP END ノードマップ・ファイル、TOP END tp_alias ファイル、およびローカル名の名前解決機能で一致させます。NI の名前解決については、『BEA TOP END Programmer's Reference Manual』の tp_alias(4T) リファレンス・ページを参照してください。

TP_SYSTEM = *string* [1..8]

ローカルまたはリモート・ドメイン・アクセス・ポイントに関連付ける BEA TOP END システムを指定します。このパラメータには、BEA TOP END システム名に対応する文字列を指定します。BEA TOP END システム名の長さは 1 ～ 8 文字です (最後のヌルを除く)。ASCII 文字 " " (32) から "~" (126) は、この文字列に使用できます。ただし、"/" (47) を除きます。文字列の値は、BEA TOP END システムの nm_script (4T) ファイルで定義される TP_SYSTEM 環境変数の値と一致する必要があります。

DM_TOPEND セクションのオプション・パラメータ

NWDEVICE = *string*[1..78]

このローカルまたはリモート・ドメイン・アクセス・ポイントのネットワーク・アドレスにバインディングするときに使用するネットワーク・デバイスを指定します。ローカル・ドメイン・アクセス・ポイント・エントリの場合、この属性は接続指示を受け付けるために使用するデバイスを指定します。リモート・ドメイン・アクセス・ポイントの場合、リモート・ドメイン・アクセス・ポイントに接続するために使用するデバイスを指定します。

NWDEVICE パラメータの指定は必須ではありません。以前のバージョンでは、TLI 対応のネットワーキング機能に対しては、デバイス名に絶対パス名を指定する必要があります。

DM_TOPEND セクション内の同じアクセス・ポイントに対する複数のエントリ

この DM_TOPEND エントリが (DM_LOCAL セクションで指定された) ローカル・ドメイン・アクセス・ポイントの場合、その NWADDR は接続指示を受け付けるためのネットワーク・アドレスです。DM_TOPEND セクションでは、ローカル・ドメイン・アクセス・ポイントに関連付けられたエントリを複数回指定して、ローカル・ドメイン・アクセス・ポイントに関連付けられているリソースを BEA Tuxedo ドメイン内の別のマシンに移行できます。

リモート・ドメイン・アクセス・ポイント (DM_REMOTE セクションで指定) に関連付けられたエントリも、DM_TOPEND セクションで複数回指定できます。最初のエントリは、一次アドレスと見なされます。つまり、その NWADDR は、リモート・ドメイン・アクセス・ポイントへの接続が最初に試行されるときのネットワーク・アドレスです。2 番目のエントリは、二次アドレスと見なされます。つまり、その NWADDR は、一次アドレスを使用して接続を確立できないときの次のネットワーク・アドレスです。後続のエントリのネットワーク・アドレスは、前のすべてのエントリのネットワーク・アドレスで接続を確立できない場合に使用されます。コンフィギュレーションされているすべてのネットワーク・アドレスへの接続が失敗すると、接続の試行は異常終了します。リモート・ドメイン・アクセス・ポイントに関連付けるエントリは、何回でも指定できます。ネットワーク・アドレスのコンフィギュレーションが多すぎる場合、または使用できないアドレスを設定した場合、性能が低下することがあります。

この DM_TOPEND エントリがリモート・ドメイン・アクセス・ポイントの別のオカレンスの場合、このエントリは、一次エントリ (および前のすべての二次エントリ) の NWADDR でネットワーク接続を確立できない場合にのみ使用されます。すべての二次エントリでは、

- TP_SYSTEM の値は、一次リモート・ゲートウェイ・エントリの TP_SYSTEM の値と一致する必要があります。
- エントリは、一次リモート・ドメインの接続先と同じノードへの代替ネットワーク接続を参照する必要があります。

二次リモート・ゲートウェイ定義は、TOP END Domain Gateway では使用しないでください。

注記　DM_TOPEND セクションのローカルまたはリモート・ドメイン・アクセス・ポイントの複数のエントリの場合、NWADDR パラメータの複数のインスタンスだけが Domains ソフトウェアによって読み取られます。他のパラメータの複数のインスタンスの場合、パラメータの最初のインスタンスだけが Domains ソフトウェアによって読み取られ、それ以外のインスタンスはすべて無視されます。

DMCONFIG for GWTOPEND(5) に関する追加情報

ファイル　BDMCONFIG 環境変数は、BDMCONFIG コンフィギュレーション・ファイルを検索するために使用します。

使用例　次の Domains コンフィギュレーション・ファイルの例は、コア BEA Tuxedo DMCONFIG(5) リファレンス・ページの使用例 1 に基づいています。この例は、BEA TOP END システムとの単一接続を持つ TOP END Domain Gateway を含むよう拡張されています。このシナリオでは、BEA TOP END システムは、BEA Tuxedo アプリケーションのユーザが必要とするサービスを提供するバンキング・アプリケーションも実行しています。逆に、特定の BEA Tuxedo サービスについては、変更されたクライアント・プログラムで使用するために BEA TOP END システムで利用できるようにすることが必要です。また、ここでは単純なキューの例も含まれています。

以下に、このコンフィギュレーションの DMCONFIG ファイルを示します。元のコア DMCONFIG(5) ファイルの使用例からの変更箇所は太字で示してあります。.

```
# Tuxedo Domains Configuration File for the Central Bank
#
#
*DM_LOCAL
#
DEFAULT: SECURITY = NONE


c01  GWGRP = bankg1
     TYPE = TDOMAIN
     ACCESSPOINTID = "BA.CENTRAL01"
     DMTLOGDEV = "/usr/apps/bank/DMTLOG"
     DMTLOGNAME = "DMTLG_C01"

c02  GWGRP = bankg2
     TYPE = OSITP
     ACCESSPOINTID = "BA.CENTRAL02"
     DMTLOGDEV = "/usr/apps/bank/DMTLOG"
     DMTLOGNAME = "DMTLG_C02"

c03  GWGRP = bankg3
     TYPE = TOPEND
     ACCESSPOINTID = "CENTRALBKGW"
     DMTLOGDEV = "/usr/apps/bank/DMTLOG"
     DMTLOGNAME = "DMTLG_C03"
```

```
     SECURITY = CLEAR

#
*DM_REMOTE
#
b01  TYPE = TDOMAIN
     ACCESSPOINTID = "BA.BANK01"

b02  TYPE = TDOMAIN
     ACCESSPOINTID = "BA.BANK02"

b03  TYPE = TDOMAIN
     ACCESSPOINTID = "BA.BANK03"

b04  TYPE = OSITP
     ACCESSPOINTID = "BA.BANK04"

b05  TYPE = TOPEND
     ACCESSPOINTID = "BANK05"

*DM_TDOMAIN
#
# ローカル・ネットワーク・アドレス
c01  NWADDR = "//newyork.acme.com:65432"  NWDEVICE ="/dev/tcp"

# リモート・ネットワーク・アドレス
b01  NWADDR = "//192.11.109.5:1025" NWDEVICE = "/dev/tcp"
b02  NWADDR = "//dallas.acme.com:65432" NWDEVICE = "/dev/tcp"
b03  NWADDR = "//192.11.109.156:4244" NWDEVICE = "/dev/tcp"

*DM_OSITP
#
c02  APT = "BA.CENTRAL02"
     AEQ = "TUXEDO.R.4.2.1"
     AET = "{1.3.15.0.3},{1}"
     ACN = "XATMI"
b04  APT = "BA.BANK04"
     AEQ = "TUXEDO.R.4.2.1"
     AET = "{1.3.15.0.4},{1}"
     ACN = "XATMI"

*DM_TOPEND
#
# ローカル・ネットワーク・アドレス
c03  NWADDR = "//newyork.acme.com:65434"
     TP_SYSTEM = "BANKSYS"
# リモート・ネットワーク・アドレス
b05  NWADDR = "//sandiego.acme.com:65434"
```

```
     TP_SYSTEM = "BANKSYS"

*DM_EXPORT
#
#TOP END で利用不可、マッピングなし
open_act   ACL = branch  LACCESSPOINT = c01
close_act  ACL = branch  LACCESSPOINT = c01


credit LACCESSPOINT = c01
debit  LACCESSPOINT = c01
loan   LACCESSPOINT = c02  ACL = loans

# TOP END およびほかのドメインにエクスポートされるサービス
balance TYPE=SERVICE  TE_PRODUCT="TUX"  TE_FUNCTION="BALANCE"  LACCESSPOINT=c03

# TOP END で使用可能なキュー
qspace  TYPE=QSPACE  TE_RTQGROUP="TUXQUEUE"  TE_RTQNAME="TUXQ"  LACCESSPOINT=c03
qname   TYPE=QNAME   TE_PRODUCT="TUX"  TE_FUNCTION="QSERV"  LACCESSPOINT=c03

*DM_IMPORT
#
tlr_add  LACCESSPOINT = c01  ROUTING = ACCOUNT
tlr_bal  LACCESSPOINT = c01  ROUTING = ACCOUNT
tlr_add  RACCESSPOINT = b04  LACCESSPOINT = c02  RNAME ="TPSU002"
tlr_bal  RACCESSPOINT = b04  LACCESSPOINT = c02  RNAME ="TPSU003"

#
# BEA Tuxedo で使用可能な新しい New TOP END サービス
DEFAULT:        LACCESSPOINT = c03  RACCESSPOINT = b05
                TYPE = SERVICE  TE_PRODUCT = "EBANK"
te_start        TE_FUNCTION = "START"
te_end          TE_FUNCTION = "END"
te_login        TE_FUNCTION = "LOGIN"
te_listacct     TE_FUNCTION = "LISTACCT"
te_getpayees    TE_FUNCTION = "GETPAYES"
te_elecpay      TE_FUNCTION = "ELECPAY"
te_bal          TE_FUNCTION = "BAL"
te_transfer     TE_FUNCTION = "TRANSFER"
te_withdrawl    TE_FUNCTION = "WITHDRAW"
te_deposit      TE_FUNCTION = "DEPOSIT"

#
#TOP END RTQ queues available to Tuxedo
DEFAULT:   LACCESSPOINT = c03  RACCESSPOINT = b05  TYPE = QSPACE
tuxqspace  TE_RTQGROUP = "TEQGROUP"  TE_RTQNAME = "TEQNAME"

#
```

```
#tpenqueue と RTQ を介して Tuxedo が使用可能な TOP END サービス
DEFAULT:   LACCESSPOINT = c03  RACCESSPOINT = b05  TYPE = QNAME
te_report  TE_PRODUCT = "EBANK"  TE_FUNCTION = "REPORT"
te_update  TE_PRODUCT = "EBANK"  TE_FUNCTION = "UPDATE"

*DM_ROUTING
#
ACCOUNT FIELD = branchid  BUFTYPE = "VIEW:account"
        RANGES = "MIN-1000:b01,1001-3000:b02,*:b03"



*DM_ACCESS_CONTROL
#
branch ACLIST = "b01,b02,b03"
loans  ACLIST = b04
```

ネットワーク・アドレス

TDomain を実行するローカル・マシンが TCP/IP アドレス指定機能を使用していて、backus.company.com という名前になっているとします。マシンのアドレスは 155.2.193.18 です。さらに、TEDG が要求を受け取るポート番号は 2334 であるとします。このポート番号 2334 は、bankapp-gwaddr という名前のネットワーク・サービス・データベースに追加されているとします。このポートの完全なアドレスは、次のように表現されます。

```
//155.2.193.18:bankapp-gwaddr

//155.2.193.18:2334

//backus.company.com:bankapp-gwaddr

//backus.company.com:2334

0x0002091E9B02C112
```

上記の最後の表現は 16 進形式です。0002 は TCP/IP アドレスの先頭部分です。091E は 16 進数に変換されたポート番号 2334 です。アドレスの残りの部分 (9B02C112) は、IP アドレス (155.2.193.12) の各要素を 16 進数に変換したものです。たとえば、9B は 155 を変換したもの、02 は 2 を変換したものです。

関連項目

dmadmin(1)、dmloadcf(1)、dmunloadcf(1)、tmboot(1)、tmshutdown(1)、DMADM(5)、GWADM(5)、GWTOPEND(5)

『BEA TOP END Programmer's Reference Manual』の tp_intro(3T)、ni_config(4T)、nm_config(4T)、nm_script(4T)

『BEA Tuxedo アプリケーション実行時の管理』

『BEA Tuxedo アプリケーションの設定』

『C 言語を使用した BEA Tuxedo アプリケーションのプログラミング』

『BEA Tuxedo Domains コンポーネント』

『ATMI アプリケーションでの BEA Tuxedo TOP END Domain Gateway の使用』

DM_MIB(5)

名前 DM_MIB— ドメインの管理情報ベース

形式

```
#include <fml32.h>
#include <tpadm.h> /* MIB Header, includes DOMAINS */
```

Domains 関連の新しい用語

BEA Tuxedo のリリース 7.1 以降では、Domains 用の MIB で、ローカル・ドメインとリモート・ドメインとの相互作用を記述するため、クラスと属性の用語が改善されています。この改善された用語は、DMCONFIG ファイルの構文にも適用されています。

そのような用語の改善により、用語「ドメイン」が繰り返し使用されることがなくなり、行われるアクションをより明解に示す用語が導入されます。たとえば、用語「アクセス・ポイント」は別のオブジェクトにアクセスするために使用するオブジェクトのことです。したがって、リモート・ドメインにはリモート・ドメイン・アクセス・ポイントを通じてアクセスし、リモート・ドメインからローカル・ドメインにはローカル・ドメイン・アクセス・ポイントを通じてアクセスします。次の表は、用語「ドメイン」を何度も使用しない結果として生じる DMCONFIG のセクション名の変更を示しています。

元の DMCONFIG セクション名 . .	新しいセクション名 . .
DM_LOCAL_DOMAINS	DM_LOCAL
DM_REMOTE_DOMAINS	DM_REMOTE

これらのセクションの中で、次のパラメータ名が変更されています。

元のパラメータ名 . .	新しいパラメータ名 . .
DOMAINID	ACCESSPOINTID
MAXRDOM	MAXACCESSPOINT
MAXRDTRAN	MAXRAPTRAN

これらの DMCONFIG セクションに対応する DM_MIB クラスは、それぞれ T_DM_LOCAL と T_DM_REMOTE です。

特定のコンフィギュレーションでは、利用可能なサービスとリソース (キュー・スペースやキュー名など) を両方ともインポートおよびエクスポートする必要があります。したがって、DMCONFIG のセクション名 DM_LOCAL_SERVICES と DM_REMOTE_SERVICES では必要なアクティビティが適切に表現されなくなりました。これらのセクション名をそれぞれ DM_EXPORT および DM_IMPORT に置き換えると、行われるアクションが明確に表現されます。つまり、単一 BEA Tuxedo ドメインの観点から見て、ローカル・アクセス・ポイントを通じてそのドメインからリソースがエクスポートされ、リモート・ドメイン・アクセス・ポイントを通じてそのドメインにリソースがインポートされます。次の表は、それらの DMCONFIG セクション名の変更を示しています。

元の DMCONFIG セクション名 . .	新しいセクション名 . .
DM_LOCAL_SERVICES	DM_EXPORT
DM_REMOTE_SERVICES	DM_IMPORT

これらのセクションの中で、次のパラメータ名が変更されています。

元のパラメータ名 . .	新しいパラメータ名 . .
LDOM	LACCESSPOINT
RDOM	RACCESSPOINT

これらの DMCONFIG セクションに対応する DM_MIB クラスは、それぞれ T_DM_EXPORT と T_DM_IMPORT です。

下位互換性

BEA Tuxedo リリース 7.1 から導入された新しい Domains 用語は、DM_MIB のリファレンス・ページ、クラス、およびエラー・メッセージと、DMCONFIG のリファレンス・ページ、セクション名、パラメータ名、およびエラー・メッセージに適用されています。

旧バージョンとの互換性のため、BEA Tuxedo 7.1 より以前に使用された DMCONFIG 用語と、改善された Domains 用の MIB 用語との間にエイリアスが提供されています。BEA Tuxedo リリース 7.1 以降では、dmloadcf は両方の DMCONFIG 用語を使用できます。ただし、dmunloadcf は、デフォルトで新しいドメイン関連の用語を使用する DMCONFIG ファイルを生成します。以前のドメイン関連の用語を使用する DMCONFIG ファイルを生成するには、dmunloadcf の -c オプションを使用します。

機能説明

Domains 用の MIB では、ドメインがドメイン・ゲートウェイおよびドメイン・ゲートウェイ管理サーバを使用してサービスをインポートまたはエクスポートするためのクラスが定義されています。このリファレンス・ページは、BEA Tuxedo システムの Domains コンポーネントに関する知識がある読者を対象としています。Domains コンポーネントについては、『BEA Tuxedo Domains コンポーネント』を参照してください。

管理要求のフォーマットと管理応答の解釈を行うには、DM_MIB(5) を共通 MIB リファレンス・ページ MIB(5) と一緒に使用してください。

DM_MIB で説明するクラスや属性を使用し、MIB(5) の説明に従ってフォーマットした要求を使用すると、アクティブなアプリケーションの既存の ATMI インターフェイスを通じて管理サービスを要求できます。DM_MIB(5) のすべてのクラス定義の追加情報については、263 ページの「DM_MIB(5) に関する追加情報」を参照してください。

DM_MIB(5) は、次のクラスで構成されています。

表 11DM_MIB のクラス

クラス名	属性
T_DM_ACL	ドメインのアクセス制御リスト
T_DM_CONNECTION	2 つのドメイン間の接続状況
T_DM_EXPORT	エクスポートされるリソース
T_DM_IMPORT	インポートされるリソース
T_DM_LOCAL	ローカル・アクセス・ポイント

表 11DM_MIB のクラス

クラス名	属性
T_DM_OSITP	OSI TP 1.3 固有の、アクセス・ポイントのコンフィギュレーション
T_DM_OSITPX	OSI TP 4.0 以降固有の、アクセス・ポイントのコンフィギュレーション
T_DM_PASSWORD	ドメインのパスワード・エントリ
T_DM_PRINCIPAL_MAP	プリンシパル・マッピング・エントリ
T_DM_REMOTE	リモート・アクセス・ポイント
T_DM_RESOURCES	グローバル Domains コンフィギュレーション情報
T_DM_ROUTING	アクセス・ポイントのルーティング基準
T_DM_RPRINCIPAL	リモート・プリンシパル・エントリ
T_DM_SNACRM	SNA-CRM 固有の、ローカル・アクセス・ポイントのコンフィギュレーション
T_DM_SNALINK	SNAX 固有の、リモート・ドメイン・アクセス・ポイントのコンフィギュレーション
T_DM_SNASTACK	特定の SNA CRM で使用される SNA スタック
T_DM_TDOMAIN	TDomain 固有の、アクセス・ポイントのコンフィギュレーション
T_DM_TOPEND	BEA TOP END 固有の、アクセス・ポイントのコンフィギュレーション
T_DM_TRANSACTION	ローカル・アクセス・ポイントと関連付けられたトランザクション・エントリ

各クラスの説明は、次の 4 つのセクションで構成されています。

- 概要 — クラスに関連付けられている属性の概要。

- 属性表 — クラスの各属性に関する名前、タイプ、パーミション、値、およびデフォルト値を示す表。属性表の形式については以下に示してあります。
- 属性の意味 — クラスの各属性の解釈を示します。
- 制限事項 — このクラスにアクセスし、このクラスを解釈する場合の制限事項。

属性表の形式

属性表はクラス内の属性のリファレンス・ガイドであり、管理者、オペレータ、一般ユーザがそれらの属性を使用してアプリケーションと対話するための方法を説明しています。

属性表の各属性の説明には、5 つの構成要素 (名前、タイプ、パーミッション、値、デフォルト) があります。各要素については、MIB(5) を参照してください。

TA_FLAGS 値

MIB(5) は、共通 TA_FLAGS 属性を定義します。この属性は long 値フィールドで、共通 MIB フラグ値とコンポーネント MIB 固有フラグ値の両方を持ちます。現時点では、DM_MIB 固有のフラグ値は定義されていません。

FML32 フィールド・テーブル

このリファレンス・ページで説明する属性のフィールド・テーブルは、システムにインストールした BEA Tuxedo システム・ソフトウェアのルート・ディレクトリからの相対パスで指定される udataobj/tpadm ファイルにあります。${TUXDIR}/udataobj ディレクトリは、FLDTBLDIR 環境変数で指定されるコロン区切りのリストにアプリケーションによって追加される必要があります。フィールド・テーブル名 tpadm は、FIELDTBLS 環境変数で指定されるカンマ区切りのリストに追加される必要があります。

相互運用性

この MIB のヘッダ・ファイルとフィールド・テーブルには、BEA Tuxedo リリース 7.1 以降のサイト (ネイティブとワークステーションの両方) からのみアクセスできます。アプリケーション内でリリース 5.0 以前のサイトがアクティブになっている場合、グローバル情報の更新 ("SET" 操作) はそれらのサイトのゲートウェイ・グループでは利用できません。

リリース 5.0 以前のサイトのローカル情報にはアクセスできません。アクセスされるクラスにグローバル情報も含まれている場合は、グローバル情報のみ返されます。グローバル情報がない場合は、エラーが返されます。

移植性　BEA Tuxedo システムの MIB を使用した管理作業をサポートするために必要な既存の FML32 および ATMI 関数、さらにこのリファレンス・ページに定義するヘッダ・ファイルとフィールド・テーブルは、すべてのサポート対象ネイティブ・プラットフォームとワークステーション・プラットフォームで使用可能です。

T_DM_ACL クラスの定義

概要 T_DM_ACL クラスは、ドメインのアクセス制御情報を表します。

属性表

表 12DM_MIB(5): T_DM_ACL **クラス定義の属性表**

属性	タイプ	パーミッション	値	デフォルト値
TA_DMACLNAME (r) (k) (*)	string	rw-r--r--	*string* [1..15]	N/A
TA_DMRACCESSPOINTLIST (*)	string	rw-r--r--	*string* [0.0,1550]	""
TA_STATE(r)	string	rw-r--r--	GET: "VAL" SET: "{NEW|INV}"	N/A N/A

(r)— 新しいオブジェクトが作成される場合に必須です。
(k)— オブジェクトを取り出すためのキー・フィールドです。
(*)— すべての SET 操作で必須のキー・フィールドです。

属性の意味

TA_DMACLNAME: *string* [1..15]

Domains コンフィギュレーションの T_DM_ACL エントリの名前の中で一意な、アクセス制御リストの名前。

TA_DMRACCESSPOINTLIST: *string* [0..1550]

このアクセス制御リストと関連付けられたリモート・ドメイン・アクセス・ポイントのリスト。TA_DMRACCESSPOINTLIST は、リモート・ドメイン・アクセス・ポイント名 (有効な T_DM_REMOTE オブジェクトの TA_DMRACCESSPOINT 属性の値) のカンマ区切りのリストです。リストには、リモート・ドメイン・アクセス・ポイントの識別子要素を 50 個まで格納できます。この属性を "*" に設定すると、コンフィギュレーションのすべてのリモート・ドメインがこのエントリと関連付けられます。"" は、リモート・ドメイン・アクセス・ポイントがこのエントリと関連付けられないことを意味します。デフォルトは "" です。

TA_STATE:

GET: "{VALid}"

GET 操作は、T_DM_ACL オブジェクトのコンフィギュレーション情報を検索します。以下に示す状態は、GET 要求に対する応答で返される TA_STATE 属性の意味を示します。以下に示されていない状態は返されません。

"VALid"	オブジェクトが定義され、非アクティブ状態です。これがこのクラスの唯一の有効な状態です。ACL グループがアクティブになることはありません。

SET: "{NEW | INValid}"

SET 操作は、選択した T_DM_ACL オブジェクトのコンフィギュレーション情報を更新します。以下に示す状態は、SET 要求で設定される TA_STATE の意味を示します。以下に示されていない状態は設定できません。

"NEW"	新しいオブジェクトを作成します。状態の変更は、"INValid" 状態でのみ可能です。正常に終了すると、オブジェクトの状態は "VALid" になります。
unset	既存のオブジェクトを変更します。この組み合わせは "INValid" 状態では使用できません。正常に終了してもオブジェクトの状態は変わりません。
"INValid"	オブジェクトを削除します。状態の変更は、"VALid" 状態でのみ可能です。正常に終了すると、オブジェクトの状態は "INValid" になります。

制限事項　なし

T_DM_CONNECTION クラスの定義

概要 T_DM_CONNECTION クラスは、ドメイン・アクセス・ポイント間の接続のステータスを表します。

属性表

表 13DM_MIB(5): T_DM_CONNECTION **クラス定義の属性表**

属性	タイプ	パーミッション	値	デフォルト値
TA_DMLACCESSPOINT(k)(*)	string	rw-r--r--	*string*[1..30]	N/A
TA_DMRACCESSPOINT(k)	string	rw-r--r--	*string*[1..30]	N/A
TA_DMTYPE	string	r--r--r--	"{TDOMAIN | TOPEND}"	N/A
TA_STATE(k)(*)	string	rwxr-xr--	GET: "{ACT | SUS | INI | INA | UNK}" SET: "{ACT | INA}"	N/A N/A
TA_DMTYPE=TDOMAIN の場合に設定可能な属性				
TA_DMCURENCRYPTBITS	string	r--------	"{0 | 40 | 56 | 128}"(注 1)	"0"

(k)— オブジェクトを取り出すためのキー・フィールドです。
(*)— すべての SET 操作で必須のキー・フィールドです。

注 1 リンク・レベルの暗号化の値 40 は、下位互換性を維持するために提供されています。

属性の意味

TA_DMLACCESSPOINT: *string*[1..30]
ドメイン間の接続を識別するローカル・ドメイン・アクセス・ポイントの名前。

GET および SET 操作では、この属性に特定のローカル・ドメイン・アクセス・ポイントを指定する必要があります。

TA_DMRACCESSPOINT: *string*[1..30]
ドメイン間の接続を識別するリモート・ドメイン・アクセス・ポイントの名前。

GET および SET 操作で、TA_DMRACCESSPOINT が指定されていない場合は、TA_DMLACCESSPOINT で指定されたローカル・アクセス・ポイントのすべての T_DM_CONNECTION エントリが選択されます。

TA_DMTYPE: "{TDOMAIN | TOPEND}"
ドメインのタイプ。"TDOMAIN" または "TOPEND" のいずれかです。

TA_STATE:

GET: "{ACTive | SUSpended | INItializing | INActive | UNKnown}"
GET 操作は、接続の実行時情報を検索します。以下に示す状態は、GET 要求に対する応答で返される TA_STATE 属性の意味を示します。以下に示されていない状態は返されません。

"ACTive"	接続がアクティブです。
"SUSpended"	接続が再試行を待っています。
"INItializing"	接続が初期化されています。
"INActive"	指定されたドメイン・アクセス・ポイントの接続が切断されています。この状態は、BEA Tuxedo リリース 7.1 以降が動作するゲートウェイによってのみ返されます。
"UNKnown"	指定されたドメイン・アクセス・ポイントの接続状態を確認できません。

SET: "{ACTive | INActive}"
SET 操作は、接続の実行時情報を更新します。以下に示す状態は、SET 要求の TA_STATE の意味を示します。以下に示されていない状態は設定できません。

"ACTive"	指定されたドメイン・アクセス・ポイントを接続します。現在の状態が "SUSpended" または "INActive" の場合、SET:"ACTive" は接続の状態を "INItializing" にします。それ以外の場合は、変更はありません。

"INActive"	指定されたドメイン・アクセス・ポイントの接続を切断し、オブジェクトを破棄します。

TA_DMTYPE=TDOMAIN の場合に設定可能な属性

TA_DMCURENCRYPTBITS: "{0 | 40 | 56 | 128}"

この接続で使用する暗号化のレベル。"0" は暗号化のないことを意味し、"40"、"56"、および "128" は暗号化キーのビット長を示します。この属性は、BEA Tuxedo リリース 7.1 以降が動作するゲートウェイでのみ有効です。ほかのゲートウェイの場合、この値は "0" に設定します。

注記　リンク・レベルの暗号化の値 40 は、下位互換性を維持するために提供されています。

制限事項　ドメイン・ゲートウェイ管理 (GWADM) サーバおよび TA_DMLACCESSPOINT 属性で指定されたローカル・ドメイン・アクセス・ポイントをサポートするドメイン・ゲートウェイがアクティブでないと、そのアクセス・ポイントへの接続で GET 操作または SET 操作を実行することはできません。

T_DM_EXPORT クラスの定義

概要 T_DM_EXPORT クラスは、ローカル・アクセス・ポイントを通じて 1 つ以上のリモート・ドメインにエクスポートされるローカル・リソースを表します。

属性表

表 14DM_MIB(5): T_DM_EXPORT **クラス定義の属性表**

属性	タイプ	パーミッション	値	デフォルト値
TA_DMRESOURCENAME(r)(k)(*)	string	rw-r--r--	*string*[1..15]	N/A
TA_DMLACCESSPOINT(k)(*)	string	rw-r--r--	*string*[1..30]	* (すべてということ)
TA_STATE(r)	string	rw-r--r--	GET: "VAL" SET: "{NEW | INV}"	N/A N/A
TA_DMACLNAME	string	rw-r--r--	*string*[1..15]	N/A
TA_DMCONV	string	rw-r--r--	"{Y | N}"	"N"
TA_DMREMOTENAME	string	rw-r--r--	*string*[1..30]	N/A
TA_DMTYPE=TOPEND のリモート・ドメイン・アクセス・ポイントから設定可能な属性				
TA_DMRESOURCETYPE	string	rw-r--r--	"{SERVICE | QSPACE | QNAME}"	"SERVICE"
TA_DMTE_PRODUCT	string	rw-r--r--	*string*[1..32]	
TA_DMTE_FUNCTION	string	rw-r--r--	*string*[1..8]	
TA_DMTE_TARGET	string	rw-r--r--	*string*[1..8]	スペース
TA_DMTE_QUALIFIER	long	rw-r--r--	0 <= *num* <= MAXLONG	0 (ゼロ)
TA_DMTE_RTQGROUP	string	rw-r--r--	*string*[1..32]	
TA_DMTE_RTQNAME	string	rw-r--r--	*string*[1..8]	
TA_DMTYPE=TOPEND|SNAX|**OSITP**|**OSITPX** のリモート・ドメイン・アクセス・ポイントから設定可能な属性				

表 14DM_MIB(5): T_DM_EXPORT **クラス定義の属性表 (続き)**

属性	タイプ	パーミッション	値	デフォルト値
TA_DMINBUFTYPE	string	rw-r--r--	*string*[0..513]	N/A
TA_DMOUTBUFTYPE	string	rw-r--r--	*string*[0..513]	N/A
TA_DMTYPE=OSITPX のリモート・ドメイン・アクセス・ポイントから設定可能な属性				
TA_DMCOUPLING(r)	string	rw-r--r--	"{TIGHT | LOOSE}"	"LOOSE"
TA_DMINRECTYPE(r)	string	rw-r--r--	*string*[0..78]	""
TA_DMOUTRECTYPE(r)	string	rw-r--r--	*string*[0..78]	""

(r)— 新しいオブジェクトが作成される場合に必須です。
(k)— オブジェクトを取り出すためのキー・フィールドです。
(*)— すべての SET 操作で必須のキー・フィールドです。

属性の意味

TA_DMRESOURCENAME: *string*[1..15]

リソース・タイプ SERVICE (サービス名)、QSPACE (キュー・スペース名)、および QNAME (キュー名) のエントリのローカル・リソース名。SERVICE エントリの場合、この属性の値はアクティブな T_SVCGRP オブジェクトの TA_SERVICENAME 属性の値に対応します。このリソースは、同じ名前あるいは TA_DMREMOTENAME 属性または TA_DMTE* 属性で定義されたエイリアスを使用してリモート・ドメインにエクスポートされます。

TA_DMLACCESSPOINT: *string*[1..30]

このローカル・リソースが利用可能なローカル・アクセス・ポイントの名前。この属性を "*" に設定すると、すべてのローカル・アクセス・ポイントでリソースが利用可能になります。

TA_STATE:

GET: "{VALid}"

GET 操作は、T_DM_EXPORT オブジェクトのコンフィギュレーション情報を検索します。以下に示す状態は、GET 要求に対する応答で返される TA_STATE 属性の意味を示します。以下に示されていない状態は返されません。

"VALid"	オブジェクトが存在します。

SET: "{NEW | INValid}"

SET 操作は、選択した T_DM_EXPORT オブジェクトのコンフィギュレーション情報を更新します。以下に示す状態は、SET 要求で設定される TA_STATE の意味を示します。以下に示されていない状態は設定できません。

"NEW"	新しいオブジェクトを作成します。
unset	既存のオブジェクトを変更します。この組み合わせは "INValid" 状態では使用できません。正常に終了してもオブジェクトの状態は変わりません。
"INValid"	オブジェクトを削除します。

TA_DMACLNAME: *string*[1..15]

このローカル・リソースのセキュリティに使用する T_DM_ACL オブジェクトの名前。TOPEND リモート・ドメイン・アクセス・ポイントからのアクセスが許可されている場合、このオプションの属性は TA_DMRESOURCETYPE="SERVICE" または "QSPACE" であれば指定できます。この属性は、TA_DMRESOURCETYPE="QNAME" の場合は指定できません。

TA_DMCONV: "{Y | N}"

このローカル・リソースが会話型かどうかを指定します。TOPEND リモート・ドメイン・アクセス・ポイントからのアクセスが許可されている場合、この属性は TA_DMRESOURCETYPE="QSPACE" または "QNAME" であれば "N" に設定する必要があります。

TA_DMREMOTENAME: *string*[1..30]

リモート・ドメイン・アクセス・ポイントを通じてエクスポートされるこのローカル・リソースの名前を指定します。この属性が指定されていない場合、ローカル・リソースの名前には TA_DMRESOURCENAME で指定された名前がデフォルトで使用されます。TA_DMREMOTENAME 属性は、TOPEND タイプのドメイン・ゲートウェイには適用されません。

TA_DMTYPE=TOPEND のリモート・ドメイン・アクセス・ポイントから設定可能な属性

TA_DMRESOURCETYPE: "{SERVICE | QSPACE | QNAME}"
: このローカル・リソースが "SERVICE"、"QSPACE"、または "QNAME" のどれであるかを指定します。デフォルトは "SERVICE" です。

TA_DMTE_PRODUCT: *string*[1..32]
: このローカル・リソースの BEA TOP END 製品名。TOPEND リモート・ドメイン・アクセス・ポイントからのアクセスが許可されている場合、TA_DMRESOURCETYPE="SERVICE" または "QNAME" であればこの属性を指定する必要があります。この属性は、TA_DMRESOURCETYPE="QSPACE" の場合は指定できません。

TA_DMTE_FUNCTION: *string*[1..8]
: このローカル・リソースの BEA TOP END 関数名。TOPEND リモート・ドメイン・アクセス・ポイントからのアクセスが許可されている場合、TA_DMRESOURCETYPE="SERVICE" または "QNAME" であればこの属性を指定する必要があります。この属性は、TA_DMRESOURCETYPE="QSPACE" の場合は指定できません。

TA_DMTE_TARGET: *string*[1..8]
: このローカル・リソースの BEA TOP END メッセージ・センシティブ・ルーティング (MSR) ターゲット。TOPEND リモート・ドメイン・アクセス・ポイントからのアクセスが許可されている場合、このオプションの属性は TA_DMRESOURCETYPE="SERVICE"、"QSPACE"、または "QNAME" であれば指定できます。

TA_DMTE_QUALIFIER: 0 <= *num* <= MAXLONG
: このローカル・リソースの BEA TOP END 関数修飾子。TOPEND リモート・ドメイン・アクセス・ポイントからのアクセスが許可されている場合、このオプションの属性は TA_DMRESOURCETYPE="SERVICE" または "QNAME" であれば指定できます。この属性は、TA_DMRESOURCETYPE="QSPACE" の場合は指定できません。

TA_DMTE_RTQGROUP: *string*[1..32]
: このローカル・リソースの BEA TOP END 回復可能トランザクション・キューイング (RTQ) キュー・グループ名。TOPEND リモート・ドメイン・アクセス・ポイントからのアクセスが許可されている場合、TA_DMRESOURCETYPE="QSPACE" であればこの属性を指定する必要があ

ります。この属性は、TA_DMRESOURCETYPE="SERVICE" または "QNAME" の場合は指定できません。

TA_DMTE_RTQNAME: *string*[1..8]
: このローカル・リソースの BEA TOP END RTQ キュー名。TOPEND リモート・ドメイン・アクセス・ポイントからのアクセスが許可されている場合、TA_DMRESOURCETYPE="QSPACE" であればこの属性を指定する必要があります。この属性は、TA_DMRESOURCETYPE="SERVICE" または "QNAME" の場合は指定できません。

TA_DMTYPE=TOPEND|SNAX|OSITP|OSITPX のリモート・ドメイン・アクセス・ポイントから設定可能な属性

TA_DMINBUFTYPE: *string*[0..513]
: *type*[:*subtype*]— このローカル・リソースの入力バッファ・タイプを (必要に応じてサブタイプも続けて) 指定します。この属性がある場合は、受け付けたバッファ・タイプ (およびサブタイプ) が定義されます。TOPEND リモート・ドメイン・アクセス・ポイントからのアクセスが許可されている場合、このオプションの属性は TA_DMRESOURCETYPE="SERVICE" または "QNAME" であれば指定できます。TA_DMRESOURCETYPE="QSPACE" の場合、この属性は指定できません。この属性は、SNAX を使用する場合、あるいは UDT アプリケーション・コンテキストで OSITP または OSITPX を使用してリモート・ドメイン・アクセス・ポイントからのアクセスが許可される場合に TA_DMRESOURCETYPE="SERVICE" のエントリに対して定義する必要があります。

  BEA TOP END サービス名エントリおよびキュー名エントリの場合、*type* の有効な値は FML32、CARRAY、および X_OCTET です。

TA_DMOUTBUFTYPE: *string*[0..513]
: *type*[:*subtype*]– このローカル・リソースの出力バッファ・タイプを (必要に応じてサブタイプも続けて) 指定します。この属性がある場合は、サービスで出力されたバッファ・タイプ (およびサブタイプ) が定義されます。TOPEND リモート・ドメイン・アクセス・ポイントからのアクセスが許可されている場合、このオプションの属性は TA_DMRESOURCETYPE="SERVICE" であれば指定できます。TA_DMRESOURCETYPE="QSPACE" または QNAME" の場合、この属性は指定

できません。この属性は、SNAX を使用する場合、あるいは UDT アプリケーション・コンテキストで OSITP または OSITPX を使用してリモート・ドメイン・アクセス・ポイントからのアクセスが許可される場合に TA_DMRESOURCETYPE="SERVICE" のエントリに対して定義する必要があります。

BEA TOP END サービス名エントリおよびキュー名エントリの場合、*type* の有効な値は FML32、CARRAY、および X_OCTET です。

TA_DMTYPE=OSITPX のリモート・ドメイン・アクセス・ポイントから設定可能な属性

TA_DMCOUPLING: *string*"{TIGHT | LOOSE}"

このローカル・サービスに対する要求が同じリモート・ドメイン・アクセス・ポイントを通じて送られてくる場合にトランザクションの結合が疎結合であるか密結合であるかを指定します。デフォルトは、"LOOSE" です。TA_DMCOUPLING="LOOSE" を設定した場合、両方の要求が同じグローバル・トランザクションに参加していても、このローカル・サービスに対する最初の要求で行われたデータベースの更新は 2 番目の要求から確認できません。TA_DMCOUPLING="TIGHT" を設定した場合、同じリモート・ドメイン・アクセス・ポイントを通じて行われた同じローカル・サービスへの複数の呼び出しは密結合されます。つまり、最初の要求で行われたデータベースの更新が 2 番目の要求から確認できるということです。

TA_DMCOUPLING="TIGHT" は、同じリモート・ドメイン・アクセス・ポイントを通じて重複したサービス要求が送られてきたときにのみ適用されます。サービス要求が異なるリモート・ドメイン・アクセス・ポイントを通る場合、要求は常に疎結合されます。

TA_DMINRECTYPE: *string*[1..78]

type[:*subtype*]— このローカル・サービスで特定のクライアントが要求する応答バッファのタイプを (必要に応じてサブタイプも続けて) 指定します。場合によっては、応答バッファの形式も指定します。この属性は、リモート・クライアントが要求するバッファとタイプおよび構造が同じバッファをローカル・サービスが送信する場合は省略できます。TA_DMINRECTYPE を指定しない場合、バッファのタイプは変わりません。

TA_DMOUTRECTYPE: *string*[1..78]

type[:*subtype*]— リモート・クライアントによってこのローカル・サービスに送られるバッファのタイプと、オプションでサブタイプを指定します。この属性は、厳密なタイプ・チェックに使用します。

制限事項 このクラスのインスタンスを追加または更新する SET 操作の実行時、および特定のローカル・ドメイン・アクセス・ポイントを TA_DMLACCESSPOINT 属性で指定する場合、アクセス・ポイントが T_DM_LOCAL クラス内に存在しなければなりません。存在しない場合、TA_DMLACCESSPOINT 属性に対して "not defined" エラーが返され、操作は失敗します。

T_DM_IMPORT クラスの定義

概要 T_DM_IMPORT クラスは、1 つまたは複数のリモート・ドメイン・アクセス・ポイントを通してインポートされ、1 つまたは複数のローカル・ドメイン・アクセス・ポイントを通してローカル・ドメインで使用可能なリモート・リソースを表します。

属性表

表 15DM_MIB(5): T_DM_IMPORT **クラス定義の属性表**

属性	タイプ	パーミッション	値	デフォルト値
TA_DMRESOURCENAME(r)(k)(*)	string	rw-r--r--	*string*[1..15]	
TA_DMRACCESSPOINTLIST(k)(*)	string	rw-r--r--	*string*[1..92]	* (すべてということ)
TA_DMLACCESSPOINT(k)(*)	string	rw-r--r--	*string*[1..30]	* (すべてということ)
TA_STATE(r)	string	rwxr-xr--	GET: "VAL" SET: "{NEW | INV}"	N/A N/A
TA_DMCONV	string	rw-r--r--	"{Y | N}"	"N"
TA_DMLOAD	short	rw-r--r--	1 <= *num* <= 32,767	50
TA_DMREMOTENAME	string	rw-r--r--	*string*[1..30]	N/A
TA_DMRESOURCETYPE	string	rw-r--r--	"{SERVICE | QSPACE | QNAME}"	"SERVICE"
TA_DMROUTINGNAME	string	rw-r--r--	*string*[1..15]	N/A
TA_DMTYPE=TOPEND: のリモート・ドメイン・アクセス・ポイントから設定可能な属性				
TA_DMTE_PRODUCT	string	rw-r--r--	*string*[1..32]	
TA_DMTE_FUNCTION	string	rw-r--r--	*string*[1..8]	

表 15DM_MIB(5): T_DM_IMPORT **クラス定義の属性表 (続き)**

属性	タイプ	パーミッション	値	デフォルト値
TA_DMTE_TARGET	string	rw-r--r--	*string*[1..8]	スペース
TA_DMTE_QUALIFIER	long	rw-r--r--	0 <= *num* <= MAXLONG	0 (ゼロ)
TA_DMTE_RTQGROUP	string	rw-r--r--	*string*[1..32]	
TA_DMTE_RTQNAME	string	rw-r--r--	*string*[1..8]	
TA_DMTYPE=TOPEND|SNAX|**OSITP**|**OSITPX**: のリモート・ドメイン・アクセス・ポイントから設定可能な属性				
TA_DMINBUFTYPE	string	rw-r--r--	*string*[0..256]	N/A
TA_DMOUTBUFTYPE	string	rw-r--r--	*string*[0..256]	N/A
TA_DMTYPE=OSITPX: のリモート・ドメイン・アクセス・ポイントから設定可能な属性				
TA_DMAUTOPREPARE(r)	string	rw-r--r--	"{Y|N}"	"N"
TA_DMINRECTYPE(r)	string	rw-r--r--	*string*[0..78]	""
TA_DMOUTRECTYPE(r)	string	rw-r--r--	*string*[0..78]	""
TA_DMTPSUTTYPE(r)	string	rw-r--r--	"{INTEGER| PRINTABLESTRING}"	""
TA_DMREMTPSUT(r)	string	rw-r--r--	*string*[0..64]	""

(r)― 新しいオブジェクトが作成される場合に必須です。
(k)― オブジェクトを取り出すためのキー・フィールドです。
(*)― すべての SET 操作で必須のキー・フィールドです。

属性の意味 TA_DMRESOURCENAME: *string*[1..15]

リソース (IPC 関連)、リソース (DBMS 関連)・タイプ SERVICE (サービス名)、QSPACE (キュー・スペース名)、および QNAME (キュー名) のエントリで使用するリモート・リソース名。このリソースは、同じ名前あるいは TA_DMREMOTENAME 属性または TA_DMTE* 属性で定義されたエイリアスを使用してリモート・ドメインからインポートされます。

TA_DMRACCESSPOINTLIST: *string*[1..92]

このリモート・リソースをインポートするためのリモート・ドメイン・アクセス・ポイントの名前を指定します。

TA_DMRACCESSPOINTLIST は、カンマ区切りのフェイルオーバー・ドメイン・リストで、最大 30 文字のリモート・ドメイン・アクセス・ポイントを 3 つまで格納できます。この属性を "*" に設定すると、すべてのリモート・ドメイン・アクセス・ポイントからリソースをインポートできます。

TA_DMLACCESSPOINT: *string*[1..30]

このリモート・リソースが利用可能なローカル・ドメイン・アクセス・ポイントの名前。この属性を "*" に設定すると、すべてのローカル・ドメイン・アクセス・ポイントを通してリソースを利用できます。

TA_STATE:

GET: "{VALid}"

GET 操作は、T_DM_IMPORT オブジェクトのコンフィギュレーション情報を検索します。以下に示す状態は、GET 要求に対する応答で返される TA_STATE 属性の意味を示します。以下に示されていない状態は返されません。

"VALid"	オブジェクトが存在します。

SET: "{NEW | INValid}"

SET 操作は、選択した T_DM_IMPORT オブジェクトのコンフィギュレーション情報を更新します。以下に示す状態は、SET 要求の TA_STATE の意味を示します。以下に示されていない状態は設定できません。

"NEW"	新しいオブジェクトを作成します。状態変更は、"INValid" 状態でのみ可能です。正常に終了すると、オブジェクトの状態は "ACTive" になります。

unset	既存のオブジェクトを変更します。この組み合わせは "INValid" 状態では使用できません。正常に終了してもオブジェクトの状態は変わりません。
"INValid"	オブジェクトを削除します。状態変更は、"ACTive" 状態でのみ可能です。正常に終了すると、オブジェクトの状態は "INValid" になります。

TA_DMCONV: "{Y | N}"

このリモート・リソースが会話型であるかどうかを指定する boolean 型の値 ("Y" または "N")。TOPEND リモート・ドメイン・アクセス・ポイントからのアクセスが許可されている場合、この属性は TA_DMRESOURCETYPE="QSPACE" または "QNAME" であれば "N" に設定する必要があります。

TA_DMLOAD: 1 <= *num* <= 32,767

このリモート・リソースのサービス負荷。インターフェイスの負荷は、ロード・バランシングのために使用します。つまり、すでに負荷が大きいキューは、新規の要求ではあまり選択されません。

TA_DMREMOTENAME: *string*[1..30]

リモート・ドメイン・アクセス・ポイントを通じてインポートされるこのリモート・リソースの名前を指定します。この属性が指定されていない場合、リモート・リソースの名前には TA_DMRESOURCENAME で指定された名前がデフォルトで使用されます。TA_DMREMOTENAME 属性は、TOPEND タイプのドメイン・ゲートウェイには適用されません。

TA_DMRESOURCETYPE: "{SERVICE | QSPACE | QNAME}"

このリモート・リソースが "SERVICE"、"QSPACE"、または "QNAME" のどれであるかを指定します。デフォルトは "SERVICE" です。この属性は、TOPEND タイプのドメイン・ゲートウェイにのみ適用されます。つまり、ほかのタイプのドメイン・ゲートウェイでは、リモート・リソースは常に "SERVICE" になります。

TA_DMROUTINGNAME: *string*[1..15]

このリモート・リソース ("SERVICE" または "QSPACE") のルーティング基準として使用する T_DM_ROUTING オブジェクトの名前。

TA_DMTYPE=TOPEND のリモート・ドメイン・アクセス・ポイントから設定可能な属性

TA_DMTE_PRODUCT: *string*[1..32]
: このリモート・リソースの BEA TOP END 製品名。この属性は、TA_DMRESOURCETYPE="SERVICE" または "QNAME" の場合は必ず指定する必要があります。TA_DMRESOURCETYPE="QSPACE" の場合は指定できません。

TA_DMTE_FUNCTION: *string*[1..8]
: このリモート・リソースの BEA TOP END 関数名。この属性は、TA_DMRESOURCETYPE="SERVICE" または "QNAME" の場合は必ず指定する必要があります。TA_DMRESOURCETYPE="QSPACE" の場合は指定できません。

TA_DMTE_TARGET: *string*[1..8]
: このリモート・リソースの BEA TOP END メッセージ・センシティブ・ルーティング (MSR) ターゲット。この属性はオプションで、TA_DMRESOURCETYPE="SERVICE"、"QSPACE"、または "QNAME" の場合に指定できます。

TA_DMTE_QUALIFIER: 0 <= *num* <= MAXLONG
: このリモート・リソースの BEA TOP END 関数修飾子。この属性はオプションで、TA_DMRESOURCETYPE="SERVICE" または "QNAME" の場合に指定できます。TA_DMRESOURCETYPE="QSPACE" の場合は指定できません。

TA_DMTE_RTQGROUP: *string*[1..32]
: このリモート・リソースの BEA TOP END 回復可能トランザクション・キューイング (RTQ) キュー・グループ名。この属性は、TA_DMRESOURCETYPE="QSPACE" の場合は必ず指定する必要があります。TA_DMRESOURCETYPE="SERVICE" または "QNAME" の場合は指定できません。

TA_DMTE_RTQNAME: *string*[1..8]
: このリモート・リソースの BEA TOP END RTQ キュー名。この属性は、TA_DMRESOURCETYPE="QSPACE" の場合は必ず指定する必要があります。TA_DMRESOURCETYPE="SERVICE" または "QNAME" の場合は指定できません。

TA_DMTYPE=TOPEND|SNAX|OSITP|OSITPX のリモート・ドメイン・アクセス・ポイントから設定可能な属性

TA_DMINBUFTYPE: *string*[0..256]

type[:*subtype*]— このリモート・リソースの入力バッファ・タイプを (必要に応じてサブタイプも続けて) 指定します。この属性がある場合は、受け付けたバッファ・タイプ (およびサブタイプ) が定義されます。TOPEND リモート・ドメイン・アクセス・ポイントからのアクセスが許可されている場合、このオプションの属性は TA_DMRESOURCETYPE="SERVICE" または "QNAME" であれば指定できます。TA_DMRESOURCETYPE="QSPACE" の場合、この属性は指定できません。この属性は、SNAX を使用する場合、あるいは UDT アプリケーション・コンテキストで OSITP または OSITPX を使用してリモート・ドメイン・アクセス・ポイントへのアクセスが許可される場合に DMRESOURCETYPE="SERVICE" のエントリに対して定義する必要があります。

BEA TOP END サービス名エントリおよびキュー名エントリの場合、*type* の有効な値は FML32、CARRAY、および X_OCTET です。

TA_DMOUTBUFTYPE: *string*[0..256]

type[:*subtype*]— このリモート・リソースの出力バッファ・タイプを (必要に応じてサブタイプも続けて) 指定します。この属性がある場合は、サービスで出力されたバッファ・タイプ (およびサブタイプ) が定義されます。"TOPEND リモート・ドメイン・アクセス・ポイントからのアクセスが許可されている場合、このオプションの属性は TA_DMRESOURCETYPE="SERVICE" であれば指定できます。TA_DMRESOURCETYPE="QSPACE" または QNAME" の場合、この属性は指定できません。この属性は、SNAX を使用する場合、あるいは UDT アプリケーション・コンテキストで OSITP または OSITPX を使用してリモート・ドメイン・アクセス・ポイントへのアクセスが許可される場合に DMTYPE="SERVICE" のエントリに対して定義する必要があります。

BEA TOP END サービス名エントリおよびキュー名エントリの場合、*type* の有効な値は FML32、CARRAY、および X_OCTET です。

TA_DMTYPE=OSITPX のリモート・ドメイン・アクセス・ポイントから設定可能な属性

TA_DMAUTOPREPARE: *string*"{Y | N}"
: このリモート・サービスに対するグローバル・トランザクションに関与する単一の tpcall() で、呼び出しを自動的に準備できるようにします。この最適化により、2 フェーズ・コミット・プロセスを 1 ステップで実行できます。リモートの OSITP ドメインは、この機能をサポートしている必要があります。デフォルトは "N" です。

TA_DMINRECTYPE: *string*[1..78]
: *type*[:*subtype*]— このリモート・サービスに必要な要求バッファのタイプを (必要に応じてサブタイプも続けて) 指定します。場合によっては、要求バッファの形式も指定します。この属性は、リモート・サービスが要求するバッファとタイプおよび構造が同じバッファをローカル・クライアントが送信する場合は省略できます。TA_DMINRECTYPE を指定しない場合、バッファのタイプは変わりません。

TA_DMOUTRECTYPE: *string*[1..78]
: *type*[:*subtype*]— このリモート・サービスが送信するバッファのタイプを (必要に応じてサブタイプも続けて) 指定します。この属性は、厳密なタイプ・チェックに使用します。

TA_DMTPSUTTYPE: *string*"{INTEGER | PRINTABLESTRING}"
: このリモート・サービスの TA_DMREMTPSUT 値で実行する符号化のタイプを指定します。"INTEGER" および "PRINTABLESTRING" は ASN.1 タイプです。デフォルトは "PRINTABLESTRING" です。

TA_DMREMTPSUT: *string*[1..64]
: このリモート・サービスを提供するリモート・システムの TP サービス・ユーザ・タイトルを識別します。OSI TP インプリメンテーションのユーザの一部はこの属性が必要です。OS 2200 OLTP-TM2200、OpenTI、A Series Open/OLTP、および BEA eLink OSI TP の場合は必要ありません。TA_DMTPSUTTYPE の値が "PRINTABLESTRING" である場合、最大長は 60 文字です。PRINTABLESTRING の ASN.1 タイプに準拠する必要があります。TA_DMTPSUTTYPE の値が "INTEGER" である場合の最大長は LONG の値になります。この値は、リモートの TPSUT を定義する前に定義する必要があります。

制限事項　なし

T_DM_LOCAL クラスの定義

概要 T_DM_LOCAL クラスは、ローカル・ドメイン・アクセス・ポイントを定義します。ローカル・ドメイン・アクセス・ポイントは、リモート・ドメインにエクスポートされるローカル・サービスへのアクセス制御、およびリモート・ドメインからインポートされるリモート・サービスへのアクセス制御に使用します。

属性表

表 16DM_MIB(5): T_DM_LOCAL クラス定義の属性表

属性	タイプ	パーミッション	値	デフォルト値
TA_DMACCESSPOINT(r)(k)(*)	string	rw-r--r--	*string*[1..30]	N/A
TA_DMACCESSPOINTID(r)	string	rw-r--r--	*string*[1..30]	N/A
TA_DMSRVGROUP(r)	string	rw-r--r--	*string*[1..30]	N/A
TA_DMTYPE	string	rw-r--r--	"{TDOMAIN | TOPEND | SNAX | OSITP | OSITPX}"	"TDOMAIN"
TA_STATE(r)	string	rw-r--r--	GET: "VAL" SET: "{NEW | INV}"	N/A N/A
TA_DMAUDITLOG	string	rw-r--r--	*string*[1..256] (注 3)	N/A
TA_DMBLOCKTIME	short	rw-r--r--	0 <= *num* <= 32,767	TA_BLOCKTIME in T_DOMAIN (注 1)
TA_DMTLOGDEV	string	rw-r--r--	*string*[1..256] (注 3)	N/A
TA_DMTLOGNAME	string	rw-r--r--	*string*[1..30]	"DMTLOG"
TA_DMTLOGSIZE	long	rw-r--r--	1 <= *num* <= 2048	100
TA_DMMAXRAPTRAN	short	rw-r--r--	0 <= *num* <= 32,767	16
TA_DMMAXTRAN	short	rw-r--r--	0 <= *num* <= 32,767	TA_MAXGTT in T_DOMAIN (注 2)

表 16DM_MIB(5): T_DM_LOCAL **クラス定義の属性表 (続き)**

属性	タイプ	パーミッション	値	デフォルト値
TA_DMSECURITY	string	rw-r--r--	"{NONE | APP_PW | DM_PW | DM_USER_PW | CLEAR | SAFE | PRIVATE}"	"NONE"

表 16DM_MIB(5): T_DM_LOCAL **クラス定義の属性表 (続き)**

属性	タイプ	パーミッション	値	デフォルト値
TA_DMTYPE=TDOMAIN|TOPEND: の場合に設定可能な属性				
TA_DMCONNECTION_POLICY	string	rwxr--r--	"{ON_DEMAND | ON_STARTUP | INCOMING_ONLY}"	"ON_DEMAND"
TA_DMMAXRETRY	long	rwxr--r--	0 <= *num* <= MAXLONG	0
TA_DMRETRY_INTERVAL	long	rwxr--r--	0 <= *num* <= MAXLONG	60
TA_DMTYPE=TDOMAIN: の場合に設定可能な属性				
TA_DMCONNPRINCIPALNAME	string	rwxr--r--	*string*[0..511]	""
TA_DMMACHINETYPE	string	rw-r--r--	*string*[0..15]	""
TA_DMTYPE=SNAX: の場合に設定可能な属性				
TA_DMBLOB_SHM_SIZE	long	rw-r--r--	1 <= *num* <= MAXLONG	1000000

(r)— 新しいオブジェクトが作成される場合に必須です。
(k)— オブジェクトを取り出すためのキー・フィールドです。
(*)— すべての SET 操作で必須のキー・フィールドです。

注 1 T_DOMAIN クラスの TA_BLOCKTIME の現在の値。
注 2 T_DOMAIN クラスの TA_MAXGTT の現在の値。
注 3 BEA Tuxedo 8.0 以前のリリースの場合、この属性の文字列の長さは最大 78 バイトです。

属性の意味

TA_DMACCESSPOINT: *string*[1..30]

この T_DM_LOCAL エントリの名前。この Domains コンフィギュレーション内の T_DM_LOCAL および T_DM_REMOTE アクセス・ポイントの範囲内で一意なユーザ指定のローカル・ドメイン・アクセス・ポイントの識別子 (論理名) です。

TA_DMACCESSPOINTID: *string*[1..30]

リモート・ドメインへの接続を設定するときのセキュリティのため、このローカル・ドメイン・アクセス・ポイントに関連付けられている

ドメイン・ゲートウェイ・グループの識別子です。この識別子は、すべてのローカルおよびリモート・ドメイン・アクセス・ポイント間で一意でなければなりません。

TA_DMSRVGROUP: *string*[1..30]
このローカル・ドメイン・アクセス・ポイントを表すドメイン・ゲートウェイ・グループの名前 (TUXCONFIG ファイルの GROUPS セクションで指定された名前) です。ローカル・ドメイン・アクセス・ポイントとゲートウェイ・サーバ・グループは、1 対 1 の関係です。

TA_DMTYPE: "{TDOMAIN | TOPEND | SNAX | OSITP | OSITPX}"
このローカル・ドメイン・アクセス・ポイントのドメインのタイプを指定します。BEA Tuxedo ドメインの場合は "TDOMAIN"、BEA TOP END ドメインの場合は "TOPEND"、SNA ドメインの場合は "SNAX"、OSI TP 1.3 ドメインの場合は "OSITP"、OSI TP 4.0 以降のドメインの場合は "OSITPX" を指定します。ほかの属性が存在するかどうかは、この属性の値に依存します。

TA_DMTYPE="OSITPX" の設定は、BEA Tuxedo 8.0 以降のソフトウェアでのみサポートされます。

TA_STATE:

GET: "{VALid}"
GET 操作は、T_DM_LOCAL オブジェクトのコンフィギュレーション情報を検索します。以下に示す状態は、GET 要求に対する応答で返される TA_STATE 属性の意味を示します。以下に示されていない状態は返されません。

"VALid"	オブジェクトが存在します。

SET: "{NEW | INValid}"
SET 操作は、選択した T_DM_LOCAL オブジェクトのコンフィギュレーション情報を更新します。以下に示す状態は、SET 要求で設定される TA_STATE の意味を示します。以下に示されていない状態は設定できません。

"NEW"	新しいオブジェクトを作成します。この状態変更は、"INValid" 状態でのみ可能です。正常に終了すると、オブジェクトの状態は "VALid" になります。
unset	既存のオブジェクトを変更します。この組み合わせは "INValid" 状態では使用できません。正常に終了してもオブジェクトの状態は変わりません。
"INValid"	オブジェクトを削除します。この状態変更は、"VALid" 状態でのみ可能です。正常に終了すると、オブジェクトの状態は "INValid" になります。

TA_DMAUDITLOG:*string*[1..256] (BEA Tuxedo 8.0 以前では最大 78 バイト)

このローカル・ドメイン・アクセス・ポイントに対する監査ログ・ファイルの名前。

TA_DMBLOCKTIME: 0 <= *num* <= 32,767

このローカル・ドメイン・アクセス・ポイントに対するブロッキング・コールの最大待ち時間を指定します。この値は、T_DOMAIN オブジェクトの SCANUNIT パラメータの乗数です。SCANUNIT * TA_BLOCKTIME の値は、SCANUNIT 以上 32,768 秒未満でなければなりません。この属性を指定しない場合、T_DOMAIN オブジェクトに指定した TA_BLOCKTIME 属性の値がデフォルト値に設定されます。ブロッキング・タイムアウト状態は、関連する要求が失敗したことを示します。

ドメイン間トランザクションでは、トランザクション期間が TA_DMBLOCKTIME 属性の値を超過すると ブロッキング・タイムアウト状態が生成されます。つまり、ドメイン間トランザクションでは、TA_DMBLOCKTIME 属性の値が T_SERVICE オブジェクトに指定された TA_TRANTIME タイムアウト値未満の場合、またはトランザクションを開始するための tpbegin() 呼び出しで渡されたタイムアウト値未満の場合、トランザクションのタイムアウトは TA_DMBLOCKTIME 値まで減らされます。一方、ドメイン内トランザクション (単一の BEA Tuxedo ドメイン内で処理されるトランザクション) の場合は、

T_DOMAIN オブジェクトに指定された TA_BLOCKTIME 属性の値は、ドメイン内トランザクションのタイムアウトに何の影響も与えません。

TA_DMTLOGDEV: *string*[1..256] (BEA Tuxedo 8.0 以前では最大 78 バイト)
このローカル・ドメイン・アクセス・ポイントの Domains トランザクション・ログ (TLOG) を含むデバイス (raw スライス) またはファイル。TLOG は、BEA Tuxedo システムの VTOC テーブルとしてデバイスに格納されています。信頼性の観点から、デバイス (raw スライス) の使用を推奨します。

この属性を指定しない場合、このローカル・ドメイン・アクセス・ポイントに関連付けられているドメイン・ゲートウェイ・グループは要求をトランザクション・モードで処理できません。同じマシンのローカル・ドメイン・アクセス・ポイント間で同じ BEA Tuxedo ファイルシステムを共有することはできますが、各ローカル・ドメイン・アクセス・ポイントは TA_DMTLOGNAME キーワードで指定された名前のログ (TA_DMTLOGDEV 内のテーブル) を保持する必要があります。

TA_DMTLOGNAME: *string*[1..30]
このローカル・ドメイン・アクセス・ポイント用の TLOG の名前。1 つのデバイスに複数の TLOG がある場合、各 TLOG の名前は一意でなければなりません。

TA_DMTLOGSIZE: 1 <= *num* <= 2048
このローカル・ドメイン・アクセス・ポイント用の TLOG のサイズ (ページ数)。このサイズは、TA_DMTLOGDEV に指定したデバイスで使用できる領域の数によって制限されます。

TA_DMMAXRAPTRAN: 0 <= *num* <= 32,767
このローカル・ドメイン・アクセス・ポイントの 1 つのトランザクションに含めることのできるリモート・ドメイン・アクセス・ポイントの最大数。

TA_DMMAXTRAN: 0 <= *num* <= 32,767
このローカル・ドメイン・アクセス・ポイントで同時に実行できるトランザクションの最大数。この値は、T_DOMAIN:TA_MAXGTT 属性値以上でなければなりません。

TA_DMSECURITY:"{NONE|APP_PW|DM_PW|DM_USER_PW|CLEAR|SAFE|PRIVATE}"

このローカル・ドメイン・アクセス・ポイントに関連付けるドメイン・ゲートウェイに対して有効なセキュリティのタイプ。この属性は、次のいずれかに設定する必要があります。

"NONE"

セキュリティは無効になります。

"APP_PW"

この値は、TA_DMTYPE="TDOMAIN" の場合のみ有効です。アプリケーション・パスワードによるセキュリティが有効になります。

"DM_PW"

この値は、TA_DMTYPE="TDOMAIN" または "OSITPX" の場合のみ有効です。ドメイン・パスワードによるセキュリティが有効になります。

"DM_USER_PW"

この値は、TA_DMTYPE="SNAX" の場合のみ有効です。プリンシパル名の変換が有効になります。

"CLEAR"

この値は、TA_DMTYPE="TOPEND" の場合のみ有効です。ローカル・ドメインと BEA TOP END システムの間で BEA TOP END セキュリティが有効になります。ネットワーク・メッセージはプレーン・テキストで送信されます。

"SAFE"

この値は、TA_DMTYPE="TOPEND" の場合のみ有効です。ローカル・ドメインと BEA TOP END システムの間で BEA TOP END セキュリティが有効になります。ネットワーク・メッセージはチェックサムによって保護されます。

"PRIVATE"

この値は、TA_DMTYPE="TOPEND" の場合のみ有効です。ローカル・ドメインと BEA TOP END システムの間で BEA TOP END セキュリティが有効になります。ネットワーク・メッセージは暗号化されます。

TA_DMTYPE=TDOMAIN|TOPEND の場合に設定可能な属性

TA_DMCONNECTION_POLICY:"{ON_DEMAND | ON_STARTUP | INCOMING_ONLY}"

このローカル・ドメイン・アクセス・ポイントに関連付けられているドメイン・ゲートウェイがリモート・ドメインへの接続を確立するときの条件を指定します。有効な値は、"ON_DEMAND"、"ON_STARTUP"、または "INCOMING_ONLY" です。

"ON_DEMAND"

クライアントがリモート・サービスを要求したとき、または dmadmin(1) connect コマンドが実行されたときにのみ接続が試行されます。TA_DMCONNECTION_POLICY 属性のデフォルト設定は "ON_DEMAND" です。"ON_DEMAND" 接続ポリシーでは、TA_DMCONNECTION_POLICY 属性を明示的に使用できなかった以前のリリースと同じ振る舞いになります。この接続ポリシーでは、再接続は行われません。

"ON_STARTUP"

ドメイン・ゲートウェイはゲートウェイ・サーバの初期化時にリモート・ドメインへの接続を試行します。リモート・ドメインへの接続が確立された場合にのみ、そのリモート・サービス (ドメイン・ゲートウェイによって宣言されたサービス) が宣言されます。したがって、リモート・ドメインとの接続が確立されていないと、リモート・サービスは中断されます。デフォルトでは、失敗した接続が 60 秒おきに再試行されるよう設定されています。再接続の間隔は、TA_DMRETRY_INTERVAL 属性で変更できます。TA_DMMAXRETRY 属性も参照してください。

"INCOMING_ONLY"

ドメイン・ゲートウェイは起動時にリモート・ドメインへの接続を試みません。このため、リモート・サービスは最初は中断されています。ドメイン・ゲートウェイは、リモート・ドメインからの接続を受信したときに利用可能になります。リモート・サービスは、ドメイン・ゲートウェイが接続を受信したときか、dmadmin(1) connect コマンドで管理接続が確立されたときに宣言されます。接続ポリシーが "INCOMING_ONLY" の場合、再接続は行われません。

TA_DMMAXRETRY: 0 <= *num* <= MAXLONG

このローカル・ドメイン・アクセス・ポイントに関連付けられているドメイン・ゲートウェイがリモート・ドメインへの接続を試行する回

数。最小値は 0、最大値は MAXLONG (2147483647) です。MAXLONG の場合、再接続処理が無限に繰り返されるか、または接続が確立されるまで繰り返されます。接続ポリシーが "ON_STARTUP" の場合、TA_DMMAXRETRY のデフォルト設定は MAXLONG になります。この属性を 0 に設定すると、自動再接続は行われません。それ以外の接続ポリシーの場合、自動再試行は無効になります。

TA_DMMAXRETRY 属性は、接続ポリシーが "ON_STARTUP" の場合のみ有効です。

TA_DMRETRY_INTERVAL: 0 <= *num* <= MAXLONG

このローカル・ドメイン・アクセス・ポイントに関連付けられているドメイン・ゲートウェイがリモート・ドメインへの接続を自動的に試行する間隔 (単位は秒)。最小値は 0、最大値は MAXLONG (2147483647) です。デフォルト値は 60 です。TA_DMMAXRETRY が 0 に設定されている場合、TA_DMRETRY_INTERVAL は設定できません。

この属性は、TA_DMCONNECTION_POLICY 属性が "ON_STARTUP" に設定されている場合にのみ有効です。それ以外の接続ポリシーの場合、自動再試行は無効になります。

TA_DMTYPE=TDOMAIN の場合に設定可能な属性

TA_DMCONNPRINCIPALNAME: string[0..511]

接続プリンシパル名の識別子。これは、ローカル・ドメイン・アクセス・ポイントに関連付けられているドメイン・ゲートウェイがリモート・ドメインに接続するときに、そのドメイン・ゲートウェイの ID を検証するためのプリンシパル名です。この属性は、BEA Tuxedo 7.1 以降のソフトウェアが実行されている TDOMAIN タイプのドメイン・ゲートウェイにのみ適用されます。

TA_DMCONNPRINCIPALNAME 属性には最大 511 文字を指定できます (最後の NULL 文字を除く)。この属性を指定しない場合は、ローカル・ドメイン・アクセス・ポイントの TA_DMACCESSPOINTID 文字列がデフォルト値になります。

デフォルトの認証プラグインで、このローカル・ドメイン・アクセス・ポイントの TA_DMCONNPRINCIPALNAME 属性に値を割り当てる場合、その値は、このローカル・ドメイン・アクセス・ポイントの TA_DMACCESSPOINTID 属性の値と同じでなければなりません。これら

の値が一致しないと、ローカル・ドメイン・ゲートウェイ・プロセスが起動せず、次の userlog(3c) メッセージが生成されます。ERROR: 証明書を取得できません。

TA_DMMACHINETYPE: *string*[0..15]
ドメインをグループ化して、このローカル・ドメイン・アクセス・ポイントに関連付けられているマシンとリモート・ドメイン・アクセス・ポイントに関連付けられているマシン間のメッセージの符号化と復号化を省略するために使用します。この属性は、TDOMAIN タイプのドメイン・ゲートウェイにのみ適用されます。

TA_DMMACHINETYPE を指定しない場合、デフォルトで符号化または復号化が実行されます。TA_DMMACHINETYPE フィールドに設定した値が接続のための T_DM_LOCAL クラスと T_DM_REMOTE クラスで共通している場合、データの符号化と復号化が省略されます。TA_DMMACHINETYPE には、15 文字までの任意の文字列値を指定できます。この値は比較のためだけに使用します。

TA_DMTYPE=SNAX の場合に設定可能な属性

TA_DMBLOB_SHM_SIZE: 1 <= *num* <= MAXLONG
この SNAX ローカル・ドメイン・アクセス・ポイント固有のバイナリ・ラージ・オブジェクトのログ情報を格納するために割り当てられた共用メモリを指定します。この属性は、ローカル・ドメイン・アクセス・ポイントおよび SNAX タイプのドメイン・ゲートウェイにのみ適用されます。

制限事項 TA_DMLACCESSPOINT 属性で指定したローカル・ドメイン・アクセス・ポイントをサポートするドメイン・ゲートウェイ管理 (GWADM) サーバがアクティブである場合、SET を実行して TA_STATE を INValid にしたり、TA_DMACCESSPOINTID、TA_DMSRVGROUP、TA_DMTYPE、TA_DMTLOGDEV、TA_DMTLOGNAME、TA_DMTLOGSIZE、TA_DMMAXRAPTRAN、TA_DMMAXTRAN、または TA_DMMACHINETYPE 属性を更新することはできません。

T_DM_OSITP クラスの定義

概要 T_DM_OSITP クラスは、特定のローカルまたはリモート・ドメイン・アクセス・ポイントに対する OSI TP 1.3 プロトコル関連のコンフィギュレーション情報を定義します。

属性表

表 17DM_MIB(5): T_DM_OSITP **クラス定義の属性表**

属性	タイプ	パーミッション	値	デフォルト値
TA_DMACCESSPOINT(r)(k)(*)	string	rw-r--r--	*string*[1..30]	N/A
TA_STATE(r)	string	rw-r--r--	GET: "VAL" SET: "{NEW | INV}"	N/A N/A
TA_DMAPT(r)	string	rw-r--r--	*string*[1..78]	N/A
TA_DMAEQ(r)	string	rw-r--r--	*string*[1..78]	N/A
TA_DMNWDEVICE	string	rw-r--r--	*string*[1..78]	N/A
TA_DMACN	string	rw-r--r--	"{XATMI | UDT}"	"XATMI"
TA_DMAPID	short	rw-r--r--	0 <= *num* <= 32767	N/A
TA_DMAEID	short	rw-r--r--	0 <= *num* <= 32767	N/A
TA_DMURCH	string	rw-r--r--	*string*[0..30]	N/A
TA_DMMAXLISTENINGEP	short	rw-r--r--	1 <= *num* <= 32767	3
TA_DMXATMIENCODING	string	rw-r--r--	"{CAE | PRELIMINARY | OLTP_TM2200}"	"CAE"

(r) - 新しいオブジェクトが作成される場合に必須です。
(k) - オブジェクトを取り出すためのキー・フィールドです。
(*) - すべての SET 操作で必須のキー・フィールドです。

属性の意味 TA_DMACCESSPOINT: *string*[1..30]
このエントリがプロトコル固有のコンフィギュレーション情報を提供するローカルまたはリモート・ドメイン・アクセス・ポイントの名

前。このフィールドは、ドメイン・アクセス・ポイントのプロトコルに依存しないコンフィギュレーションを定義する T_DM_LOCAL または T_DM_REMOTE エントリで指定したドメイン・アクセス・ポイント名と一致します。

TA_STATE:

GET: "{VALid}"
GET 操作は、T_DM_OSITP オブジェクトのコンフィギュレーション情報を検索します。以下に示す状態は、GET 要求に対する応答で返される TA_STATE 属性の意味を示します。以下に示されていない状態は返されません。

"VALid"	オブジェクトが存在します。

SET: "{NEW | INValid}"
SET 操作は、選択した T_DM_OSITP オブジェクトのコンフィギュレーション情報を更新します。以下に示す状態は、SET 要求の TA_STATE の意味を示します。以下に示されていない状態は設定できません。

"NEW"	新しいオブジェクトを作成します。この状態変更は、"INValid" 状態でのみ可能です。正常に終了すると、オブジェクトの状態は "VALid" になります。
unset	既存のオブジェクトを変更します。この組み合わせは "INValid" 状態では使用できません。正常に終了してもオブジェクトの状態は変わりません。
"INValid"	オブジェクトを削除します。この状態変更は、"VALid" 状態でのみ可能です。正常に終了すると、オブジェクトの状態は "INValid" になります。

TA_DMAPT: *string*[1..78]
ローカルまたはリモート・ドメイン・アクセス・ポイントのアプリケーション・プロセス・タイトル (オブジェクト識別子形式)。

TA_DMAEQ: *string*[1..78]
ローカルまたはリモート・ドメイン・アクセス・ポイントのアプリケーション・エンティティ修飾子 (整数形式)。

TA_DMNWDEVICE: *string*[1..78]
このローカル・ドメイン・アクセス・ポイントで使用するネットワーク・デバイスを指定します。この属性は、ローカル・ドメイン・アクセス・ポイントを定義する場合にのみ有効です。リモート・ドメイン・アクセス・ポイントでは無視されます。

TA_DMACN: "{XATMI | UDT}"
このローカルまたはリモート・ドメイン・アクセス・ポイントで使用するアプリケーション・コンテキストの名前。リモート・ドメイン・アクセス・ポイントからのアプリケーション・コンテキスト名が存在する場合は、これをリモート・ドメイン・アクセス・ポイントへのダイアログを確立する際に使用します。存在しない場合は、ローカル・ドメイン・アクセス・ポイントからのアプリケーション・コンテキスト名が使用されます。"XATMI" では、X/Open 定義の XATMI アプリケーション・サービス・エレメント (ASE) および符号化を使用するかどうかを選択します。"UDT" では、ISO/IEC 10026-5 ユーザ・データ転送の符号化を使用するかどうかを選択します。

TA_DMAPID: 0 <= *num* <= 32767
このローカルまたはリモート・ドメイン・アクセス・ポイントで使用するアプリケーション・プロセスの呼び出し識別子を定義する属性 (オプション)。

TA_DMAEID: 0 <= *num* <= 32767
このローカルまたはリモート・ドメイン・アクセス・ポイントで使用するアプリケーション・エンティティの呼び出し識別子を定義する属性 (オプション)。

TA_DMURCH: *string*[0..30]
このローカル・ドメイン・アクセス・ポイントに対する OSI TP リカバリ・コンテキスト・ハンドルのユーザ部分を指定します。OSI TP リカバリ・コンテキスト・ハンドルは、通信回線またはシステムに障害

が発生した後に、OSI TP プロバイダによる分散トランザクションの回復処理で必要になる場合があります。

この属性は、ローカル・ドメイン・アクセス・ポイントを定義する場合にのみ有効です。リモート・ドメイン・アクセス・ポイントでは無視されます。

TA_DMMAXLISTENINGEP: 0 <= *num* <= 32767

このローカル・ドメイン・アクセス・ポイントに対する着信用 OSI TP ダイアログを待機している端点の数を指定します。この属性は、ローカル・ドメイン・アクセス・ポイントを定義する場合にのみ有効です。リモート・ドメイン・アクセス・ポイントでは無視されます。

TA_DMXATMIENCODING: "{CAE | PRELIMINARY | OLTP_TM2200}"

リモート・システムとの通信に使用する XATMI プロトコルのバージョンを指定します。この属性は、リモート・ドメイン・アクセス・ポイントの記述でのみ有効です。次の値を指定できます。

"CAE" (デフォルト)

"PRELIMINARY" (Unisys MCP OLTP システムで使用)

"OLTP_TM2200" (Unisys TM 2200 システムで使用)

制限事項 以下の場合は、このクラスのインスタンスを削除または更新することはできません。

- このクラスのインスタンスがローカル・ドメイン・アクセス・ポイントに対応し、ローカル・アクセス・ポイントに関連付けられたドメイン・ゲートウェイ・グループがアクティブである場合。
- このクラスのインスタンスがリモート・ドメイン・アクセス・ポイントに対応し、リモート・アクセス・ポイントに関連付けられたドメイン・ゲートウェイ・グループがアクティブである場合。

このクラスのインスタンスを追加または更新する SET 操作の実行時には、TA_DMACCESSPOINT 属性に指定した特定のローカルまたはリモート・ドメイン・アクセス・ポイントが、T_DM_LOCAL クラスまたは T_DM_REMOTE クラスに存在している必要があります。ドメイン・アクセス・ポイントが存在しない場合、TA_DMACCESSPOINT 属性に対して "not defined" エラーが返され、操作は失敗します。

T_DM_OSITPX クラスの定義

概要 T_DM_OSITPX クラスは、特定のローカルまたはリモート・ドメイン・アクセス・ポイントに対する OSI TP 4.0 以降のプロトコル関連のコンフィギュレーション情報を定義します。T_DM_OSITPX クラスは、BEA Tuxedo 8.0 以降のソフトウェアでのみサポートされます。

属性表

表 18DM_MIB(5): T_DM_OSITPX クラス定義の属性表

属性	タイプ	パーミッション	値	デフォルト値
TA_DMACCESSPOINT(r)(k)(*)	string	rw-r--r--	*string*[1..30]	N/A
TA_STATE(r)	string	rw-r--r--	GET: "VAL" SET: "{NEW | INV}"	N/A N/A
TA_DMAET(r)	string	rw-r--r--	*string*[1..78]	N/A
TA_DMNWADDR(r)	string	rw-r--r--	*string*[1..631]	N/A
TA_DMTSEL	string	rw-r--r--	*string*[1..66]	N/A
TA_DMDNSRESOLUTION	string	rw-r--r--	"{STARTUP | RUNTIME}"	"STARTUP"
TA_DMPSEL	short	rw-r--r--	*string*[1..10]	""
TA_DMSSEL	short	rw-r--r--	*string*[1..34]	""
TA_DMTAILORPATH	short	rw-r--r--	*string*[1..78]	""
TA_DMXATMIENCODING	string	rw-r--r--	"{CAE | PRELIMINARY | OLTP_TM2200 | NATIVE_A_SERIES}"	"CAE"
TA_DMEXTENSIONS	short	rw-r--r--	*string*[1..78]	""
TA_DMOPTIONS	short	rw-r--r--	"{SECURITY_SUPPORTED}"	""

(r)― 新しいオブジェクトが作成される場合に必須です。
(k)― オブジェクトを取り出すためのキー・フィールドです。
(*)― すべての SET 操作で必須のキー・フィールドです。

属性の意味 TA_DMACCESSPOINT: *string*[1..30]

このエントリがプロトコル固有のコンフィギュレーション情報を提供するローカルまたはリモート・ドメイン・アクセス・ポイントの名前。このフィールドは、ドメイン・アクセス・ポイントのプロトコルに依存しないコンフィギュレーションを定義する T_DM_LOCAL または T_DM_REMOTE エントリで指定したドメイン・アクセス・ポイント名と一致します。

TA_STATE:

GET: "{VALid}"

GET 操作は、T_DM_OSITPX オブジェクトのコンフィギュレーション情報を検索します。以下に示す状態は、GET 要求に対する応答で返される TA_STATE 属性の意味を示します。以下に示されていない状態は返されません。

"VALid"	オブジェクトが存在します。

SET: "{NEW | INValid}"

SET 操作は、選択した T_DM_OSITPX オブジェクトのコンフィギュレーション情報を更新します。以下に示す状態は、SET 要求の TA_STATE の意味を示します。以下に示されていない状態は設定できません。

"NEW"	新しい T_DM_OSITPX オブジェクトを作成します。この状態変更は、"INValid" 状態でのみ可能です。正常に終了すると、オブジェクトの状態は "VALid" になります。
unset	既存の T_DM_OSITPX オブジェクトを変更します。この組み合わせは "INValid" 状態では使用できません。正常に終了してもオブジェクトの状態は変わりません。
"INValid"	T_DM_OSITPX オブジェクトを削除します。この状態変更は、"VALid" 状態でのみ可能です。正常に終了すると、オブジェクトの状態は "INValid" になります。

TA_DMAET: *string*[1..78]

ローカルまたはリモート・ドメイン・アクセス・ポイントのアプリケーション・エンティティ・タイトル。このアドレスは、OSI TP ネットワーク内のすべてのホスト通信において一意である必要があり、リモート (OLTP) ノード上のローカル AE タイトルに一致します。

この属性の値には、オブジェクト識別子としてアプリケーション・プロセス・タイトルが含まれます。オブジェクト識別子の後には、"{*object identifier*},{*integer qualifier*}" のように、整数値としてアプリケーション・エンティティ修飾子が続きます。中かっこは構文の一部で、引用符の内側に入れる必要があります。

TA_DMNWADDR: *string*[1..631]

このアクセス・ポイントで使用するネットワーク・アドレスのセミコロン区切りのリスト。ネットワーク・アドレスは、TCP/IP ネットワークを使用している場合は IP アドレス、そうでない場合は DNS 名になります。ネットワーク・アドレスの形式は次のいずれかになります。

"*#.#.#.#*:*port_number*" IP アドレス
"*//hostname*:*port_number*" DNS 名
"*//hostname*:*port_number*;*//hostname*:*port_number*; ..."

port_number 構成要素が指定されていない場合は、デフォルト・ポートの 102 が使用されます。

ローカル・ドメイン・アクセス・ポイントの場合、この属性の値には接続要求をリスンするための待機用アドレスを 8 個までセミコロンで区切って指定できます。リモート・ドメイン・アクセス・ポイントの場合、この属性の値には目的のドメインの優先アドレスを指定し、その後に最大 7 個の代替アドレスを優先順位の高い順に指定します。代替アドレスは、最初のアドレスが使用できない場合に使用されます。

TA_DMTSEL: *string*[1..66]

このローカルまたはリモート・ドメイン・アクセス・ポイントで使用する転送サービス・アクセス・ポイントのアドレス。1 ～ 32 の ASCII 非制御文字 (16 進数では 20 から 7E)、先頭に 0x が付く 1 ～ 32 の 16 進数オクテット、または "NONE"― (NULL 文字列) のいずれかを指定します。

TA_DMDNRESOLUTION: "{STARTUP | RUNTIME}"

このローカル・ドメイン・アクセス・ポイントに関連付けられたドメイン・ゲートウェイ (GWOSITP) 用に、TA_DMNWADDR 属性で定義したネットワーク・アドレスの DNS 名を解決するタイミングを指定します。この属性を "STARTUP" (デフォルト) に設定すると、ゲートウェイの起動時にホスト名が実 IP アドレスに解決されます。この属性を "RUNTIME" に設定すると、ゲートウェイの実行時にホスト名が実 IP アドレスに解決されます。

この属性は、ローカル・ドメイン・アクセス・ポイントを定義する場合にのみ有効です。リモート・ドメイン・アクセス・ポイントでは無視されます。リモート・ドメイン・アクセス・ポイントのインスタンスに対する GET 呼び出しでは、この属性は NULL 文字列に設定されます。

TA_DMPSEL: *string*[1..10]
: このローカルまたはリモート・ドメイン・アクセス・ポイントで使用するプレゼンテーション・サービス・アクセス・ポイントのアドレス。1 ～ 4 の ASCII 非制御文字 (16 進数では 20 から 7E)、先頭に 0x が付く 1 ～ 4 の 16 進数オクテット、または "NONE" (デフォルト) のいずれかを指定します。

TA_DMSSEL: *string*[1..34]
: このローカルまたはリモート・ドメイン・アクセス・ポイントで使用するセッション・サービス・アクセス・ポイントのアドレス。1 ～ 16 の ASCII 非制御文字 (16 進数では 20 から 7E)、先頭に 0x が付く 1 ～ 16 の 16 進数オクテット、または "NONE" (デフォルト) のいずれかを指定します。

TA_DMTAILORPATH: *string*[1..78]
: このローカル・ドメイン・アクセス・ポイントの OSI TP スタックの調整に使用するオプションの OSI TP 調整ファイルの絶対パス名を示します。二重引用符が必要です。値を指定しない場合、および NULL 文字列を設定した場合、OSI TP スタックはデフォルトの調整パラメータで実行されます。

  この属性は、ローカル・ドメイン・アクセス・ポイントを定義する場合にのみ有効です。リモート・ドメイン・アクセス・ポイントでは無視されます。

TA_DMXATMIENCODING: "{CAE | PRELIMINARY | OLTP_TM2200 | NATIVE_A_SERIES}"
: リモート・システムとの通信に使用する XATMI プロトコルのバージョンを指定します。この属性は、リモート・ドメイン・アクセス・ポイントの記述でのみ有効です。次の値を指定できます。

"CAE" (デフォルト)

"PRELIMINARY" (Unisys MCP OLTP システムで使用)

"OLTP_TM2200" (Unisys TM 2200 システムで使用)

"NATIVE_A_SERIES" (この符号化タイプをサポートする Unisys MCP OLTP
システムで使用)

TA_DMEXTENSIONS: *string*[1..78]
このリモート・ドメイン・アクセス・ポイントに関連付けられたリモート・ドメインの操作を制御します。有効な値としては、セミコロン (;) 区切りで "ONLINE=N/Y" (デフォルトは Y) と "RdomAssocRetry=*nn*" を含めます。*nn* には、オンライン・リモート・ドメインへの再接続の間隔を秒単位で指定します。RdomAssocRetry 調整パラメータが存在する場合は、その値がこの属性のデフォルトになります。RdomAssocRetry が存在せず、*nn* が指定されていない場合のデフォルトは 60 秒です。

TA_DMOPTIONS: "{SECURITY_SUPPORTED}"
このリモート・ドメイン・アクセス・ポイントのオプション・パラメータを示します。"SECURITY_SUPPORTED" は、このリモート・ドメイン・アクセス・ポイントに関連付けられたリモート・ドメインが OSITP セキュリティの拡張をサポートすることを示します。この属性は下位互換性があり、リモート・ドメイン・アクセス・ポイントの記述でのみ有効です。

制限事項 以下の場合は、このクラスのインスタンスを削除または更新することはできません。

- このクラスのインスタンスがローカル・ドメイン・アクセス・ポイントに対応し、ローカル・アクセス・ポイントに関連付けられたドメイン・ゲートウェイ・グループがアクティブである場合。
- このクラスのインスタンスがリモート・ドメイン・アクセス・ポイントに対応し、リモート・アクセス・ポイントに関連付けられたドメイン・ゲートウェイ・グループがアクティブである場合。

このクラスのインスタンスを追加または更新する SET 操作の実行時には、TA_DMACCESSPOINT 属性に指定した特定のローカルまたはリモート・ドメイン・アクセス・ポイントが、T_DM_LOCAL クラスまたは T_DM_REMOTE クラスに存在している必要があります。ドメイン・アクセス・ポイントが存在しない場合、TA_DMACCESSPOINT 属性に対して "not defined" エラーが返され、操作は失敗します。

T_DM_PASSWORD クラスの定義

概要 T_DM_PASSWORDS クラスは、TDOMAIN タイプのアクセス・ポイントを介したドメイン間認証のコンフィギュレーション情報を表します。

属性表

表 19DM_MIB(5): T_DM_PASSWORD クラス定義の属性表

属性	タイプ	パーミッション	値	デフォルト値
TA_DMLACCESSPOINT(r)(k)(*)	string	rw-r--r--	*string*[1..30]	N/A
TA_DMRACCESSPOINT(r)(k)(*)	string	rw-r--r--	*string*[1..30]	N/A
TA_DMLPWD(r)	string	-w-------	*string*[1..30]	N/A
TA_DMRPWD(r)	string	-w-------	*string*[1..30]	N/A
TA_STATE(r)	string	rw-r--r--	GET: "VAL" SET: "{NEW | INV | REC}"	N/A N/A

(r)— 新しいオブジェクトが作成される場合に必須です。
(k)— オブジェクトを取り出すためのキー・フィールドです。
(*)— すべての SET 操作で必須のキー・フィールドです。

属性の意味

TA_DMLACCESSPOINT: *string*[1..30]
パスワードを適用するローカル・ドメイン・アクセス・ポイントの名前。

TA_DMRACCESSPOINT: *string*[1..30]
パスワードを適用するリモート・ドメイン・アクセス・ポイントの名前。

TA_DMLPWD: *string*[1..30]
ローカル・ドメイン・アクセス・ポイント (TA_DMLACCESSPOINT で識別) とリモート・ドメイン・アクセス・ポイント (TA_DMRACCESSPOINT で識別) の接続を認証するためのローカル・パスワード。

TA_DMRPWD: *string*[1..30]

ローカル・ドメイン・アクセス・ポイント (TA_DMLACCESSPOINT で識別) とリモート・ドメイン・アクセス・ポイント (TA_DMRACCESSPOINT で識別) の接続を認証するためのリモート・パスワード。

TA_STATE:

GET:"{VALid}"

GET 操作は、選択した T_DM_PASSWORD オブジェクトのコンフィギュレーション情報を検索します。以下に示す状態は、GET 要求に対する応答で返される TA_STATE 属性の意味を示します。以下に示されていない状態は返されません。

"VALid"	オブジェクトが存在します。

SET: "{NEW | INValid | RECrypt}"

SET 操作は、選択した T_DM_PASSWORD オブジェクトのコンフィギュレーション情報を更新します。以下に示す状態は、SET 要求の TA_STATE の意味を示します。以下に示されていない状態は設定できません。

"NEW"	新しいオブジェクトを作成します。状態変更は、"INValid" 状態でのみ可能です。正常に終了すると、オブジェクトの状態は "VALid" になります。
unset	既存のオブジェクトを変更します。この組み合わせは "INValid" 状態では使用できません。
"INValid"	オブジェクトを削除します。状態変更は、"VALid" 状態でのみ可能です。正常に終了すると、オブジェクトの状態は "INValid" になります。
"RECrypt"	暗号化キーを使用してすべてのパスワードを再暗号化します。T_DM_PASSWORD クラスおよび T_DM_TOPEND クラスのすべてのパスワード・インスタンスに適用されます。

制限事項　ドメイン・ゲートウェイ管理サーバ (GWADM) の実行中は、パスワードを再暗号化 (TA_STATE から "RECrypt" に SET) することはできません。

T_DM_PRINCIPAL_MAP クラスの定義

概要 T_DM_PRINCIPAL_MAP クラスは、タイプ SNAX のアクセス・ポイントを介してプリンシパル名を外部プリンシパル名との間でマッピングするためのコンフィギュレーション情報を表します。

属性表

表 20DM_MIB(5): T_DM_PRINCIPAL_MAP クラス定義の属性表

属性	タイプ	パーミッション	値	デフォルト値
TA_DMLACCESSPOINT(r)(k)(*)	string	rw-r--r--	*string*[1..30]	N/A
TA_DMRACCESSPOINT(r)(k)(*)	string	rw-r--r--	*string*[1..30]	N/A
TA_DMPRINNAME(r)(k)(*)	string	rw-------	*string*[1..30]	N/A
TA_DMRPRINNAME(r)(k)(*)	string	rw-------	*string*[1..30]	N/A
TA_DMDIRECTION(k)	string	rw-r-----	"{IN | OUT | BOTH}"	"BOTH"
TA_STATE(r)	string	rw-r--r--	GET:"VAL" SET:"{NEW | INV}"	N/A N/A

(r)― 新しいオブジェクトが作成される場合に必須です。
(k)― オブジェクトを取り出すためのキー・フィールドです。
(*)― すべての SET 操作で必須のキー・フィールドです。

属性の意味

TA_DMLACCESSPOINT: *string*[1..30]
プリンシパル・マッピングを適用するローカル・ドメイン・アクセス・ポイント。

TA_DMRACCESSPOINT: *string*[1..30]
プリンシパル・マッピングを適用するリモート・ドメイン・アクセス・ポイント。

TA_DMPRINNAME: *string*[1..30]
プリンシパル・マッピングにおけるローカル・プリンシパル名。

TA_DMRPRINNAME: *string*[1..30]
プリンシパル・マッピングにおけるリモート・プリンシパル名。

TA_DMDIRECTION: "{IN | OUT | BOTH}"

プリンシパル・マッピングを適用する方向。

"IN"

指定したリモート・ドメイン・アクセス・ポイントとローカル・ドメイン・アクセス・ポイントを経由した BEA Tuxedo ドメインへの INcoming であることを示します。

"OUT"

指定したローカル・ドメイン・アクセス・ポイントとリモート・ドメイン・アクセス・ポイントを経由した BEA Tuxedo ドメインへの OUTgoing であることを示します。

"BOTH"

INcoming と OUTgoing の両方に適用されます。

TA_STATE:

GET:"{VALid}"

GET 操作は、選択した T_DM_PRINCIPAL エントリのコンフィギュレーション情報を検索します。以下に示す状態は、GET 要求に対する応答で返される TA_STATE 属性の意味を示します。以下に示されていない状態は返されません。

"VALid"	オブジェクトが存在します。

SET:"{NEW | INValid}"

SET 操作は、選択した T_DM_PRINCIPAL エントリのコンフィギュレーション情報を更新します。以下に示す状態は、SET 要求の TA_STATE の意味を示します。以下に示されていない状態は設定できません。

"NEW"	新しいオブジェクトを作成します。状態変更は、"INValid" 状態でのみ可能です。正常に終了すると、オブジェクトの状態は "VALid" になります。
unset	既存のオブジェクトを変更します。この組み合わせは "INValid" 状態では使用できません。

"INValid"	オブジェクトを削除します。状態変更は、"VALid" 状態でのみ可能です。正常に終了すると、オブジェクトの状態は "INValid" になります。

制限事項　BEA Tuxedo リリース 7.1 以降では、T_DM_PRINCIPAL_MAP クラスは SNAX ドメイン・ゲートウェイ・タイプにのみ適用されます。

T_DM_REMOTE クラスの定義

概要 T_DM_REMOTE クラスは、リモート・ドメイン・アクセス・ポイントのコンフィギュレーション情報を表します。1 つまたは複数のローカル・ドメイン・アクセス・ポイントを通してエクスポートされるローカル・リソースは、リモート・ドメイン・アクセス・ポイントを通してリモート・ドメインにアクセスできます。同様に、リモート・リソースはリモート・ドメイン・アクセス・ポイントを通してリモート・ドメインからインポートされます。

属性表

表 21DM_MIB(5): T_DM_REMOTE **クラス定義の属性表**

属性	タイプ	パーミッション	値	デフォルト値
TA_DMACCESSPOINT(r)(k)(*)	string	rw-r--r--	*string*[1..30]	N/A
TA_DMACCESSPOINTID(r)	string	rw-r--r--	*string*[1..30]	N/A
TA_DMTYPE(k)	string	rw-r--r--	"{TDOMAIN | TOPEND | SNAX | OSITP | OSITPX}"	"TDOMAIN"
TA_STATE(r)	string	rw-r--r--	GET: "VAL" SET: "{NEW | INV}"	N/A N/A
TA_DMPRIORITY_TYPE	string	rw-r--r--	"{LOCAL_RELATIVE | LOCAL_ABSOLUTE | GLOBAL}"	"LOCAL_ RELATIVE"
TA_DMINPRIORITY	string	rw-r--r--	-99 <= num <= 100	0 or 50
TA_DMTYPE=TDOMAIN|OSITPX: の場合に設定可能な属性				
TA_DMACLPOLICY	string	rwxr--r--	"{LOCAL | GLOBAL}"	"LOCAL"
TA_DMLOCALPRINCIPALNAME	string	rwxr--r--	*string*[0..511]	""
TA_DMTYPE=TDOMAIN: の場合に設定可能な属性				
TA_DMCONNPRINCIPALNAME	string	rwxr--r--	*string*[0..511]	""
TA_DMCREDENTIALPOLICY	string	rwxr--r--	"{LOCAL | GLOBAL}"	"LOCAL"
TA_DMMACHINETYPE	string	rw-r--r--	*string*[0..15]	""

表 21DM_MIB(5): T_DM_REMOTE **クラス定義の属性表 (続き)**

属性	タイプ	パーミッション	値	デフォルト値
TA_DMTYPE=SNAX|OSITPX: の場合に設定可能な属性				
TA_DMCODEPAGE	string	rw-r--r--	*string*[1..20]	N/A

(r)― 新しいオブジェクトが作成される場合に必須です。
(k)― オブジェクトを取り出すためのキー・フィールドです。
(*)― すべての SET 操作で必須のキー・フィールドです。

属性の意味

TA_DMACCESSPOINT: *string*[1..30]

この T_DM_REMOTE エントリの名前。この Domains コンフィギュレーション内の T_DM_REMOTE および T_DM_REMOTE アクセス・ポイントの範囲内で一意なユーザ指定のリモート・ドメイン・アクセス・ポイントの識別子 (論理名) です。

TA_DMACCESSPOINTID: *string*[1..30]

リモート・ドメインへの接続を設定するときのセキュリティのため、このリモート・ドメイン・アクセス・ポイントに関連付けられているリモート・ドメインの識別子。この識別子は、すべてのローカルおよびリモート・ドメイン・アクセス・ポイント間で一意でなければなりません。

TA_DMTYPE: "{TDOMAIN | TOPEND | SNAX | OSITP | OSITPX}"

このリモート・ドメイン・アクセス・ポイントのドメインのタイプを指定します。BEA Tuxedo ドメインの場合は "TDOMAIN"、BEA TOP END ドメインの場合は "TOPEND"、SNA ドメインの場合は "SNAX"、OSI TP 1.3 ドメインの場合は "OSITP"、OSI TP 4.0 以降のドメインの場合は "OSITPX" を指定します。ほかの属性が存在するかどうかは、この属性の値に依存します。

TA_DMTYPE="OSITPX" の設定は、BEA Tuxedo 8.0 以降のソフトウェアでのみサポートされます。

TA_STATE:

GET: "{VALid}"

GET 操作は、T_DM_REMOTE オブジェクトのコンフィギュレーション情報を検索します。以下に示す状態は、GET 要求に対する応答で返される TA_STATE 属性の意味を示します。以下に示されていない状態は返されません。

"VALid"	オブジェクトが存在します。

SET: "{NEW | INValid}"

SET 操作は、選択した T_DM_REMOTE オブジェクトのコンフィギュレーション情報を更新します。以下に示す状態は、SET 要求の TA_STATE の意味を示します。以下に示されていない状態は設定できません。

"NEW"	新しいオブジェクトを作成します。
unset	既存のオブジェクトを変更します。この組み合わせは "INValid" 状態では使用できません。正常に終了してもオブジェクトの状態は変わりません。
"INValid"	オブジェクトを削除します。

TA_DMPRIORITY_TYPE = "{LOCAL_RELATIVE | LOCAL_ABSOLUTE | GLOBAL}"

TA_DMINPRIORITY = -99 <= *num* <= 100

TA_DMPRIORITY_TYPE パラメータと TA_DMINPRIORITY 属性では、このリモート・ドメイン・アクセス・ポイントのメッセージの優先順位に関する処理を指定します。これらの属性は、BEA Tuxedo 8.0 以降のソフトウェアでサポートされます。

TA_DMPRIORITY_TYPE 属性の場合、"LOCAL_RELATIVE" と "LOCAL_ABSOLUTE" はすべてのリモート・ドメイン・タイプに対して有効ですが、GLOBAL は TDOMAIN のリモート・ドメイン・タイプに対してのみ有効です。TA_DMPRIORITY_TYPE 属性を設定しない場合、デフォルトは LOCAL_RELATIVE です。

TA_DMPRIORITY_TYPE="LOCAL_RELATIVE" は、tpsprio 呼び出しなどによるリモート・ドメイン・アクセス・ポイントからの要求の優先順位がローカル・ドメインによって使用されないことを意味します。代わりに、リモート・ドメイン・アクセス・ポイントから受信する要求の優先順位は TA_DMINPRIORITY の値を基準に設定されます。この値は -99 (最低の優先順位) ～ +99 (最高の優先順位) です。デフォルトは 0 です。TA_DMINPRIORITY の設定によって、サービスのデフォルトの優先順位は、設定値の符号に応じて最大 100、最小 1 まで増減します。最高の優先順位は 100 です。リモート・ドメイン・アクセス・ポイントへの要求の場合、要求に関連付けられている優先順位も一緒にリモート・ドメイン・アクセス・ポイントに送信されます。

TA_DMPRIORITY_TYPE="LOCAL_ABSOLUTE" は、このリモート・ドメイン・アクセス・ポイントからの要求の優先順位がローカル・ドメインによって使用されないことを意味します。代わりに、リモート・ドメイン・アクセス・ポイントから受信する要求の優先順位は TA_DMINPRIORITY の値を基準に設定されます。この値は 1 (最低の優先順位) ～ 100 (最高の優先順位) です。デフォルトは 50 です。TA_DMINPRIORITY の設定によって、サービスのデフォルトの優先順位は、設定値の符号に応じて最大 100、最小 1 まで増減します。最高の優先順位は 100 です。リモート・ドメイン・アクセス・ポイントへの要求の場合、要求に関連付けられている優先順位も一緒にリモート・ドメイン・アクセス・ポイントに送信されます。

TA_DMPRIORITY_TYPE="GLOBAL" は、このリモート・ドメイン・アクセス・ポイントからの要求の優先順位がローカル・ドメインによって調整されることを意味します。リモート・ドメイン・アクセス・ポイントから受信する要求の優先順位は TA_DMINPRIORITY の値を基準に調整されます。この値は -99 (最低の優先順位) ～ +99 (最高の優先順位) です。デフォルトは 0 です。TA_DMINPRIORITY を設定した場合、受信した要求に関連付けられている優先順位は TA_DMINPRIORITY の値に加算され、その要求の優先順位の絶対値が設定されます。TA_DMINPRIORITY を設定しない場合、受信する要求の優先順位がそのままローカル・ドメインによって使用されます。リモート・ドメイン・アクセス・ポイントへの要求の場合、要求に関連付けられている優先順位も一緒にリモート・ドメイン・アクセス・ポイントに送信されます。

TA_DMTYPE=TDOMAIN|OSITPX の場合に設定可能な属性

TA_DMACLPOLICY: {LOCAL | GLOBAL}

このリモート・ドメイン・アクセス・ポイント用のアクセス制御リスト (ACL) 方針。この属性は、BEA Tuxedo 7.1 以降のソフトウェアが実行されている TDOMAIN タイプのドメイン・ゲートウェイ、および BEA Tuxedo 8.0 以降が実行されている OSITPX タイプのドメイン・ゲートウェイにのみに適用されます。

LOCAL を指定すると、ローカル・ドメインは、リモート・ドメインから受け取ったサービス要求のクリデンシャル (ID) を、このリモート・ドメイン・アクセス・ポイントに対する TA_DMLOCALPRINCIPALNAME 属性で指定されたプリンシパル名に置き換えます。GLOBAL を指定すると、ローカル・ドメインは、リモート・サービス要求で受け取ったクリデンシャルを置き換えません。リモート・サービス要求でクリデンシャルを受け取らなかった場合、ローカル・ドメインはそのサービス要求をそのままの状態でローカル・サービスに転送します (通常これは失敗する)。この属性を指定しない場合、デフォルトは LOCAL です。

TA_DMACLPOLICY 属性は、ローカル・ドメインがリモート・ドメインから受信したサービス要求のクリデンシャルを TA_DMLOCALPRINCIPALNAME 属性に指定されているプリンシパル名に置き換えるかどうかを制御します。TA_DMCREDENTIALPOLICY はこの属性に関連する属性で、ローカル・ドメインがリモート・ドメインにローカル・サービス要求を送信する前にその要求からクリデンシャルを削除するかどうかを制御します。

TA_DMLOCALPRINCIPALNAME: string[0..511]

ローカル・プリンシパル名の識別子 (クリデンシャル) を指定します。これは、このリモート・ドメイン・アクセス・ポイントの TA_DMACLPOLICY 属性が LOCAL (デフォルト) に設定されている場合、このリモート・ドメインから受け取ったサービス要求に対してローカル・ドメインが割り当てる ID です。この属性は、BEA Tuxedo 7.1 以降のソフトウェアが実行されている TDOMAIN タイプのドメイン・ゲートウェイ、および BEA Tuxedo 8.0 以降が実行されている OSITPX タイプのドメイン・ゲートウェイにのみに適用されます。

TA_DMLOCALPRINCIPALNAME 属性には最大 511 文字を指定できます (最後の NULL 文字を除く)。この属性を指定しないと、ローカル・プリン

シパル名はデフォルトでこのリモート・ドメイン・アクセス・ポイントの TA_DMACCESSPOINTID 文字列になります。

TA_DMTYPE=TDOMAIN の場合に設定可能な属性

TA_DMCONNPRINCIPALNAME: string[0..511]

接続プリンシパル名の識別子。これは、ローカル・ドメイン・アクセス・ポイントに接続するこのリモート・ドメイン・アクセス・ポイントの ID を検証するためのプリンシパル名です。この属性は、BEA Tuxedo 7.1 以降のソフトウェアが実行されている TDOMAIN タイプのドメイン・ゲートウェイにのみ適用されます。

TA_DMCONNPRINCIPALNAME 属性には最大 511 文字を指定できます (最後の NULL 文字を除く)。この属性を指定しない場合は、リモート・ドメイン・アクセス・ポイントの TA_DMACCESSPOINTID 文字列がデフォルト値になります。

デフォルトの認証プラグインで、このリモート・ドメイン・アクセス・ポイントの TA_DMCONNPRINCIPALNAME 属性に値を割り当てる場合、その値は、このリモート・ドメイン・アクセス・ポイントの TA_DMACCESSPOINTID 属性の値と同じでなければなりません。これらの値が一致しないと、ローカル・ドメイン・ゲートウェイとリモート・ドメイン・ゲートウェイを接続しようとしても失敗し、次の userlog(3c) メッセージが生成されます。ERROR: ドメイン *domain_name* の管理用キーを初期化できません。

TA_DMCREDENTIALPOLICY: {LOCAL | GLOBAL}

The credential policy for this remote domain access point. この属性は、BEA Tuxedo 8.0 以降のソフトウェアが実行されている TDOMAIN タイプのドメイン・ゲートウェイにのみ適用されます。

LOCAL の場合は、このリモート・ドメイン・アクセス・ポイントに対するローカル・サービス要求からクリデンシャル (ID) がローカル・ドメインによって削除されます。GLOBAL の場合、このリモート・ドメイン・アクセス・ポイントに対するローカル・サービス要求からクリデンシャルは削除されません。この属性を指定しない場合、デフォルトは LOCAL です。

TA_DMCREDENTIALPOLICY 属性は、リモート・ドメインに送信する前にローカル・ドメインがローカル・サービス要求からクリデンシャルを削除するかどうかを指定します。TA_DMACLPOLICY 属性は、リモート・ドメインから受信したサービス要求のクリデンシャルが、TA_DMLOCALPRINCIPALNAME 属性で指定されたプリンシパル名に置換されるかどうかを指定します。

TA_DMMACHINETYPE: *string*[0..15]
ドメインをグループ化して、このリモート・ドメイン・アクセス・ポイントに関連付けられているマシンとローカル・ドメイン・アクセス・ポイントに関連付けられているマシン間のメッセージの符号化と復号化を省略するために使用します。TA_DMMACHINETYPE を指定しない場合、デフォルトで符号化または復号化が実行されます。TA_DMMACHINETYPE フィールドに設定した値が接続のための T_DM_LOCAL クラスと T_DM_REMOTE クラスで共通している場合、データの符号化と復号化が省略されます。TA_DMMACHINETYPE には、15 文字までの任意の文字列値を指定できます。この値は比較のためだけに使用します。

TA_DMTYPE=SNAX|OSITPX の場合に設定可能な属性

TA_DMCODEPAGE: *string*[1..20]
このリモート・ドメイン・アクセス・ポイントを通して送信される要求と応答を変換する際に使用するデフォルトの変換テーブルの名前。

制限事項 この要求と同じドメイン・タイプのローカル・ドメイン・アクセス・ポイントをサポートするドメイン・ゲートウェイ管理サーバ (GWADM) がアクティブのとき、SET を実行して TA_STATE を INValid にしたり、TA_DMACCESSPOINTID、TA_DMTYPE、TA_DMMACHINETYPE、または TA_DMCODEPAGE 属性を更新することはできません。

T_DM_ACL、T_DM_IMPORT、T_DM_OSITP、T_DM_OSITPX、T_DM_ROUTING、または T_DM_TDOMAIN クラスのインスタンスによって T_DM_REMOTE クラスのインスタンスが参照される場合、そのインスタンスを削除することはできません。

T_DM_RESOURCES クラスの定義

概要 T_DM_RESOURCES クラスは、ドメイン固有のコンフィギュレーション情報を表します。

属性表

表 22DM_MIB(5): T_DM_RESOURCES クラス定義の属性表

属性	タイプ	パーミッション	値	デフォルト値
TA_DMVERSION(r)	string	rw-r--r--	*string*[1..30]	N/A

(r)― 新しいオブジェクトが作成される場合に必須です。
(k)― オブジェクトを取り出すためのキー・フィールドです。
(*)― すべての SET 操作で必須のキー・フィールドです。

属性の意味 TA_DMVERSION: *string*[1..30]
Domains コンフィギュレーションに対するユーザ指定の識別子。

制限事項 なし

T_DM_ROUTING クラスの定義

概要 T_DM_ROUTING クラスは、リモート・ドメイン・アクセス・ポイントを通して要求をドメインにルーティングするためのルーティング基準情報を表します。

属性表

表 23DM_MIB(5): T_DM_ROUTING クラス定義の属性表

属性	タイプ	パーミッション	値	デフォルト値
TA_DMROUTINGNAME(r)(k)(*)	string	rw-r--r--	*string*[1..15]	N/A
TA_DMBUFTYPE(r)(k)(*)	string	rw-r--r--	*string*[1..256]	N/A
TA_DMFIELD(r)	string	rw-r--r--	*string*[1..30]	N/A
TA_DMFIELDTYPE	string	rw-r--r--	"{CHAR | SHORT | LONG | FLOAT | DOUBLE | STRING}"	N/A
TA_DMRANGES(r)	string	rw-r--r--	*string*[1..4096]	N/A
TA_STATE(r)	string	rw-r--r--	GET: "VAL" SET: "{NEW | INV}"	N/A N/A

(r)— 新しいオブジェクトが作成される場合に必須です。
(k)— オブジェクトを取り出すためのキー・フィールドです。
(*)— すべての SET 操作で必須のキー・フィールドです。

属性の意味

TA_DMROUTINGNAME: *string*[1..15]
ルーティング基準テーブル・エントリの名前。Domains コンフィギュレーションの T_DM_ROUTING エントリの範囲内で一意な識別子です。

TA_DMBUFTYPE: *string*[1..256]
"type1[:*subtype1*[,*subtype2* . . .]][;*type2*[:*subtype3*[,*subtype4* . . .]] . . .]"
このルーティング・エントリで有効なデータ・バッファのタイプとサブタイプのリスト。最大で 32 のタイプとサブ・タイプの組み合わせを使用できます。タイプは、FML、FML32、XML、VIEW、VIEW32、

X_C_TYPE、および X_COMMON に制限されています。FML、FML32、または XML に対してはサブタイプを指定できず、VIEW、VIEW32、X_C_TYPE、および X_COMMON ではサブタイプを指定する必要があります ("*" は使用できません)。サブ・タイプの名前には、セミコロン (;)、カンマ (,)、コロン (:)、アスタリスク (*) は使用できません。タイプとサブタイプのペアのうち、重複するものは同じルーティング基準名として指定できません。タイプとサブタイプのペアが一意の場合、複数のルーティング・エントリは同じ基準名を持つことができます。単一のルーティング・エントリに複数のバッファ・タイプが指定される場合、各バッファ・タイプに対するルーティング・フィールドのデータ型は同じでなければなりません。

TA_DMFIELD: *string*[1..30]

ルーティングが適用されるフィールドの名前。

FML (および FML32) バッファ・タイプの場合、TA_DMFIELD に指定する FML フィールド名は、FML フィールド・テーブルで定義されている必要があります。ルーティングを実行する際、フィールド名は FLDTBLDIR および FIELDTBLS (FML32 の場合は FLDTBLDIR32 および FIELDTBLS32) 環境変数を使用して検索されます。

VIEW (および VIEW32) バッファ・タイプの場合、TA_DMFIELD に指定する VIEW フィールド名は、FML VIEW テーブルで定義されている必要があります。ルーティングを実行する際、フィールド名は VIEWDIR および VIEWFILES (VIEW32 の場合は VIEWDIR32 および VIEWFILES32) 環境変数を使用して検索されます。

バッファを正しいリモート・ドメイン・アクセス・ポイントにルーティングする際は、該当するテーブルを使用してバッファ内のデータ依存型ルーティング・フィールド値を取得します。

XML バッファ・タイプの場合、TA_DMFIELD には、ルーティング要素のタイプ (または名前) か、ルーティング要素の属性名のいずれかが含まれます。

XML バッファ・タイプの場合、TA_DMFIELD 属性の構文は次のとおりです。

"*root_element*[/*child_element*][/*child_element*]
[/...][/@*attribute_name*]"

要素は、XML ドキュメントまたはデータグラム要素のタイプとして処理されます。インデックスはサポートされません。したがって、BEA Tuxedo システムは、データ依存型ルーティングで XML バッファを処理する際に、与えられた要素タイプの最初のオカレンスだけを認識します。この情報は、メッセージの送信時に、データ依存型ルーティングに関連する要素の内容を取得するために使用されます。内容は UTF-8 で符号化された文字列である必要があります。

属性は、定義されている要素の XML ドキュメントまたはデータグラム属性として処理されます。この情報は、メッセージの送信時に、データ依存型ルーティングに関連する属性値を取得するために使用されます。値は UTF-8 で符号化された文字列である必要があります。

要素名と属性名の組み合わせの長さは最大で 30 文字です。

ルーティング・フィールドのタイプは、TA_DMFIELDTYPE 属性で指定できます。

TA_DMFIELDTYPE: "{CHAR | SHORT | LONG | FLOAT | DOUBLE | STRING}"

TA_DMFIELD 属性で指定したルーティング・フィールドのタイプ。タイプには CHAR、SHORT、LONG、FLOAT、DOUBLE、または STRING のいずれか 1 つを指定できます。この属性は、TA_DMBUFTYPE が XML の場合に必要です。TA_DMBUFTYPE が FML、VIEW、X_C_TYPE、または X_COMMON. の場合、この属性には何も指定しません。

TA_DMRANGES: *string*[1..4096]

TA_DMFIELD ルーティング・フィールドの範囲、およびそれに関連付けられたリモート・ドメイン・アクセス・ポイントを指定します。文字列の形式は、カンマで区切って並べられた範囲とグループ名の組み合わせです。範囲とグループ名の組み合わせの形式は次のとおりです。

"*lower*[-*upper*]:*accesspoint*"

lower と *upper* は、符号を持つ数値、または一重引用符で囲んだ文字列です。*lower* には、*upper* 以下の値を設定する必要があります。文字列値で一重引用符を使用する場合は、引用符の前にバックスラッシュを 2 つ入力します (例 : 'O\\'Brien')。マシン上の関連するフィールドのデータ型の最小値を示すには、MIN を使用します。マシン上の関連するフィールドのデータ型の最大値を示すには、MAX を使

用します。したがって、"MIN - -5 " は -5 以下のすべての数を表し、"6-MAX" は 6 以上のすべての数を表します。

範囲 (range) 内のメタキャラクタ "*" (ワイルドカード) は、既にエントリとして指定した範囲では使用されなかった任意の値を示します。各エントリでは 1 つのワイルドカードによる範囲指定だけが可能です。* は最後に指定します。続けて範囲を指定すると無視されます。

数値ルーティング・フィールドには数値で範囲を指定し、文字列ルーティング・フィールドには、文字列で範囲を指定する必要があります。

文字列で範囲を設定する場合は、文字列、carray、および文字フィールド型の値を一重引用符で囲みます。先頭に符号を付けることはできません。short 型および long 型の整数値は数字の文字列であり、オプションで先頭に正の符号または負の符号を付けることができます。C コンパイラまたは atof(3) で使用できる浮動小数点数は、まず任意の符号、次に数字列 (小数点が入ってもよい)、任意の e または E、任意の符号またはスペース、最後に整数という形式を取ります。

raccesspoint パラメータは、フィールドが範囲と一致する場合に要求のルーティング先となるリモート・ドメイン・アクセス・ポイントを示します。*raccesspoint* に "*" を指定すると、サービスをインポートする任意のリモート・ドメイン・アクセス・ポイントに要求が送られることを示します。

TA_STATE:

GET: "{VALid}"
GET 操作は、T_DM_ROUTING オブジェクトのコンフィギュレーション情報を検索します。以下に示す状態は、GET 要求に対する応答で返される TA_STATE 属性の意味を示します。以下に示されていない状態は返されません。

"VALid"	オブジェクトが存在します。

SET: "{NEW | INValid}"
SET 操作は、選択した T_DM_ROUTING オブジェクトのコンフィギュレーション情報を更新します。以下に示す状態は、SET 要

求の TA_STATE の意味を示します。以下に示されていない状態は設定できません。

"NEW"	新しいオブジェクトを作成します。
unset	既存のオブジェクトを変更します。この組み合わせは "INValid" 状態では使用できません。正常終了した場合、オブジェクトの状態は変わりません。
"INValid"	オブジェクトを削除します。

制限事項 T_DM_ROUTING クラスのインスタンスは、T_DM_IMPORT クラスのインスタンスによって参照されている場合は削除できません。

T_DM_RPRINCIPAL クラスの定義

概要 T_DM_RPRINCIPAL クラスは、リモート・プリンシパル名のパスワード・コンフィギュレーション情報を表します。

属性表

表 24DM_MIB(5): T_DM_RPRINCIPAL クラス定義の属性表

属性	タイプ	パーミッション	値	デフォルト値
TA_DMRACCESSPOINT(r)(k)(*)	string	rw-r--r--	*string*[1..30]	N/A
TA_DMRPRINNAME(r)(k)(*)	string	rw-------	*string*[1..30]	N/A
TA_DMRPRINPASSWD(r)(*)	string	-w-------	*string*[0..8]	N/A
TA_STATE(r)	string	rw-r--r--	GET: "VAL" SET: "{NEW | INV}"	N/A N/A

(r)— 新しいオブジェクトが作成される場合に必須です。
(k)— オブジェクトを取り出すためのキー・フィールドです。
(*)— すべての SET 操作で必須のキー・フィールドです。

属性の意味

TA_DMRACCESSPOINT: *string*[1..30]
プリンシパルが適用可能なリモート・ドメイン・アクセス・ポイント。

注記 TA_DMRACCESSPOINT と TA_DMRPRINNAME の組み合わせは、Domains コンフィギュレーション内の TA_DM_RPRINCIPAL エントリの範囲内で一意でなければなりません。

TA_DMRPRINNAME: *string*[1..30]
リモート・プリンシパル名。

注記 TA_DMRACCESSPOINT と TA_DMRPRINNAME の組み合わせは、Domains コンフィギュレーション内の TA_DM_RPRINCIPAL エントリの範囲内で一意でなければなりません。

TA_DMRPRINPASSWD: *string*[0..8]

TA_DMACCESSPOINT で識別されるリモート・ドメイン・アクセス・ポイントを通して通信するとき、プリンシパル名に対して使用されるリモート・パスワード。

TA_STATE:

GET: "{VALid}"

GET 操作は、T_DM_RPRINCIPAL オブジェクトのコンフィギュレーション情報を検索します。以下に示す状態は、GET 要求に対する応答で返される TA_STATE 属性の意味を示します。以下に示されていない状態は返されません。

"VALid"	オブジェクトが存在します。

SET: "{NEW | INValid}"

SET 操作は、選択した T_DM_RPRINCIPAL オブジェクトのコンフィギュレーション情報を更新します。以下に示す状態は、SET 要求の TA_STATE の意味を示します。以下に示されていない状態は設定できません。

"NEW"	新しいオブジェクトを作成します。状態変更は、"INValid" 状態でのみ可能です。正常に終了すると、オブジェクトの状態は "VALid" になります。
unset	既存のオブジェクトを変更します。この組み合わせは "INValid" 状態では使用できません。
"INValid"	オブジェクトを削除します。状態変更は、"VALid" 状態でのみ可能です。正常に終了すると、オブジェクトの状態は "INValid" になります。

制限事項　BEA Tuxedo リリース 7.1 以降では、T_DM_RPRINCIPAL クラスは SNAX ドメイン・ゲートウェイ・タイプにのみ適用されます。

T_DM_SNACRM クラスの定義

概要 T_DM_SNACRM クラスは、指定したローカル・ドメイン・アクセス・ポイントに対する SNA-CRM 固有のコンフィギュレーションを定義します。

属性表

表 25DM_MIB(5): T_DM_SNACRM クラス定義の属性表

属性	タイプ	パーミッション	値	デフォルト値
TA_DMSNACRM(k)(r)(*)	string	rw-r--r--	*string*[1..30]	N/A
TA_DMLACCESSPOINT(k)(r)	string	rw-r--r--	*string*[1..30]	N/A
TA_STATE(r)	string	rw-r--r--	GET: "VAL" SET: "{NEW | INV}"	N/A N/A
TA_DMNWADDR(r)	string	rw-r--r--	*string*[1..78]	N/A
TA_DMNWDEVICE(r)	string	rw-r--r--	*string*[1..78]	N/A

(r)— 新しいオブジェクトが作成される場合に必須です。
(k)— オブジェクトを取り出すためのキー・フィールドです。
(*)— すべての SET 操作で必須のキー・フィールドです。

属性の意味

TA_DMSNACRM: *string*[1..30]
この T_DM_SNACRM エントリの名前。TA_DMSNACRM は、この SNA CRM エントリを識別するために使用する Domains コンフィギュレーション内の SNA CRM エントリの範囲内で一意な識別子です。

TA_DMLACCESSPOINT: *string*[1..30]
この SNA CRM を使用するローカル・ドメイン・アクセス・ポイント・エントリの名前。

TA_STATE:

GET: "{VALid}"
GET 操作は、T_DM_SNACRM オブジェクトのコンフィギュレーション情報を検索します。以下に示す状態は、GET 要求に対す

る応答で返される TA_STATE 属性の意味を示します。以下に示されていない状態は返されません。

"VALid"	オブジェクトが存在します。

SET: "{NEW | INValid}"

SET 操作は、選択した T_DM_SNACRM オブジェクトのコンフィギュレーション情報を更新します。以下に示す状態は、SET 要求で設定される TA_STATE の意味を示します。以下に示されていない状態は設定できません。

"NEW"	新しいオブジェクトを作成します。この状態変更は、"INValid" 状態でのみ可能です。正常に終了すると、オブジェクトの状態は "VALid" になります。
unset	既存のエントリを変更します。この組み合わせは "INValid" 状態では使用できません。
"INValid"	オブジェクトを削除します。この状態変更は、"VALid" 状態でのみ可能です。正常に終了すると、オブジェクトの状態は "INValid" になります。

TA_DMNWADDR: *string*[1..78]

ローカル・ドメイン・アクセス・ポイントのドメイン・ゲートウェイと SNA CRM の間の通信に使用するネットワーク・アドレスを指定します。

TA_DMNWDEVICE: *string*[1..78]

ローカル・ドメイン・アクセス・ポイントのドメイン・ゲートウェイと SNA CRM の間の通信に使用するネットワーク・デバイスを指定します。

制限事項　参照するローカル・アクセス・ポイントのドメイン・ゲートウェイ管理 (GWADM) サーバがアクティブのとき、T_DM_SNACRM クラスのインスタンスを削除または更新することはできません。

このクラスのインスタンスを追加または更新する SET 操作の実行時には、TA_DMLACCESSPOINT で指定するローカル・ドメイン・アクセス・ポイントが T_DM_LOCAL クラス内に存在しなければなりません。アクセス・ポイントが存在しない場合、TA_DMLACCESSPOINT 属性に対して "not defined" エラーが返され、操作は失敗します。

T_DM_SNALINK クラスの定義

概要 T_DM_SNALINK クラスは、リモート・ドメイン・アクセス・ポイントの SNAX 固有のコンフィギュレーション情報を表します。

属性表

表 26DM_MIB(5): T_DM_SNALINK クラス定義の属性表

属性	タイプ	パーミッション	値	デフォルト値
TA_DMSNALINK(r)(k)(*)	string	rw-r--r--	*string*[1..30]	N/A
TA_DMSNASTACK(r)(k)	string	rw-r--r--	*string*[1..30]	N/A
TA_DMRACCESSPOINT(r)(k)	string	rw-r--r--	*string*[1..30]	N/A
TA_DMLSYSID(r)	string	rw-r--r--	*string*[1..4]	N/A
TA_DMRSYSID(r)	string	rw-r--r--	*string*[1..4]	N/A
TA_DMLUNAME(r)	string	rw-r--r--	*string*[1..8]	N/A
TA_DMMINWIN(r)	short	rw-r--r--	0 <= *num* <= 32767	N/A
TA_DMMODENAME(r)	string	rw-r--r--	*string*[1..8]	N/A
TA_STATE(r)	string	rw-r--r--	GET: "VAL" SET: "{NEW | INV}"	N/A N/A
TA_DMSECTYPE	string	rw-r--r--	"{LOCAL | IDENTIFY | VERIFY | PERSISTENT | MIXIDPE}"	"LOCAL"
TA_DMSTARTTYPE	string	rw-r--r--	"{AUTO | COLD}"	"AUTO"
TA_DMMAXSNASESS	short	rw-r--r--	0 <= *num* <= 32767	64
TA_DMMAXSYNCLVL	short	r--r--r--	0 <= *num* <= 2	0

(r)— 新しいオブジェクトが作成される場合に必須です。
(k)— オブジェクトを取り出すためのキー・フィールドです。
(*)— すべての SET 操作で必須のキー・フィールドです。

属性の意味

TA_DMSNALINK: *string*[1..30]

T_DM_SNALINK エントリの名前。この TA_DMSNALINK エントリを識別するために使用する Domains コンフィギュレーション内の SNA LINK エントリの範囲内で一意な識別子です。

TA_DMSNASTACK: *string*[1..30]

このリモート・ドメイン・アクセス・ポイントに到達するために使用する SNAX スタック・エントリの名前。

TA_DMRACCESSPOINT: *string*[1..30]

このエントリが SNAX コンフィギュレーション・データを提供するリモート・ドメイン・アクセス・ポイントの名前を識別します。

TA_DMLSYSID: *string*[1..4]

リモート論理ユニット (LU) への SNA リンクを確立する際に使用するローカル SYSID。

TA_DMRSYSID: *string*[1..4]

リモート LU への SNA リンクを確立する際に使用するリモート SYSID。

TA_DMLUNAME: *string*[1..8]

リモート・ドメイン・アクセス・ポイントに関連付ける LU 名を指定します。

TA_DMMINWIN: 0 <= *num* <= 32767

リモート LU への winner セッションの最小数。

TA_DMMODENAME: *string*[1..8]

リモート LU へのセッションのセッション特性および名前を指定します。

TA_STATE:

GET: "{VALid}"

GET 操作は、T_DM_SNALINK オブジェクトのコンフィギュレーション情報を検索します。以下に示す状態は、GET 要求に対する応答で返される TA_STATE 属性の意味を示します。以下に示されていない状態は返されません。

"VALid"	オブジェクトが存在します。

SET: "{NEW | INValid}"
SET 操作は、選択した T_DM_SNALINK オブジェクトのコンフィギュレーション情報を更新します。以下に示す状態は、SET 要求で設定される TA_STATE の意味を示します。以下に示されていない状態は設定できません。

"NEW"	新しいオブジェクトを作成します。
unset	既存のオブジェクトを変更します。この組み合わせは "INValid" 状態では使用できません。
"INValid"	オブジェクトを削除します。

TA_DMSECTYPE: "{LOCAL | IDENTIFY | VERIFY | PERSISTENT | MIXIDPE}"
リモート LU へのセッションで使用する SNA セキュリティのタイプを指定します。この属性の有効な値は、"LOCAL"、"IDENTIFY"、"VERIFY"、"PERSISTENT"、および "MIXIDPE" です。

TA_DMSTARTTYPE: "{AUTO | COLD}"
宛先 LU のセッション起動のタイプを指定します。この属性を "COLD" に設定すると、LU が COLDSTART で起動されます。"AUTO" に設定すると、SNACRM とドメイン・ゲートウェイによって、LU を COLDSTART するか WARMSTART するかが決まります。

TA_DMMAXSNASESS: 0 <= *num* <= 32767
リモート LU で確立するセッションの最大数を指定します。

TA_DMMAXSYNCLVL: 0 <= *num* <= 2
このリモート LU でサポートできる最大の SYNC LEVEL。

制限事項 ドメイン・ゲートウェイ管理 (GWADM) サーバがアクティブなローカル・ドメイン・アクセス・ポイントを参照する T_DM_SNACRM クラスのインスタンスを T_DM_SNASTACK クラスのインスタンスが参照している場合、T_DM_SNASTACK クラスのインスタンスを参照する T_DM_SNALINK クラスのインスタンスを削除または更新することはできません。

このクラスのインスタンスを追加または更新する SET 操作の実行時には、次の条件を満たしている必要があります。

- TA_DMRACCESSPOINT 属性で指定されている特定のドメイン・アクセス・ポイントが、T_DM_REMOTE クラス内に存在している。アクセス・ポイントが存在しない場合、TA_DMRACCESSPOINT 属性に対して "not defined" エラーが返され、操作は失敗します。
- TA_DMSNASTACK 属性で指定されている SNA スタック参照名が、T_DM_SNASTACK クラス内に存在している。参照名が存在しない場合、TA_DMSNASTACK 属性に対して "not defined" エラーが返され、操作は失敗します。

T_DM_SNASTACK クラスの定義

概要 T_DM_SNASTACK クラスは、特定の SNA CRM が使用する SNA スタックを定義します。

属性表

表 27DM_MIB(5): T_DM_SNASTACK クラス定義の属性表

属性	タイプ	パーミッション	値	デフォルト値
TA_DMSNASTACK(r)(k)(*)	string	rw-r--r--	*string*[1..30]	N/A
TA_DMSNACRM(r)(k)	string	rw-r--r--	*string*[1..30]	N/A
TA_DMSTACKTYPE(r)	string	rw-r--r--	*string*[1..30]	N/A
TA_DMLUNAME(r)	string	rw-r--r--	*string*[1..8]	N/A
TA_DMTPNAME(r)	string	rw-r--r--	*string*[1..8]	N/A
TA_DMSTACKPARMS(r)	string	rw-r--r--	*string*[1..128]	N/A
TA_STATE(r)	string	rw-r--r--	GET: "VAL" SET: "{NEW | INV}"	N/A N/A

(r)— 新しいオブジェクトが作成される場合に必須です。
(k)— オブジェクトを取り出すためのキー・フィールドです。
(*)— すべての SET 操作で必須のキー・フィールドです。

属性の意味

TA_DMSNASTACK: *string*[1..30]
この T_DM_SNASTACK エントリの名前。TA_DMSNASTACK は、Domains コンフィギュレーション内の T_DM_SNASTACK エントリ名の範囲内で一意な識別子です。

TA_DMSNACRM: *string*[1..30]
この SNA プロトコル・スタック定義を使用する SNA CRM の T_DM_SNACRM エントリを識別します。

TA_DMSTACKTYPE: *string*[1..30]
使用するプロトコル・スタックを識別します。

TA_DMLUNAME: *string*[1..8]
このスタック定義を使用して確立するセッションで使用する LU 名を指定します。

TA_DMTPNAME: *string*[1..8]
SNA スタックに関連付けられた TP 名を指定します。値 "*" を指定すると、すべての TP 名を使用できることを示します。

TA_DMSTACKPARMS: *string*[1..128]
プロトコル・スタック固有のパラメータを提供します。

TA_STATE:

GET: "{VALid}"
GET 操作は、T_DM_SNASTACK オブジェクトのコンフィギュレーション情報を検索します。以下に示す状態は、GET 要求に対する応答で返される TA_STATE 属性の意味を示します。以下に示されていない状態は返されません。

"VALid"	オブジェクトが存在します。

SET: "{NEW | INValid}"
SET 操作は、選択した T_DM_SNASTACK オブジェクトのコンフィギュレーション情報を更新します。以下に示す状態は、SET 要求の TA_STATE の意味を示します。以下に示されていない状態は設定できません。

"NEW"	新しいオブジェクトを作成します。この状態変更は、"INValid" 状態でのみ可能です。正常に終了すると、オブジェクトの状態は "VALid" になります。
unset	既存のオブジェクトを変更します。この組み合わせは "INValid" 状態では使用できません。
"INValid"	オブジェクトを削除します。この状態変更は、"VALid" 状態でのみ可能です。正常に終了すると、オブジェクトの状態は "INValid" になります。

制限事項　ドメイン・ゲートウェイ管理 (GWADM) サーバがアクティブなローカル・ドメイン・アクセス・ポイントを参照する T_DM_SNACRM オブジェクトをこのクラスのインスタンスが参照している場合、このクラスのインスタンスを削除または更新することはできません。

このクラスのインスタンスを追加または更新する SET 操作の実行時には、TA_DMSNACRM 属性で指定する SNA CRM 名が T_DM_SNACRM クラス内に存在しなければなりません。SNA CRM 名が存在しない場合、TA_DMSNACRM 属性に対して "not defined" エラーが返され、操作は失敗します。

T_DM_TDOMAIN クラスの定義

概要 T_DM_TDOMAIN クラスは、ローカルまたはリモート・ドメイン・アクセス・ポイントに対する TDomain 固有のコンフィギュレーションを定義します。

属性表

表 28DM_MIB(5): T_DM_TDOMAIN クラス定義の属性表

属性	タイプ	パーミッション	値	デフォルト値
TA_DMACCESSPOINT(r)(k)(*)	string	rw-r--r--	*string*[1..30]	N/A
TA_DMNWADDR(r)(k)(*)	string	rw-r--r--	*string*[1..256] (注 1)	N/A
TA_STATE(r)	string	rw-r--r--	GET: "VAL" SET: "{NEW | INV}"	N/A N/A
TA_DMNWDEVICE	string	rw-r--r--	*string*[1..78]	N/A
TA_DMCMPLIMIT	long	rw-rw-r--	0 <= *num* <= MAXLONG	MAXLONG
TA_DMMINENCRYPTBITS	string	rw-------	"{0 | 40 | 56 | 128}" (注 2)	"0"
TA_DMMAXENCRYPTBITS	string	rw-------	"{0 | 40 | 56 | 128}" (注 2)	"128"
TA_DMCONNECTION_POLICY	string	rwxr--r--	"{LOCAL | ON_DEMAND | ON_STARTUP | INCOMING_ONLY}"	"LOCAL" (注 3) (注 5 も参照)
TA_DMMAXRETRY	long	rwxr--r--	0 <= *num* <= MAXLONG	0
TA_DMRETRY_INTERVAL	long	rwxr--r--	0 <= *num* <= MAXLONG	60
TA_DMTCPKEEPALIVE	string	rwxr--r--	"{LOCAL | NO | YES}"	"LOCAL" (注 3) "NO" (注 4)
TA_DMKEEPALIVE	long	rwxr--r--	-1 <= *num* <= 2147483647	-1 (注 3) 0 (注 4)
TA_DMKEEPALIVEWAIT	long	rwxr--r--	0 <= *num* <= 2147483647	0

表 28DM_MIB(5): T_DM_TDOMAIN クラス定義の属性表 (続き)

属性	タイプ	パーミッション	値	デフォルト値

(r)— 新しいオブジェクトが作成される場合に必須です。
(k)— オブジェクトを取り出すためのキー・フィールドです。
(*)— すべての SET 操作で必須のキー・フィールドです。

注 1 BEA Tuxedo 8.0 以前のリリースの場合、この属性の文字列の長さは最大 78 バイトです。
注 2 リンク・レベルの暗号化の値 40 は、下位互換性を維持するために提供されています。
注 3 リモート・ドメイン・アクセス・ポイント用のデフォルト。
注 4 ローカル・ドメイン・アクセス・ポイント用のデフォルト。
注 5 ローカル･ドメイン･アクセス･ポイント用の TA_DMCONNECTION_POLICY のデフォルト値は、T_DM_LOCAL クラスで指定した TA_DMCONNECTION_POLICY の値です。

属性の意味

TA_DMACCESSPOINT: *string*[1..30]

このエントリが TDomain 固有のコンフィギュレーション・データを提供するローカルまたはリモート・ドメイン・アクセス・ポイントの名前。

Domains リンク・レベルのフェイルオーバーを使用している場合は、同じ TA_DMACCESSPOINT 属性値で複数の T_DM_TDOMAIN クラス･エントリを定義できます。

TA_DMNWADDR: *string*[1..256] (BEA Tuxedo 8.0 以前では最大 78 バイト)

アクセス・ポイントに関連付けられたネットワーク・アドレスを指定します。ローカル・ドメイン・アクセス・ポイントの場合は、受信する接続指示を受け付けるためのアドレスを指定します。リモート・ドメイン・アクセス・ポイントの場合は、リモート・ドメイン・アクセス・ポイントに接続するために使用する接続先アドレスを指定します。このフィールドの値は、すべての T_DM_TDOMAIN エントリ間で一意でなければなりません。

TA_STATE:

GET: "{VALid}"

GET 操作は、T_DM_TDOMAIN オブジェクトのコンフィギュレーション情報を検索します。以下に示す状態は、GET 要求に対す

る応答で返される TA_STATE 属性の意味を示します。以下に示されていない状態は返されません。

"VALid"	オブジェクトが存在します。

SET: "{NEW | INValid}"

SET 操作は、選択した T_DM_TDOMAIN オブジェクトのコンフィギュレーション情報を更新します。以下に示す状態は、SET 要求で設定される TA_STATE の意味を示します。以下に示されていない状態は設定できません。

"NEW"	新しいオブジェクトを作成します。この状態変更は、"INValid" 状態でのみ可能です。正常に終了すると、オブジェクトの状態は "VALid" になります。
unset	既存のオブジェクトを変更します。この組み合わせは "INValid" 状態では使用できません。
"INValid"	オブジェクトを削除します。この状態変更は、"VALid" 状態でのみ可能です。正常に終了すると、オブジェクトの状態は "INValid" になります。

TA_DMNWDEVICE: *string*[1..78]

使用するネットワーク・デバイスを指定します。ローカル・ドメイン・アクセス・ポイント・エントリの場合、この属性は接続指示を受け付けるために使用するデバイスを指定します。リモート・ドメイン・アクセス・ポイントの場合、リモート・ドメイン・アクセス・ポイントに接続するために使用するデバイスを指定します。

TA_DMCMPLIMIT: 0 <= *num* <= MAXLONG

リモート・ドメイン・アクセス・ポイントでのみ有効です。このアクセス・ポイントへのトラフィックを圧縮する際のしきい値メッセージです。

TA_DMMINENCRYPTBITS: "{0 | 40 | 56 | 128}"

リモート・ドメイン・アクセス・ポイントでのみ有効です。このアクセス・ポイントへの接続を確立する際に必要となる暗号化の最小レベルを指定します。"0" は暗号化のないことを意味し、"40"、"56"、お

よび "128" は暗号化キーのビット長を示します。ここで指定する最小レベルの暗号化が満たされない場合、リンクの確立は失敗します。デフォルト値は "0" です。

40 ビットの値は、下位互換性のために用意されています。

注記　この属性を変更しても、確立済みの接続には反映されません。

TA_DMMAXENCRYPTBITS: "{0 | 40 | 56 | 128}"

リモート・ドメイン・アクセス・ポイントでのみ有効です。このアクセス・ポイントへのネットワーク・リンクを確立する際に許可する暗号化の最大レベルを指定します。"0" は暗号化のないことを意味し、"40"、"56"、および "128" は暗号化キーのビット長を示します。デフォルト値は "128" です。

40 ビットの値は、下位互換性のために用意されています。

注記　この属性を変更しても、確立済みの接続には反映されません。

TA_DMCONNECTION_POLICY = "{LOCAL | ON_DEMAND | ON_STARTUP | INCOMING_ONLY}"

このローカルまたはリモート・ドメイン・アクセス・ポイントに関連付けられている TDomain ゲートウェイが接続を確立するための条件を指定します。有効な値は、"LOCAL"、"ON_DEMAND"、"ON_STARTUP"、または "INCOMING_ONLY" です。"LOCAL" は、リモート・ドメイン・アクセス・ポイントにのみ適用されます。

BEA Tuxedo 8.1 以降のソフトウェアを実行する場合、TA_DMCONNECTION_POLICY 属性は T_DM_TDOMAIN クラスでも指定できます。特定のローカルまたはリモート・ドメイン・アクセス・ポイントの T_DM_TDOMAIN クラスの値は、T_DM_LOCAL クラスのグローバル値に優先します。グローバル接続ポリシーを無効にできるので、リモート・ドメイン単位で接続ポリシーをコンフィギュレーションできます。

ローカル・ドメイン・アクセス・ポイントの接続ポリシーを指定しない場合、デフォルトとして T_DM_LOCAL クラスに指定されるグローバル接続ポリシーが使用されます。T_DM_TDOMAIN クラスにグローバル接続ポリシーを指定する場合、T_DM_LOCAL クラスにグローバル接続ポリシーを指定しないでください。

"LOCAL"
: 接続ポリシーが "LOCAL" の場合、リモート・ドメイン・アクセス・ポイントは T_DM_LOCAL クラスに指定されるグローバル接続ポリシーを受け入れます。"LOCAL" は、リモート・ドメイン・アクセス・ポイントに対するデフォルトの接続ポリシーです。"LOCAL" を除き、リモート・ドメイン・アクセス・ポイントに対する接続ポリシーは、ローカル・ドメイン・アクセス・ポイントに対する接続ポリシーに優先します。

"ON_DEMAND"
: 接続ポリシーが "ON_DEMAND" の場合、クライアントがリモート・サービスを要求したとき、または dmadmin(1) connect コマンドが実行されたときにのみ、TDomain ゲートウェイは接続を試行します。接続ポリシーが "ON_DEMAND" の場合、再接続は行われません。

"ON_STARTUP"
: 接続ポリシーが "ON_STARTUP" の場合、TDomain ゲートウェイはゲートウェイ・サーバの初期化時に接続を試行します。"ON_STARTUP" に設定した場合、リモート・ドメインへの接続が確立された場合にのみそのリモート・サービス (TDomain ゲートウェイによって宣言されたサービス) が宣言されますつまり、リモート・ドメインとの接続が確立されていないと、リモート・サービスは中断されます。デフォルトでは、失敗した接続が 60 秒おきに再試行されるよう設定されています。再接続の間隔は、T_DM_TDOMAIN クラスの TA_DMRETRY_INTERVAL 属性で変更できます。このクラスの TA_DMMAXRETRY 属性も参照してください。

"INCOMING_ONLY"
: 接続ポリシーが "INCOMING_ONLY" の場合、TDomain ゲートウェイは起動時にリモート・ドメインへの接続を試みません。このため、リモート・サービスは最初は中断されています。TDomain ゲートウェイは、リモート・ドメインからの接続を受信したときに利用可能になります。リモート・サービスは、ドメイン・ゲートウェイが接続を受信したときか、dmadmin(1) connect コマンドで管理接続が確立されたときに宣言されます。接続ポリシーが "INCOMING_ONLY" の場合、再接続は行われません。

TA_DMMAXRETRY: 0 <= *num* <= MAXLONG

このローカルまたはリモート・ドメイン・アクセス・ポイントに関連付けられている TDomain ゲートウェイが接続を試行する回数。この属性は、BEA Tuxedo 8.1 以降のソフトウェアが実行されているときに T_DM_TDOMAIN クラスで使用でき、このアクセス・ポイントの TA_DMCONNECTION_POLICY 属性が "ON_STARTUP" に設定されている場合に有効です。それ以外の接続ポリシーの場合、自動再試行は無効になります。

TA_DMMAXRETRY の最小値は 0 で、最大値は MAXLONG (2147483647) です。MAXLONG (デフォルト) の場合、再接続処理が無限に繰り返されるか、または接続が確立されるまで繰り返されます。

TA_DMRETRY_INTERVAL: 0 <= *num* <= MAXLONG

このローカルまたはリモート・ドメイン・アクセス・ポイントに関連付けられている TDomain ゲートウェイが接続を自動的に試行する間隔。この属性は、BEA Tuxedo 8.1 以降のソフトウェアが実行されているときに T_DM_TDOMAIN クラスで使用でき、このアクセス・ポイントの TA_DMCONNECTION_POLICY 属性が "ON_STARTUP" に設定されている場合に有効です。それ以外の接続ポリシーの場合、自動再試行は無効になります。

TA_DMRETRY_INTERVAL の最小値は 0 で、最大値は MAXLONG (2147483647) です。デフォルト値は 60 です。TA_DMMAXRETRY が 0 に設定されている場合、TA_DMRETRY_INTERVAL は設定できません。

TA_DMTCPKEEPALIVE = "{LOCAL | NO | YES}"

ローカルまたはリモート・ドメイン・アクセス・ポイントの TCP レベル・キープアライブを有効にします。有効な値は、"LOCAL"、"NO"、または "YES" です。"LOCAL" は、リモート・ドメイン・アクセス・ポイントにのみ適用されます。

TA_DMTCPKEEPALIVE 属性は、BEA Tuxedo 8.1 以降のソフトウェアが実行されている TDOMAIN タイプのドメイン・ゲートウェイにのみ適用されます。リモート・ドメイン・アクセス・ポイントに対するこの値は、ローカル・ドメイン・アクセス・ポイントに対する値に優先します。ローカル・ドメイン・アクセス・ポイント値を無効にできるので、リモート・ドメイン単位で TCP レベル・キープアライブをコンフィギュレーションできます。

"LOCAL" を指定すると、リモート・ドメイン・アクセス・ポイントは、ローカル・ドメイン・アクセス・ポイントに対して定義されている TCP レベル・キープアライブ値を受け入れます。"LOCAL" は、リモート・ドメイン・アクセス・ポイントに対するデフォルトの TCP レベル・キープアライブ値です。

"NO" を指定すると、このアクセス・ポイントに対する TCP レベル・キープアライブが無効になります。"NO" は、ローカル・ドメイン・アクセス・ポイントに対するデフォルトの TCP レベル・キープアライブ値です。

"YES" を指定すると、このアクセス・ポイントに対する TCP レベル・キープアライブが有効になります。接続の TCP レベル・キープアライブが有効になった場合、その接続のキープアライブ間隔は、オペレーティング・システムの TCP キープアライブ・タイマ用にコンフィギュレーションされているシステム・レベル値です。この間隔は、TDomain ゲートウェイが接続でトラフィックを受信せずに待機する最長時間です。この最長時間を超えると、ゲートウェイは TCP レベル・キープアライブ要求メッセージを送信します。接続がまだオープンしており、リモート TDomain ゲートウェイが正常に動作している場合、リモート・ゲートウェイは肯定応答を返信します。ローカル TDomain ゲートウェイは、要求メッセージを送信してから一定時間内に肯定応答を受信しなかった場合、接続が切断されたと見なして、その接続に関連するすべてのリソースを解放します。

TCP レベル・キープアライブを使用すると、BEA Tuxedo のドメイン間接続を非アクティブな期間にわたってオープンにできるだけでなく、TDomain ゲートウェイが接続の障害を迅速に検出できるようになります。

注記　TA_DMTCPKEEPALIVE と TA_DMKEEPALIVE は、相互に排他的ではありません。つまり、両方の属性を使用してドメイン間接続をコンフィギュレーションできます。

TA_DMKEEPALIVE = -1 <= *num* <= 2147483647

ローカルまたはリモート・ドメイン・アクセス・ポイントのアプリケーション・レベル・キープアライブを制御します。この値は、-1 以上 2147483647 以下でなければなりません。値 -1 は、リモート・ドメイン・アクセス・ポイントにのみ適用されます。

TA_DMKEEPALIVE 属性は、BEA Tuxedo 8.1 以降のソフトウェアが実行されている TDOMAIN タイプのドメイン・ゲートウェイにのみ適用されます。リモート・ドメイン・アクセス・ポイントに対するこの値は、ローカル・ドメイン・アクセス・ポイントに対する値に優先します。ローカル・ドメイン・アクセス・ポイント値を無効にできるので、リモート・ドメイン単位で アプリケーション・レベル・キープアライブをコンフィギュレーションできます。

-1 を指定すると、リモート・ドメイン・アクセス・ポイントは、ローカル・ドメイン・アクセス・ポイントに対して定義されているアプリケーション・レベル・キープアライブ値を受け入れます。-1 は、リモート・ドメイン・アクセス・ポイントに対するデフォルトのアプリケーション・レベル・キープアライブ値です。

0 を指定すると、このアクセス・ポイントに対するアプリケーション・レベル・キープアライブが無効になります。0 は、ローカル・ドメイン・アクセス・ポイントに対するデフォルトのアプリケーション・レベル・キープアライブ値です。

1 以上 2147483647 以下の値 (単位はミリ秒で、Domains ソフトウェアによって最も近い秒数に切り上げられる) を指定すると、このアクセス・ポイントに対するアプリケーション・レベル・キープアライブが有効になります。指定した時間は、TDomain ゲートウェイが接続でトラフィックを受信せずに待機する最長時間です。この最長時間を超えると、ゲートウェイはアプリケーション・レベル・キープアライブ要求メッセージを送信します。接続がまだオープンしており、リモート TDomain ゲートウェイが正常に動作している場合、リモート・ゲートウェイは肯定応答を返信します。ローカル TDomain ゲートウェイは、要求メッセージを送信してから指定の時間内 (TA_DMKEEPALIVEWAIT 属性を参照) に肯定応答を受信しなかった場合、接続が切断されたと見なして、その接続に関連するすべてのリソースを解放します。

アプリケーション・レベル・キープアライブを使用すると、BEA Tuxedo のドメイン間接続を非アクティブな期間にわたってオープンにできるだけでなく、TDomain ゲートウェイが接続の障害を迅速に検出できるようになります。

注記　TA_DMKEEPALIVE と TA_DMTCPKEEPALIVE は、相互に排他的ではありません。つまり、両方の属性を使用してドメイン間接続をコンフィギュレーションできます。

TA_DMKEEPALIVEWAIT = 0 <= *num* <= 2147483647

このローカルまたはリモート・ドメイン・アクセス・ポイントに関連付けられている TDomain ゲートウェイが送信したキープアライブ・メッセージに対する肯定応答を受信するまでの待ち時間を指定します。この値は、0 以上 2147483647 以下でなければなりません (単位はミリ秒で、Domains ソフトウェアによって最も近い秒数に切り上げられる)。デフォルトは 0 です。この属性は、BEA Tuxedo 8.1 以降のソフトウェアが実行されている TDOMAIN タイプのドメイン・ゲートウェイにのみ適用されます。

このアクセス・ポイントに対する TA_DMKEEPALIVE が 0 (キープアライブが無効) の場合、TA_DMKEEPALIVEWAIT の設定は無効です。

このアクセス・ポイントに対する TA_DMKEEPALIVE を有効にし、TA_DMKEEPALIVEWAIT を TA_DMKEEPALIVE より大きい値に設定した場合、ローカル TDomain ゲートウェイは TA_DMKEEPALIVEWAIT タイマが期限切れになるまでに複数のアプリケーション・レベル・キープアライブ・メッセージを送信します。このような設定の組み合わせも可能です。

このアクセス・ポイントの TA_DMKEEPALIVE が有効で、TA_DMKEEPALIVEWAIT が 0 に設定されている場合、送信したキープアライブ・メッセージに対する肯定応答を受信することは重要ではありません。そのような肯定応答はすべて TDomain ゲートウェイで無視されます。ゲートウェイは、TA_DMKEEPALIVE タイマがタイム・アウトするたびにキープアライブ・メッセージを送信します。この設定の組み合わせは、ファイアウォールを介したアイドル接続を保持するために使用します。

制限事項 以下の場合は、このクラスのインスタンスを削除したり、このクラスのインスタンスの TA_DMNWDEVICE 属性を更新したりすることはできません。

- このクラスのインスタンスがローカル・ドメイン・アクセス・ポイントに対応しており、ローカル・アクセス・ポイントのドメイン・ゲートウェイ管理 (GWADM) サーバがアクティブである場合。
- このクラスのインスタンスがリモート・ドメイン・アクセス・ポイントに対応しており、いずれかの Tdomain ドメイン・ゲートウェイ管理 (GWADM) サーバがアクティブである場合。

T_DM_TOPEND クラスの定義

概要 T_DM_TOPEND クラスは、ローカルまたはリモート・ドメイン・アクセス・ポイントに対する BEA TOP END システム固有のコンフィギュレーションを定義します。

属性表

表 29DM_MIB(5): T_DM_TOPEND クラス定義の属性表

属性	タイプ	パーミッション	値	デフォルト値
TA_DMACCESSPOINT(k)(r)(*)	string	rw-r--r--	*string*[1..30]	N/A
TA_DMNWADDR(r)(k)(*)	string	rw-r--r--	*string*[1..78]	N/A
TA_DMTE_TP_SYSTEM(r)	string	rw-r--r--	*string*[1..8]	
TA_STATE(r)	string	rw-r--r--	GET: "VAL" SET: "{NEW | INV | REC}"	N/A N/A
TA_DMNWDEVICE	string	rw-r--r--	*string*[1..78]	N/A
TA_DMTE_PWD	string	rwx------	*string*[1..12]	

(r)— 新しいオブジェクトが作成される場合に必須です。
(k)— オブジェクトを取り出すためのキー・フィールドです。
(*)— すべての SET 操作で必須のキー・フィールドです。

属性の意味

TA_DMACCESSPOINT: *string*[1..30]
このエントリが BEA TOP END 固有のコンフィギュレーション・データを提供するローカルまたはリモート・ドメイン・アクセス・ポイントの名前を指定します。

TA_DMNWADDR: *string*[1..78]
アクセス・ポイントに関連付けられたネットワーク・アドレスを指定します。ローカル・ドメイン・アクセス・ポイントの場合は、受信する接続指示を受け付けるためのアドレスを指定します。リモート・ドメイン・アクセス・ポイントの場合は、リモート・ドメイン・アクセス・ポイントに接続するために使用する接続先アドレスを指定しま

す。このフィールドの値は、すべての T_DM_TOPEND エントリ間で一意でなければなりません。

TA_DMTE_TP_SYSTEM: *string*[1..8]
BEA TOP END システムの名前を指定します。

注記　ローカル・ドメイン・アクセス・ポイントを通してアクセス可能なリモート・ドメイン・アクセス・ポイントの BEA TOP END システム名はすべて同じでなければなりません。

TA_STATE:

GET: "{VALid}"
GET 操作は、T_DM_TOPEND オブジェクトのコンフィギュレーション情報を検索します。以下に示す状態は、GET 要求に対する応答で返される TA_STATE 属性の意味を示します。以下に示されていない状態は返されません。

"VALid"	オブジェクトが存在します。

SET: "{NEW | INValid | RECrypt}"
SET 操作は、選択した T_DM_TOPEND オブジェクトのコンフィギュレーション情報を更新します。以下に示す状態は、SET 要求で設定される TA_STATE の意味を示します。以下に示されていない状態は設定できません。

"NEW"	新しいオブジェクトを作成します。この状態変更は、"INValid" 状態でのみ可能です。正常に終了すると、オブジェクトの状態は "VALid" になります。
unset	既存のオブジェクトを変更します。この組み合わせは "INValid" 状態では使用できません。正常に終了してもオブジェクトの状態は変わりません。
"INValid"	オブジェクトを削除します。この状態変更は、"VALid" 状態でのみ可能です。正常に終了すると、オブジェクトの状態は "INValid" になります。

"RECrypt"	暗号化キーを使用してすべてのパスワードを再暗号化します。T_DM_PASSWORD クラスおよび T_DM_TOPEND クラスのすべてのパスワード・インスタンスに適用されます。

TA_DMNWDEVICE: *string*[1..78]

このローカルまたはリモート・ドメイン・アクセス・ポイントに関連付けるネットワーク・デバイスを指定します。

TA_DMTE_PWD: *string*[1..12]

メッセージを BEA TOP END システムに送信する際に使用するパスワードを指定します。ローカル・ドメイン・アクセス・ポイント・エントリでのみ有効です。

T_DM_TRANSACTION クラスの定義

概要 T_DM_TRANSACTION クラスは、複数のドメインにまたがるトランザクションに関する実行時情報を表します。このオブジェクトを使用すると、トランザクションに関与しているリモート・ドメイン・アクセス・ポイント、親ドメイン・アクセス・ポイント、トランザクション状態、およびその他の情報を検索できます。

GET 操作では、特定のトランザクションを選択するために、TA_DMTPTRANID、TA_DMTXACCESSPOINT、および TA_DMTXNETTRANID 属性を指定できます。

属性表

表 30DM_MIB(5): T_DM_TRANSACTION **クラス定義の属性表**

属性	タイプ	パーミッション	値	デフォルト値
TA_DMLACCESSPOINT(k)(*)	string	rw-r--r--	*string*[1..30]	N/A
TA_DMTPTRANID(k)	string	rw-r--r--	*string*[1..78]	N/A
TA_STATE(r)(k)	string	rwxr-xr--	GET: "{ABD | ABY | ACT | COM | DEC | DON | HAB | HCO | HEU | REA | UNK}" SET: "INV"	N/A N/A
TA_DMTXACCESSPOINT(k)	string	r--r--r--	*string*[1..30]	N/A
TA_DMTXNETTRANID(k)	string	r--r--r--	*string*[1..78]	N/A
TA_DMBRANCHCOUNT	long	r--r--r--	0 <= *num*	N/A
TA_DMBRANCHINDEX	long	r--r--r--	0 <= *num*	N/A
ブランチごとの属性				
TA_DMBRANCHNO	long	r--r--r--	0 <= *num*	N/A
TA_DMRACCESSPOINT	string	r--r--r--	*string*[1..30]	N/A
TA_DMNETTRANID	string	r--r--r--	*string*[1..78]	N/A

表 30DM_MIB(5): T_DM_TRANSACTION クラス定義の属性表 (続き)

属性	タイプ	パーミッション	値	デフォルト値
TA_DMBRANCHSTATE	string	r--r--r--	GET: "{ABD | ABY | ACT | COM | DEC | DON | HAB | HCO | HHZ | HMI | REA | UNK}"	N/A

(r)— 新しいオブジェクトが作成される場合に必須です。
(k)— オブジェクトを取り出すためのキー・フィールドです。
(*)— すべての SET 操作で必須のキー・フィールドです。

属性の意味

TA_DMLACCESSPOINT: *string*[1..30]

トランザクションが関連付けられたローカル・ドメイン・アクセス・ポイントの名前。このフィールドは GET 操作では必須です。SET 操作では、TA_DMLACCESSPOINT を指定する必要があります。

TA_DMTPTRANID: *string*[1..78]

tpsuspend(3c) から返され、文字列表現にマップされるトランザクション識別子。等号比較の場合を除き、ユーザはこのフィールドのデータを直接解釈することはできません。

TA_STATE:

GET: "{ABorteD | ABortonlY | ACTive | COMcalled | DECided | DONe | HABort | HCOmmit | HEUristic | REAdy | UNKnown}"

GET 操作は、T_DM_TRANSACTION オブジェクトの実行時情報を検索します。以下に示す状態は、GET 要求に対する応答で返される TA_STATE 属性の意味を示します。以下に示されていない状態は返されません。

"ABorteD"	トランザクションはロールバックされています。
"ABortonlY"	トランザクションはロールバックされるものと識別されています。
"ACTive"	トランザクションはアクティブです。

"COMcalled"	トランザクションはコミットの第 1 フェーズを開始しました。
"DECided"	トランザクションはコミットの第 2 フェーズを開始しました。
"DONe"	トランザクションはコミットの第 2 フェーズを完了しました。
"HABort"	このトランザクションはヒューリスティックにロールバックされました。
"HCOmmit"	このトランザクションはヒューリスティックにコミットされました。
"HEUristic"	トランザクションのコミットまたはロールバックはヒューリスティックに完了しました。ブランチ状態によって、どのブランチがヒューリスティックに完了したかが分かります。
"REAdy"	このトランザクションでは、2 フェーズ・コミットの第 1 フェーズが完了しています。すべての参加グループおよびリモート・ドメインはコミットの第 1 フェーズを完了し、コミット可能な状態です。
"UNKnown"	トランザクションの状態を判別できませんでした。

SET:"{INValid}"

SET 操作は、選択した T_DM_TRANSACTION オブジェクトの実行時情報を更新します。以下に示す状態は、SET 要求で設定される TA_STATE の意味を示します。以下に示されていない状態は設定できません。

"INValid"	指定したトランザクション・オブジェクトを破棄します。状態の変更は、"HCOmmit"、"HABort"、および "HEUristic" 状態でのみ可能です。TA_DMTPTRANID 属性値を指定しない場合は、指定したローカル・ドメイン・アクセス・ポイントのすべてのヒューリスティック・トランザクション・ログ・レコードが破棄されます。

TA_DMTXACCESSPOINT: *string*[1..30]

トランザクションがリモート・ドメインから開始された場合、TA_DMTXACCESSPOINT はトランザクションの開始に使用したリモート・ドメイン・アクセス・ポイントの名前です。トランザクションがこのドメイン内から開始された場合、TA_DMTXACCESSPOINT はローカル・ドメイン・アクセス・ポイントの名前です。

TA_DMTXNETTRANID: *string*[1..78]

トランザクションがリモート・ドメインから開始された場合、TA_DMTXNETTRANID はトランザクションの開始に使用したリモート・ドメイン・アクセス・ポイントから受け取った外部トランザクション識別子です。トランザクションがこのドメイン内で開始された場合、TA_DMTXNETTRANID は TA_DMTPTRANID 属性と同じ値になります。

注記　この属性は、BEA Tuxedo リリース 7.1 以降を実行するゲートウェイでのみ使用できます。それ以前のリリースの BEA Tuxedo システムを実行するゲートウェイでは NULL 文字列 "" に設定されます。

TA_DMBRANCHCOUNT: 0 <= *num*

トランザクションに関与するリモート・ドメイン・アクセス・ポイントに対するブランチ数。ブランチ情報を使用できないドメイン・ゲートウェイの場合、この値はゼロになります。

TA_DMBRANCHINDEX: 0 <= *num*

このオブジェクトに対応する最初のブランチ固有の属性値 (TA_DMBRANCHNO、TA_DMRACCESSPOINT、TA_DMNETTRANID、および TA_DMBRANCHSTATE) のインデックス。

ブランチごとの属性

TA_DMBRANCHNO: 0 <= *num*

参加ブランチのブランチ番号 (ゼロから始まる)。

TA_DMRACCESSPOINT: *string*[1..30]

このブランチのリモート・ドメイン・アクセス・ポイントの名前。

TA_DMNETTRANID: *string*[1..78]

このブランチのリモート・ドメイン・アクセス・ポイントで使用する外部トランザクション識別子。ドメイン・ゲートウェイのタイプによっては、この情報が返されないことがあります。その場合、この属性は空文字列に設定されます。たとえば Tdomains では、リモート・ドメイン・アクセス・ポイントへのブランチに TA_DMTPTRANID のローカル・トランザクション識別子を使用し、この値を空文字列に設定します。

TA_DMBRANCHSTATE:

GET: "{ABD | ABY | ACT | COM | DEC | DON | HAB | HCO | HHZ | HMI | REA | UNK}"

GET 操作は、トランザクション・ブランチに関する実行時情報を検索します (特定のドメイン・ゲートウェイ・タイプで使用可能な場合)。

"ABorteD"	トランザクション・ブランチはロールバックされています。
"ABortonlY"	トランザクション・ブランチはロールバックされるものと識別されています。
"ACTive"	トランザクション・ブランチはアクティブです。
"COMcalled"	トランザクション・ブランチはコミットの第 1 フェーズを開始しました。
"DECided"	トランザクション・ブランチはコミットの第 2 フェーズを開始しました。
"DONe"	トランザクション・ブランチはコミットの第 2 フェーズを完了しました。

"HABort"	このトランザクションはヒューリスティックにロールバックされました。
"HCOmmit"	このトランザクションはヒューリスティックにコミットされました。
"Heuristic HaZard"	トランザクション・ブランチの通信は失敗しました。ロールバックが正常終了したかどうかは不明です。
"Heuristic MIxed"	トランザクション・ブランチのコミットまたはロールバックは完了しました。リモート・ドメインからのレポートによれば、コミットまたはロールバックに使用したリソースの一部の状態がトランザクションの結果と一致していません。
"REAdy"	このトランザクションでは、2 フェーズ・コミットの第 1 フェーズが完了しています。すべての参加グループおよびリモート・ドメインはコミットの第 1 フェーズを完了し、コミット可能な状態です。
"UNKnown"	トランザクションの状態を判別できませんでした。

注記　この属性は、BEA Tuxedo リリース 7.1 以降を実行するゲートウェイでのみ使用できます。それ以前のリリースの BEA Tuxedo システムを実行するゲートウェイでは "UNKnown" に設定されます。

制限事項　このオブジェクトは、管理者が明示的に作成するのではなく、マルチ・ドメイン・トランザクションの開始時に生成されます。このオブジェクトに対して管理者が実行できるのは、状態を "INValid" に設定して、ヒューリスティック・トランザクション・ログ・レコードを破棄することだけです。ほかの属性を設定することはできません。トランザクション状態を "INValid" に設定した場合、返されるバッファの状態は、ヒューリスティック・トランザクション・ログ・レコードが破棄される前のトランザクション状態であり、破棄後の状態ではありません。

GET および SET 操作では、TA_DMLACCESSPOINT 属性に特定のローカル・ドメイン・アクセス・ポイントを指定する必要があります。

GET 操作や SET 操作の実行時には、TA_DMLACCESSPOINT 属性で識別されるローカル・アクセス・ポイントのドメイン・ゲートウェイ管理 (GWADM) サーバがアクティブでなければなりません。アクティブでない場合は "not defined" エラーが返されます。

DM_MIB(5) に関する追加情報

ファイル

```
${TUXDIR}/include/tpadm.h
${TUXDIR}/udataobj/tpadm
```

関連項目

tpacall(3c), tpalloc(3c), tpcall(3c), tpdequeue(3c), tpenqueue(3c), tpgetrply(3c), tprealloc(3c), FML 関数の紹介 , Fadd、Fadd32(3fml), Fchg、Fchg32(3fml), Ffind、Ffind32(3fml), MIB(5), TM_MIB(5)

『BEA Tuxedo アプリケーション実行時の管理』

『BEA Tuxedo アプリケーションの設定』

『C 言語を使用した BEA Tuxedo アプリケーションのプログラミング』

『FML を使用した BEA Tuxedo アプリケーションのプログラミング』

EVENTS(5)

名前 EVENTS— システム生成イベントのリスト

機能説明 システム・イベント・モニタ機能は、システム・オペレータが把握する必要のある定義済みイベント (おもに異常終了) を検出して通知する機能です。各イベント・レポートは FML32 バッファです。このバッファには、イベントについて記述した共通フィールドと、そのイベントに関連のあるオブジェクトついて記述したその他のフィールドが含まれます。

BEA Tuxedo システムでは、システム容量の限界を定期的にチェックします。リソースを使い切っているか限界に近い場合は、システム WARN イベントまたは ERROR イベントが通知されます。これらのイベントは、条件が解消されるまで継続的に通知されます。

このリファレンス・ページでは、まず共通のイベント通知フィールドについて説明し、その後で BEA Tuxedo の現行リリースで検出できるシステム・イベントの一覧を示します。システム・イベント名はピリオド (.) で始まります。

制限事項 イベントの通知は、現時点では TM_MIB(5) で定義されるクラスに限定されています。イベントの通知では、MIB 情報ベースを使用します。「ローカル属性」の定義および可用性については、MIB(5) および TM_MIB(5) を参照してください。また、ローカル属性が使用できるかどうかは、アプリケーションのネットワーク内での通信状態によって異なる点に注意してください。

条件がごく短時間しか存在しない場合、システム容量の限界に関するイベント (たとえば、.SysMachineFullMaxgtt) は通知されないことがあります。

共通のイベント通知フィールド

TA_OPERATION:*string*
: このバッファがイベント通知バッファであることを示すリテラル文字列 EVT 。

TA_EVENT_NAME:*string*
: このイベントを特定するための文字列。システムが生成したイベントはすべて .Sys で始まります。

TA_EVENT_SEVERITY:*string*
: イベントの重要度を示す文字列 ERROR、WARN、または INFO。

TA_EVENT_LMID:*string*
: イベントが検出されたマシンを示す文字列。

TA_EVENT_TIME:*long*
: イベントを検出したマシンのクロックに基づくイベント検出時間 (秒) を示す long 型整数。

TA_EVENT_USEC:*long*
: イベントを検出したマシンのクロックに基づくイベント検出時間 (マイクロ秒) を示す long 型整数。この値の単位は常にマイクロ秒ですが、時間の実際の解像度は、使用しているオペレーティング・システムやハードウェアによって異なります。

TA_EVENT_DESCRIPTION:*string*
: イベントを要約した 1 行の文字列。

TA_CLASS:*string*
: イベントに関連のあるオブジェクトのクラス。TA_CLASS に応じて、このクラスのオブジェクトに固有の追加フィールドをイベント通知バッファに含めるかどうかが決まります。

TA_ULOGCAT:*string*
: メッセージがメッセージ・カタログから生成された場合は、そのカタログの名前。

TA_ULOGMSGNUM: *num*
: メッセージがカタログから生成された場合は、そのカタログのメッセージ番号。

イベント・リスト

T_ACLPERM イベント・リスト

```
.SysAclPerm
      INFO: .SysACLPerm: system ACL permission change
```

T_DOMAIN イベント・リスト

```
.SysResourceConfig
      INFO: .SysResourceConfig: system configuration change

.SysLicenseInfo
      INFO: .SysLicenseInfo: reached 100% of Tuxedo System
             Binary Licensed User Count, DBBL/BBL lockout canceled
```

```
                .SysLicenseInfo: reached 90% of Tuxedo System
                 Binary Licensed User Count

                .SysLicenseInfo: reached 90% of Tuxedo System
                 Binary Licensed User Count, DBBL/BBL lockout canceled

                .SysLicenseInfo: reached below 90% of Tuxedo System
                 Binary Licensed User Count, DBBL/BBL lockout canceled

SysLicenseWarn
      WARN: .SysLicenseWarn: reached 100% of Tuxedo System
             Binary Licensed User Count

SysLicenseError
      ERROR: .SysLicenseError: exceeded 110% of Tuxedo System
              Binary Licensed User Count, DBBL/BBL lockout occurs,
              no new clients can join the application

              .SysLicenseError: exceeded 110% of Tuxedo System
               Binary Licensed User Count, %hour, %minutes,
               %seconds left before DBBL/BBL lockout occurs
```

T_GROUP イベント・リスト

```
.SysGroupState
      INFO: .SysGroupState: system configuration change
```

T_MACHINE イベント・リスト

```
.SysMachineBroadcast
      WARN: .SysMachineBroadcast: %TA_LMID broadcast delivery
      failure

.SysMachineConfig
      INFO: .SysMachineConfig: %TA_LMID configuration change

.SysMachineFullMaxaccessers
      WARN: .SysMachineFullMaxaccessers: %TA_LMID capacity limit

.SysMachineFullMaxconv
      WARN: .SysMachineFullMaxconv: %TA_LMID capacity limit

.SysMachineFullMaxgtt
      WARN: .SysMachineFullMaxgtt: %TA_LMID capacity limit

.SysMachineFullMaxwsclients
      WARN: .SysMachineFullMaxwsclients: %TA_LMID capacity limit
```

```
.SysMachineMsgq
      WARN: .SysMachineMsgq: %TA_LMID message queue blocking

.SysMachinePartitioned
      ERROR: .SysMachinePartitioned: %TA_LMID is partitioned

.SysMachineSlow
      WARN: .SysMachineSlow: %TA_LMID slow responding to DBBL

.SysMachineState
      INFO: .SysMachineState: %TA_LMID state change to %TA_STATE

.SysMachineUnpartitioned
      ERROR: .SysMachinePartitioned: %TA_LMID is unpartitioned
```

T_BRIDGE イベント・リスト

```
.SysNetworkConfig
      INFO: .SysNetworkConfig: %TA_LMID[0]->%TA_LMID[1]
      configuration change

.SysNetworkDropped
      ERROR: .SysNetworkDropped: %TA_LMID[0]->%TA_LMID[1]
      connection dropped

.SysNetworkFailure
      ERROR: .SysNetworkFailure: %TA_LMID[0]->%TA_LMID[1]
      connection failure

.SysNetworkFlow
      WARN: .SysNetworkFlow: %TA_LMID[0]->%TA_LMID[1] flow control

.SysNetworkState
      INFO: .SysNetworkState: %TA_LMID[0]->%TA_LMID[1] state change
      to %TA_STATE
```

T_SERVER イベント・リスト

```
.SysServerCleaning
      ERROR: .SysServerCleaning: %TA_SERVERNAME, group %TA_SRVGRP,
      id %TA_SRVID server cleaning

.SysServerConfig
      INFO: .SysServerConfig: %TA_SERVERNAME, group %TA_SRVGRP, id
      %TA_SRVID configuration change
```

```
.SysServerDied
      ERROR: .SysServerDied: %TA_SERVERNAME, group %TA_SRVGRP, id
      %TA_SRVID server died

.SysServerInit
      ERROR: .SysServerInit: %TA_SERVERNAME, group %TA_SRVGRP, id
      %TA_SRVID server initialization failure

.SysServerMaxgen
      ERROR: .SysServerMaxgen: %TA_SERVERNAME, group %TA_SRVGRP, id
      %TA_SRVID server exceeded MAXGEN restart limit

.SysServerRestarting
      ERROR: .SysServerRestarting: %TA_SERVERNAME, group
      %TA_SRVGRP, id %TA_SRVID server restarting

.SysServerState
      INFO: .SysServerState: %TA_SERVERNAME, group %TA_SRVGRP, id
      %TA_SRVID state change to %TA_STATE

.SysServerTpexit
      ERROR: .SysServerTpexit: %TA_SERVERNAME, group %TA_SRVGRP, id
      %TA_SRVID server requested TPEXIT
```

T_SERVICE イベント・リスト

```
.SysServiceTimeout
      ERROR:.SysServiceTimeout: %TA_SERVERNAME, group %TA_SRVGRP,
      id %TA_SRVID server killed due to a service timeout
```

T_CLIENT イベント・リスト

```
.SysClientConfig
      INFO: .SysClientConfig: User %TA_USRNAME on %TA_LMID
      configuration change

.SysClientDied
      WARN: .SysClientDied: User %TA_USRNAME on %TA_LMID client died

.SysClientSecurity
      WARN: .SysClientSecurity: User %TA_USRNAME on %TA_LMID
      authentication failure

.SysClientState
      INFO: .SysClientState: User %TA_USRNAME on %TA_LMID state
      change to %TA_STATE
```

```
T_TRANSACTION イベント・リスト

.SysTransactionHeuristicAbort
      ERROR: .SysTransactionHeuristicAbort: Transaction %TA_GTRID
      in group %TA_GRPNO

.SysTransactionHeuristicCommit
      ERROR: .SysTransactionHeuristicCommit: Transaction %TA_GTRID
      in group %TA_GRPNO

T_EVENT イベント・リスト

.SysEventDelivery
      ERROR: .SysEventDelivery: System Event Monitor delivery
      failure on %TA_LMID

.SysEventFailure
      ERROR: .SysEventFailure: System Event Monitor subsystem
      failure on %TA_LMID
```

ファイル `${TUXDIR}/udataobj/evt_mib`

関連項目 MIB(5), TM_MIB(5)

EVENT_MIB(5) に関する追加情報

名前 EVENT_MIB― イベント・ブローカの管理情報ベース

形式

```
#include <tpadm.h>
#include <fml32.h>
#include <evt_mib.h>
```

機能説明 BEA Tuxedo イベント・ブローカ MIB は、イベント・ブローカで管理できるクラスの集合を定義します。

管理要求のフォーマットと管理応答の解釈を行うには、EVENT_MIB(5) を共通 MIB リファレンス・ページ MIB(5) と一緒に使用します。コンポーネント MIB のリファレンス・ページを使用し、MIB(5) の説明に従ってフォーマットした要求を使用すると、アクティブなアプリケーションの既存の ATMI インターフェイスの 1 つを通じて管理サービスを要求できます。EVENT_MIB(5) のすべてのクラス定義の追加情報については、285 ページの「EVENT_MIB(5) に関する追加情報」を参照してください。

EVENT_MIB は、次のクラスで構成されています。

表 31EVENT_MIB クラス

クラス名	属性
T_EVENT_CLIENT	任意通知をトリガするサブスクリプション
T_EVENT_COMMAND	システム・コマンドをトリガするサブスクリプション
T_EVENT_QUEUE	キュー・ベースの通知のサブスクリプション
T_EVENT_SERVICE	サーバ・ベースの通知のサブスクリプション
T_EVENT_USERLOG	userlog メッセージを書き込むためのサブスクリプション

これらのクラスの各オブジェクトは、単一のサブスクリプション要求を表します。

各のクラスのパターン表現 TA_EVENT_EXPR により、SYSTEM EVENT 要求を照会するか USER EVENT 要求を照会するが決まります。この決定は次のように行われます。

- TA_EVENT_EXPR または TA_EVENT_SERVER を指定しない単純な GET 要求では、常に SYSTEM EVENT 要求が照会されます。USER EVENT 要求は照会されません。
- TA_EVENT_EXPR を指定し TA_EVENT_SERVER を指定しない GET 要求では、表現が "\.". で始まる場合は SYSTEM EVENT 要求が照会されます。それ以外の場合は USER EVENT 要求が照会されます。
- TA_EVENT_SERVER の値として "SYSTEM" を指定した GET 要求では、SYSTEM EVENT 要求が照会されます。値に "USER" を指定すると、USER EVENT 要求が照会されます。

FML32 フィールド・テーブル

このリファレンス・ページで説明する属性のフィールド・テーブルは、BEA Tuxedo システム・ソフトウェアのルート・ディレクトリからの相対パスで指定される udataobj/evt_mib ファイルにあります。${TUXDIR}/udataobj ディレクトリは、FLDTBLDIR32 環境変数で指定されるコロン区切りのリストにアプリケーションによって追加される必要があり、フィールド・テーブル名 evt_mib は、FIELDTBLS32 環境変数で指定されるカンマ区切りのリストに追加される必要があります。

T_EVENT_CLIENT クラスの定義

概要 T_EVENT_CLIENT クラスは、クライアント・ベースの通知用にイベント・ブローカに登録するサブスクリプションの集まりを表します。

イベントが検出されると、そのイベントと各 T_EVENT_CLIENT オブジェクトを比較します。そのイベント名が TA_EVENT_EXPR 内の値と一致し、オプションのフィルタ規則が TRUE である場合、イベント・バッファは指定されたクライアントの任意通知型メッセージの処理ルーチンに送られます。

属性表

表 32T_EVENT_CLIENT クラス定義の属性表

属性	タイプ	パーミッション	値	デフォルト値
TA_EVENT_EXPR(r) (*)	string	R--R--R--	*string*[1..255]	N/A
TA_EVENT_FILTER(k)	string	R--R--R--	*string*[1..255]	なし
TA_EVENT_FILTER_BINARY(k)	carray	R--R--R--	*carray*[1..64000]	なし
TA_STATE(r)	string	R-xR-xR-x	GET: ACT	N/A
			SET: {NEW | INV}	N/A
TA_CLIENTID(r) (*)	string	R--R--R--	*string*[1..78]	N/A

(k)— オブジェクトを検索するためのキー・フィールド
(r)— 新しいオブジェクトを作成する際に必要なフィールド
(*)—GET/SET キー、SET 操作では 1 つ以上必要

パーミッションについては、MIB(5) を参照してください。

属性の意味

TA_EVENT_EXPR: *string*[1..255]
: イベント・パターン表現。この表現 (正規表現形式) により、どのイベント名がこのサブスクリプションに一致するかを制御します。

TA_EVENT_FILTER: *string*[1..255]
: イベント・フィルタ表現。この表現が存在する場合は、ポストされたバッファの内容に対して評価されます。この表現は TRUE と評価される必要があります。それ以外の場合、このサブスクリプションは一致しません。

TA_EVENT_FILTER_BINARY: *carray*[1..64000]

バイナリ (carray) 形式のイベント・フィルタ表現。TA_EVENT_FILTER と同じですが、任意のバイナリ・データを含むことができます。TA_EVENT_FILTER または TA_EVENT_FILTER_BINARY のどちらか 1 つのみを指定できます。

TA_STATE:

GET: ACTive

GET 操作は、一致した T_EVENT_CLIENT オブジェクトのコンフィギュレーション情報を検索します。

SET: {NEW | INValid}

SET 操作は、T_EVENT_CLIENT オブジェクトのコンフィギュレーション情報を更新します。以下に示す状態は、SET 要求で設定される TA_STATE の意味を示します。以下に示されていない状態は設定できません。

NEW	T_EVENT_CLIENT オブジェクトを作成します。正常に終了すると、オブジェクトの状態は ACTive になります。
INValid	T_EVENT_CLIENT オブジェクトを削除します。正常終了すると、オブジェクトの状態は INValid になります。

TA_CLIENTID: *string*[1..78]

一致するイベントが検出されると、このクライアントに任意通知型メッセージを送信します。

T_EVENT_COMMAND クラスの定義

概要 T_EVENT_COMMAND クラスは、システム・コマンドの実行をトリガするイベント・ブローカに登録するサブスクリプションの集まりを表します。イベントが検出されると、そのイベントと各 T_EVENT_COMMAND オブジェクトを比較します。そのイベント名が TA_EVENT_EXPR 内の値と一致し、オプションのフィルタ規則が TRUE である場合、イベント・バッファはフォーマットされてシステムのコマンド・インタープリタに渡されます。

属性表

表 33T_EVENT_COMMAND クラス定義の属性表

属性	タイプ	パーミッション	値	デフォルト値
TA_EVENT_EXPR(r) (*)	string	R--------	*string*[1..255]	N/A
TA_EVENT_FILTER(k)	string	R--------	*string*[1..255]	なし
TA_EVENT_FILTER_BINARY(k)	carray	R--------	*carray*[1..64000]	なし
TA_STATE(r)	string	R-x------	GET: ACT	N/A
			SET: {NEW | INV}	N/A
TA_COMMAND(r) (*)	string	R--------	*string*[1..255]	N/A

(k)― オブジェクトを検索するためのキー・フィールド
(r)― 新しいオブジェクトを作成する際に必要なフィールド
(*)―GET/SET キー、SET 操作では 1 つ以上必要

パーミッションについては、MIB(5) を参照してください。

属性の意味

TA_EVENT_EXPR: *string*[1..255]
イベント・パターン表現。この表現 (正規表現形式) により、どのイベント名がこのサブスクリプションに一致するかを制御します。

TA_EVENT_FILTER: *string*[1..255]
イベント・フィルタ表現。この表現が存在する場合は、ポストされたバッファの内容に対して評価されます。この表現は TRUE と評価される必要があります。それ以外の場合、このサブスクリプションは一致しません。

TA_EVENT_FILTER_BINARY: *carray*[1..64000]

バイナリ (carray) 形式のイベント・フィルタ表現。TA_EVENT_FILTER と同じですが、任意のバイナリ・データを含むことができます。TA_EVENT_FILTER または TA_EVENT_FILTER_BINARY のどちらか 1 つのみを指定できます。

TA_STATE:

GET: ACTive

GET 操作は、一致した T_EVENT_COMMAND オブジェクトのコンフィギュレーション情報を検索します。

SET: {NEW | INValid}

SET 操作は、T_EVENT_COMMAND オブジェクトのコンフィギュレーション情報を更新します。以下に示す状態は、SET 要求で設定される TA_STATE の意味を示します。以下に示されていない状態は設定できません。

NEW	T_EVENT_COMMAND オブジェクトを作成します。正常に終了すると、オブジェクトの状態は ACTive になります。
INValid	T_EVENT_COMMAND オブジェクトを削除します。正常終了すると、オブジェクトの状態は INValid になります。

TA_COMMAND: *string*[1..255]

このオブジェクトと一致するイベントが検出されると、このシステム・コマンドを実行します。このコマンドは、UNIX システム・プラットフォームでは system(3) を使用してバックグラウンドで実行されます。

T_EVENT_QUEUE クラスの定義

概要 T_EVENT_QUEUE クラスは、キュー・ベースの通知用にイベント・ブローカに登録するサブスクリプションの集まりを表します。イベントが検出されると、そのイベントと各 T_EVENT_QUEUE オブジェクトを比較します。そのイベント名が TA_EVENT_EXPR 内の値と一致し、オプションのフィルタ規則が TRUE である場合、イベント・バッファは指定された信頼性の高いキューに格納されます。

属性表

表 34T_EVENT_QUEUE クラス定義の属性表

属性	タイプ	パーミッション	値	デフォルト値
TA_EVENT_EXPR(r) (*)	string	R--------	*string*[1..255]	N/A
TA_EVENT_FILTER(k)	string	R-x------	*string*[1..255]	なし
TA_EVENT_FILTER_BINARY(k)	carray	R-x------	*carray*[1..64000]	なし
TA_STATE(r)	string	R-x------	GET: ACT	N/A
			SET: {NEW | INV}	N/A
TA_QSPACE(r) (*)	string	R--------	*string*[1..15]	N/A
TA_QNAME(r) (*)	string	R--------	*string*[1..15]	N/A
TA_QCTL_QTOP	short	R-x------	*short*	0
TA_QCTL_BEFOREMSGID	short	R-x------	*short*	0
TA_QCTL_QTIME_ABS	short	R-x------	*short*	0
TA_QCTL_QTIME_REL	short	R-x------	*short*	0
TA_QCTL_DEQ_TIME	short	R-x------	*short*	0
TA_QCTL_PRIORITY	short	R-x------	*short*	0
TA_QCTL_MSGID	string	R-x------	*string*[1..31]	なし
TA_QCTL_CORRID(k)	string	R-x------	*string*[1..31]	なし
TA_QCTL_REPLYQUEUE	string	R-x------	*string*[1..15]	なし
TA_QCTL_FAILUREQUEUE	string	R-x------	*string*[1..15]	なし
TA_EVENT_PERSIST	short	R-x------	*short*	0
TA_EVENT_TRAN	short	R-x------	*short*	0

表 34T_EVENT_QUEUE **クラス定義の属性表**

属性	タイプ	パーミッション	値	デフォルト値

(k)— オブジェクトを検索するためのキー・フィールド
(r)— 新しいオブジェクトを作成する際に必要なフィールド
(*)—GET/SET キー、SET 操作では 1 つ以上必要

パーミッションについては、MIB(5) を参照してください。

属性の意味

TA_EVENT_EXPR: *string*[1..255]
イベント・パターン表現。この表現 (正規表現形式) により、どのイベント名がこのサブスクリプションに一致するかを制御します。

TA_EVENT_FILTER: *string*[1..255]
イベント・フィルタ表現。この表現が存在する場合は、ポストされたバッファの内容に対して評価されます。この表現は TRUE と評価される必要があります。それ以外の場合、このサブスクリプションは一致しません。

TA_EVENT_FILTER_BINARY: *carray*[1..64000]
バイナリ (carray) 形式のイベント・フィルタ表現。TA_EVENT_FILTER と同じですが、任意のバイナリ・データを含むことができます。TA_EVENT_FILTER または TA_EVENT_FILTER_BINARY のどちらか 1 つのみを指定できます。

TA_STATE:

GET: ACTive
GET 操作は、一致した T_EVENT_QUEUE オブジェクトのコンフィギュレーション情報を検索します。

SET: {NEW | INValid}
SET 操作は、T_EVENT_QUEUE オブジェクトのコンフィギュレーション情報を更新します。以下に示す状態は、SET 要求で設定される TA_STATE の意味を示します。以下に示されていない状態は設定できません。

NEW	T_EVENT_QUEUE オブジェクトを作成します。正常に終了すると、オブジェクトの状態は ACTive になります。
INValid	T_EVENT_QUEUE オブジェクトを削除します。正常終了すると、オブジェクトの状態は INValid になります。

TA_QSPACE: *string*[1..15]
一致するイベントが検出されると、通知メッセージをこのキュー・スペースの信頼性のあるキューに登録します。

TA_QNAME: *string*[1..15]
一致するイベントが検出されると、通知メッセージをこの信頼性のあるキューに登録します。

TA_QCTL_QTOP:*short*
この値は、tpenqueue() の TPQCTL 制御構造体に渡されます (値が設定されている場合のみ)。これにより、/Q サブシステム経由で通知が要求され、メッセージがキューの先頭に登録されます。

TA_QCTL_BEFOREMSGID:*short*
この値は、tpenqueue() の TPQCTL 制御構造体に渡されます (値が設定されている場合のみ)。これにより、/Q サブシステム経由で通知が要求され、メッセージがキュー内の指定したメッセージの前に登録されます。

TA_QCTL_QTIME_ABS:*short*
この値は、tpenqueue() の TPQCTL 制御構造体に渡されます (値が設定されている場合のみ)。これにより、/Q サブシステム経由で通知が要求され、メッセージが指定した時間に処理されます。

TA_QCTL_QTIME_REL:*short*
この値は、tpenqueue() の TPQCTL 制御構造体に渡されます (値が設定されている場合のみ)。これにより、/Q サブシステム経由で通知が要求され、キューからの取り出し時からの相対時間にメッセージが処理されます。

TA_QCTL_DEQ_TIME:*short*
: この値は、tpenqueue() の TPQCTL 制御構造体に渡されます (値が設定されている場合のみ)。

TA_QCTL_PRIORITY:*short*
: この値は、tpenqueue() の TPQCTL 制御構造体に渡されます (値が設定されている場合のみ)。

TA_QCTL_MSGID: *string*[1..31]
: この値は、tpenqueue() の TPQCTL 制御構造体に渡されます (値が設定されている場合のみ)。

TA_QCTL_CORRID: *string*[1..31]
: この値は、tpenqueue() の TPQCTL 制御構造体に渡されます (値が設定されている場合のみ)。

TA_QCTL_REPLYQUEUE: *string*[1..15]
: この値は、tpenqueue() の TPQCTL 制御構造体に渡されます (値が設定されている場合のみ)。

TA_QCTL_FAILUREQUEUE: *string*[1..15]
: この値は、tpenqueue() の TPQCTL 制御構造体に渡されます (値が設定されている場合のみ)。

TA_EVENT_PERSIST:*short*
: ゼロ以外の値が設定されている場合は、指定したキューが使用できなくなっていても、このサブスクリプションはキャンセルされません。

TA_EVENT_TRAN:*short*
: ゼロ以外の値が設定されており、クライアントの tppost() 呼び出しがトランザクションに関与している場合、クライアントのトランザクションに tpenqueue() 呼び出しを含めます。

T_EVENT_SERVICE クラスの定義

概要 T_EVENT_SERVICE クラスは、サービス・ベースの通知用にイベント・ブローカに登録するサブスクリプションの集まりを表します。イベントが検出されると、そのイベントと各 T_EVENT_SERVICE オブジェクトを比較します。そのイベント名が TA_EVENT_EXPR 内の値と一致し、オプションのフィルタ規則が TRUE である場合、イベント・バッファは指定された BEA Tuxedo サービス・ルーチンに送られます。

属性表

表 35T_EVENT_SERVICE クラス定義の属性表

属性	タイプ	パーミッション	値	デフォルト値
TA_EVENT_EXPR(r) (*)	string	R--R--R--	*string*[1. .255]	N/A
TA_EVENT_FILTER(k)	string	R--R--R--	*string*[1. .255]	なし
TA_EVENT_FILTER_BINARY(k)	carray	R--R--R--	*carray*[1. .64000]	なし
TA_STATE(r)	string	R-xR-xR-x	GET: ACT	N/A
			SET: {NEW | INV}	N/A
TA_SERVICENAME(r) (*)	string	R--R--R--	*string*[1..15]	N/A
TA_EVENT_PERSIST	short	R-xR-xR-x	*short*	0
TA_EVENT_TRAN	short	R-xR-xR-x	*short*	0

(k)― オブジェクトを検索するためのキー・フィールド
(r)― 新しいオブジェクトを作成する際に必要なフィールド
(*)―GET/SET キー、SET 操作では 1 つ以上必要

パーミッションについては、MIB(5) を参照してください。

属性の意味 TA_EVENT_EXPR: *string*[1..255]
イベント・パターン表現。この表現 (正規表現形式) により、どのイベント名がこのサブスクリプションに一致するかを制御します。

TA_EVENT_FILTER: *string*[1..255]

イベント・フィルタ表現。この表現が存在する場合は、ポストされたバッファの内容に対して評価されます。この表現は TRUE と評価される必要があります。それ以外の場合、このサブスクリプションは一致しません。

TA_EVENT_FILTER_BINARY: *carray*[1..64000]

バイナリ (carray) 形式のイベント・フィルタ表現。TA_EVENT_FILTER と同じですが、任意のバイナリ・データを含むことができます。TA_EVENT_FILTER または TA_EVENT_FILTER_BINARY のどちらか 1 つのみを指定できます。

TA_STATE:

GET: ACTive

GET 操作は、一致した T_EVENT_SERVICE オブジェクトのコンフィギュレーション情報を検索します。

SET: {NEW | INValid}

SET 操作は、T_EVENT_SERVICE オブジェクトのコンフィギュレーション情報を更新します。以下に示す状態は、SET 要求で設定される TA_STATE の意味を示します。以下に示されていない状態は設定できません。

NEW	T_EVENT_SERVICE オブジェクトを作成します。正常に終了すると、オブジェクトの状態は ACTive になります。
INValid	T_EVENT_SERVICE オブジェクトを削除します。正常終了すると、オブジェクトの状態は INValid になります。

TA_SERVICENAME: *string*[1..15]

一致するイベントが検出されると、この BEA Tuxedo サービスが呼び出されます。

TA_EVENT_PERSIST:*short*

ゼロ以外の値が設定されている場合は、TA_SERVICENAME サービスが使用できなくなっていても、このサブスクリプションはキャンセルされません。

TA_EVENT_TRAN:*short*

ゼロ以外の値が設定されており、クライアントの tppost() 呼び出しがトランザクションに関与している場合、クライアントのトランザクションに TA_SERVICENAME サービス呼び出しを含めます。

T_EVENT_USERLOG クラスの定義

概要 T_EVENT_USERLOG クラスは、システム userlog(3c) メッセージを書き込むためにイベント・ブローカに登録するサブスクリプションの集まりを表します。イベントが検出されると、そのイベントと各 T_EVENT_USERLOG オブジェクトを比較します。そのイベント名が TA_EVENT_EXPR 内の値と一致し、オプションのフィルタ規則が TRUE である場合、イベント・バッファはフォーマットされて BEA Tuxedo userlog(3c) 関数に渡されます。

属性表

表 36T_EVENT_USERLOG クラス定義の属性表

属性	タイプ	パーミッション	値	デフォルト値
TA_EVENT_EXPR(r)	string	R--R-----	*string*[1..255]	N/A
TA_EVENT_FILTER(k)	string	R--R-----	*string*[1..255]	なし
TA_EVENT_FILTER_BINARY(k)	carray	R--R-----	*carray*[1..64000]	なし
TA_STATE(r)	string	R-xR-x---	GET: ACT	N/A
			SET: {NEW | INV}	N/A
TA_USERLOG(r)	string	R--R-----	*string*[1..255]	N/A

(k)— オブジェクトを検索するためのキー・フィールド
(r)— 新しいオブジェクトを作成する際に必要なフィールド

パーミッションについては、MIB(5) を参照してください。

属性の意味

TA_EVENT_EXPR: *string*[1..255]
イベント・パターン表現。この表現 (正規表現形式) により、どのイベント名がこのサブスクリプションに一致するかを制御します。

TA_EVENT_FILTER: *string*[1..255]
イベント・フィルタ表現。この表現が存在する場合は、ポストされたバッファの内容に対して評価されます。この表現は TRUE と評価される必要があります。それ以外の場合、このサブスクリプションは一致しません。

TA_EVENT_FILTER_BINARY: *carray*[1..64000]
バイナリ (carray) 形式のイベント・フィルタ表現。TA_EVENT_FILTER と同じですが、任意のバイナリ・データを含むことができます。TA_EVENT_FILTER または TA_EVENT_FILTER_BINARY のどちらか 1 つのみを指定できます。

TA_STATE:

GET: ACTive
GET 操作は、一致した T_EVENT_USERLOG オブジェクトのコンフィギュレーション情報を検索します。

SET: {NEW | INValid}
SET 操作は、T_EVENT_USERLOG オブジェクトのコンフィギュレーション情報を更新します。以下に示す状態は、SET 要求で設定される TA_STATE の意味を示します。以下に示されていない状態は設定できません。

NEW	T_EVENT_USERLOG オブジェクトを作成します。正常に終了すると、オブジェクトの状態は ACTive になります。
INValid	T_EVENT_USERLOG オブジェクトを削除します。正常終了すると、オブジェクトの状態は INValid になります。

TA_USERLOG: *string*[1..255]
一致するイベントが検出されると、この userlog(3c) サービスが書き込まれます。

EVENT_MIB(5) に関する追加情報

ファイル　`${TUXDIR}/udataobj/evt_mib ${TUXDIR}/include/evt_mib.h`

関連項目　EVENTS(5), TM_MIB(5)

factory_finder.ini(5)

名前 factory_finder.ini—FactoryFinder の Dmains コンフィギュレーション・ファイル

機能説明 factory_finder.ini は、Domains 用の FactoryFinder コンフィギュレーション・ファイルです。このテキスト (ASCII) ファイルは、Master NameManager として起動された場合に TMFFNAME サービスによって解析されます。NameManager は、このファイルに格納されている情報を使用して、ほかのドメインとの間でのファクトリ・オブジェクトのオブジェクト参照のインポートやエクスポートを制御します。factory_finder.ini ファイルの情報を使用するには、TMFFNAME サーバ・プロセスの -f オプションで factory_finder.ini ファイルを指定する必要があります。

FactoryFinder Domains コンフィギュレーション・ファイルにはどのような名前を付けることもできますが、ファイルの内容はこのリファレンス・ページで説明する形式に準拠している必要があります。

定義 BEA Tuxedo ドメインは、単一の TUXCONFIG ファイルに記述された環境として定義されます。BEA Tuxedo ドメインは、別の BEA Tuxedo ドメインや別の TP アプリケーション (別の TP システムで実行されているアプリケーション) と、ドメイン・ゲートウェイ・グループを介して通信できます。BEA Tuxedo 用語では、ドメインとアプリケーション (ビジネス・アプリケーション) は同義です。

リモート・ファクトリは、リモート・ドメイン内に存在するファクトリ・オブジェクトです。アプリケーションでは、BEA Tuxedo FactoryFinder を介してこのオブジェクトを使用できます。

ローカル・ファクトリは、ローカル・ドメイン内に存在するファクトリ・オブジェクトです。リモート・ドメインでは、BEA Tuxedo FactoryFinder を介してこのオブジェクトを使用できます。

ファイル形式 このファイルは、2 つの仕様セクションで構成されます。使用可能なセクションは、DM_REMOTE_FACTORIES および DM_LOCAL_FACTORIES です。

- フォーマット処理のガイドライン

通常、パラメータは *KEYWORD = value* で指定します。これにより、*KEYWORD* が *value* に設定されます。有効なキーワードについては、以下の各セクションで説明します。*KEYWORD* は予約されているため、引用符が付いている場合を除き、値としては使用できません。

値が識別子の場合は、標準 C の規則が適用されます。標準 C の識別子の先頭には英字またはアンダースコア (_) を使用し、以降の識別子には英数字またはアンダースコアを使用する必要があります。*KEYWORD* と同じ identifier を使用することはできません。

値が識別子でない場合は、二重引用符で囲まなければなりません。

入力フィールドは、1 つ以上の空白またはタブ文字で区切ります。

文字はコメントを示します。復帰改行文字でコメントを終了します。

空白行とコメントは無視されます。

行は、復帰改行の後に最低 1 つのタブを置いて継続できます。コメントを継続することはできません。

- DM_LOCAL_FACTORIES セクション

  このセクションは、各ローカル・ドメインによってエクスポートされたファクトリに関する情報を提供します。このセクションはオプションで、指定されていない場合はすべてのローカル・ファクトリをリモート・ドメインにエクスポートできます。このセクションを指定することにより、リモート・ドメインから取得できるローカル・ファクトリ・オブジェクトのセットが制限されます。予約されている *factory_id.factory_kind* 識別子 NONE を使用して、リモード・ドメインから取得するローカル・ファクトリを制限できます。

  このセクション内にある各行の形式は次のとおりです。

  factory_id.factory_kind

  factory_id.factory_kind は、ファクトリのローカル名 (識別子) です。この名前は、1 つまたは複数の BEA Tuxedo サーバ・アプリケーションが BEA Tuxedo FactoryFinder を使用して登録したファクトリ・オブジェクトの識別子に対応している必要があります。

  TMFFNAME で適切なファクトリを検索するには、*factory_kind* が指定されていなければなりません。*factory_kind* が指定されていないエントリには、デフォルト値の FactoryInterface が適用されます。

■ DM_REMOTE_FACTORIES セクション

このセクションは、インポートされてリモート・ドメインで使用可能になったファクトリ・オブジェクトに関する情報を提供します。このセクション内にある各行の形式は次のとおりです。

factory_id.factory_kind required_parameters

factory_id.factory_kind は、ローカルの BEA Tuxedo システム・ドメインが特定のリモート・ファクトリ・オブジェクトに使用する名前 (識別子) です。リモート・ファクトリ・オブジェクトは、特定のリモート・ドメインに関連付けられています。

注記　TobjFactoryFinder インターフェイスを使用する場合、*factory_kind* は FactoryInterface でなければなりません。

以下は、必須パラメータです。

DOMAINID = *domain_id*

このパラメータは、ファクトリ・オブジェクトを検索するリモート・ドメインの ID を指定します。*domain_id* の長さは 32 オクテット以内です。文字列を指定する場合は、32 文字以内で指定する必要があります (最後の NULL を含む)。*domain_id* の値は、一連の文字か、または 0x で始まる 16 進数です。

以下は、オプション・パラメータです。

RNAME = *string*

このパラメータは、リモート・ドメインによってエクスポートされる名前を指定します。リモート・ドメインでは、この名前を使用してこのファクトリ・オブジェクトを要求します。このパラメータを指定しない場合、リモート・ファクトリ・オブジェクト名は *factory_id.factory_kind* で指定した名前と同じになります。

DOMAINID または RNAME パラメータのどちらかに関連する値で固有のファクトリ・オブジェクトが識別できれば、同じ名前の複数のエントリを指定することができます。

使用例　■ 例 1

次の FactoryFinder Domains コンフィギュレーション・ファイルは、あるファクトリ・オブジェクトの 2 つのエントリを定義しています。このファクトリ・オブジェクトは、ローカル・ドメインでは

Teller.FactoryIdentity という識別子で識別され、2 つの異なるリモート・ドメインからインポートされます。

```
# BEA Tuxedo FactoryFinder Domains
# コンフィギュレーション・ファイル
#
*DM_REMOTE_FACTORIES
  Teller.FactoryIdentity
    DOMAINID="Northwest"
    RNAME=Teller.FactoryType
  Teller.FactoryIdentity
    DOMAINID="Southwest"
```

最初のエントリでは、Teller.FactoryType というファクトリ ID で登録された Northwest という ID のリモート・ドメインから、ファクトリ・オブジェクトがインポートされます。

2 番目のエントリでは、Teller.FactoryIdentity というファクトリ ID で登録された Southwest という ID のリモート・ドメインから、ファクトリ・オブジェクトがインポートされます。RNAME パラメータが指定されていないため、リモート・ドメインのファクトリ・オブジェクトの名前は、ローカル・ドメインのファクトリの名前と同じとみなされます。

- 例 2

次に示す FactoryFinder Domains コンフィギュレーション・ファイルの定義では、ローカル・ドメインの Teller.FactoryInterface という ID で登録されたファクトリ・オブジェクトのみをリモート・ドメインにエクスポートできます。ほかのファクトリに対する要求は拒否されます。

```
# BEA Tuxedo FactoryFinder Domains
# コンフィギュレーション・ファイル
#
*DM_LOCAL_FACTORIES
  Teller.FactoryInterface
```

- 例 3

次に示す FactoryFinder Domains コンフィギュレーション・ファイルでは、BEA Tuxedo FactoryFinder で登録されたファクトリ・オブジェクトがリモート・ドメインにエクスポートされないように定義されています。

```
# BEA Tuxedo FactoryFinder Domains
 # コンフィギュレーション・ファイル
#
```

```
                *DM_LOCAL_FACTORIES
                  NONE
```

関連項目　UBBCONFIG(5), DMCONFIG(5), TMFFNAME(5), TMIFRSVR(5)

Ferror、Ferror32(5)

名前 Ferror、Ferror32—FML エラー・コード

形式
```
#include "fml.h"
#include "fml32.h"
```

機能説明 エラー条件のシンボル名によって表される数値は、多くの FML ライブラリ・ルーチンの実行時に発生するエラー用の Ferror に割り当てられます。

名前 Ferror は、タイプ int を持つ変更可能な *lvalue* に拡張できます。この値は、FML ライブラリ・ルーチンによって正のエラー番号に設定されます。Ferror は、オブジェクトの識別子である必要はなく、関数呼び出しによって生じる変更可能な *lvalue* に拡張されます。Ferror がマクロであるかまたは外部リンクで宣言される識別子であるかは特定されていません。実際のオブジェクトをアクセスするための tperrno() マクロの定義が抑止されている場合、または、あるプログラムが名前 Ferror を使用して識別子を定義している場合、動作は不確定です。

FML ルーチンのリファレンス・ページには、各ルーチンのエラー条件とそのコンテキストにおけるエラーの意味が掲載されています。掲載されているエラーの順番は重要ではなく、優先順位を示すものでもありません。Ferror の値は、エラーが指摘された後にのみ検査します。すなわち、構成要素の戻り値がエラーを示していて、構成要素の定義で tperrno() のエラー時の設定が指定されている場合です。Ferror の値を検査するアプリケーションは、ヘッダ・ファイル fml.h をインクルードしなければなりません。

Ferror32 は、同様の機能を FML32 ルーチンのユーザに提供します。Ferror32 の値を検査するアプリケーションは、ヘッダ・ファイル fml32.h をインクルードしなければなりません。

以下に、FML および FML32 のルーチンが返すエラー・コードを示します。
```
#define FMINVAL 0   /* エラー・メッセージ・コードの最後尾 */
#define FALIGNERR 1  /* フィールド化バッファは整列しない */
#define FNOTFLD 2   /* バッファはフィールド化しない */
#define FNOSPACE 3   /* フィールド化バッファにスペースを入れない */
#define FNOTPRES 4   /* フィールドは存在しない */
#define FBADFLD 5   /* 不明なフィールド番号またはタイプ */
#define FTYPERR 6   /* 不正なフィールド・タイプ */
```

```
#define FEUNIX 7    /* Unix システム呼び出しエラー */
#define FBADNAME 8   /* 不明なフィールド名 */
#define FMALLOC 9   /* malloc が失敗 */
#define FSYNTAX 10   /* 論理式の構文が不正 */
#define FFTOPEN 11   /* フィールド・テーブルが見つからないかオープンできな
い */
#define FFTSYNTAX 12  /* フィールド・テーブル内に構文エラーがある */
#define FEINVAL 13   /* 関数に対する引数が無効 */
#define FBADTBL 14   /* フィールド･テーブルに対する破壊的な同時アクセス */
#define FBADVIEW 15  /* ビューが見つからないか取得できない */
#define FVFSYNTAX 16  /* 不正なビューファイル */
#define FVFOPEN 17   /* ビューファイルが見つからないかオープンできない */
#define FBADACM 18   /* ACM に負の値が含まれている */
#define FNOCNAME 19   /* cname が見つからない */
```

使用法 ルーチンには、エラーの戻り値がないものもあります。Ferror をゼロに設定するルーチンはないため、アプリケーションは、Ferror をゼロに設定し、ルーチンを呼び出してから、エラーが発生したかを調べるために再度 Ferror をチェックできます。

この変数は、DOS および OS/2 環境では FMLerror です。

関連項目 個々の FML ライブラリ・ルーチンの ERRORS の項を参照してください。

C 言語アプリケーション・トランザクション・モニタ・インターフェイスについて , tperrordetail(3c), tpstrerror(3c), tpstrerrordetail(3c), FML 関数の紹介 , F_error、F_error32(3fml)

field_tables(5)

名前 field_tables— フィールド名に対する FML マッピング・ファイル

機能説明 フィールド操作言語 (FML) の関数は、フィールド化バッファのインプリメントと管理を行います。フィールド化バッファの各フィールドには、short 整数のタグを付けます。可変長フィールド (文字列など) には、長さを示す修飾子を付けます。したがって、フィールド化バッファは、数値識別子 / データの組み合わせ、または数値識別子 / 長さ / データの組み合わせから構成されることになります。

フィールドの数値識別子をそのフィールドのフィールド識別子といい、FLDID によりそのタイプを定義します。フィールドの名前は、フィールド・テーブルの FLDID と英数字の文字列 (名前) を組み合わせて指定します。

従来の FML インターフェイスでは、16 ビットのフィールド識別子、フィールド長、およびバッファ・サイズをサポートします。新しい 32 ビット・インターフェイスである FML32 では、より大きい識別子、フィールド長、およびバッファ・サイズをサポートします。すべてのタイプや関数名などには、接尾辞として "32" (たとえば、フィールド識別子タイプ定義は FLDID32) を付けます。

フィールド識別子 FML 関数では、フィールド値のタイプを決めることができます。現在サポートされているタイプは、char、string、short、long、float、double、carray (文字配列)、ptr (バッファへのポインタ)、FML32 (埋め込み型の FML32 バッファ)、および VIEW32 (埋め込み型の VIEW32 バッファ) です。ptr 型、FML32 型、および VIEW32 型は、FML32 インターフェイスでのみサポートされています。フィールド・タイプの定数は fml.h (FML32 では fml32.h) で定義します。フィールド化バッファは完全な自己記述型であるため、フィールドのタイプは FLDID でエンコードされてフィールドとともに渡されます。したがって、FLDID はフィールド・タイプとフィールド番号という 2 つの要素から構成されます。フィールド番号は、100 よりも大きくなければなりません。これは、1 から 100 までの番号がシステムで予約されているためです。

フィールド・マッピング

フィールド名からフィールド識別子へのマッピングは、コンパイル時に実行できると効率的です。また、実行時にもマッピングできればより実用的です。この両方を満たすため、FML ではテキスト・ファイルにフィールド・テーブルを保持し、対応する C ヘッダ・ファイルを生成するコマンドも用意されています。これにより、コンパイル時のマッピングは C プリプロセッサ (cpp) で通常の `#define` マクロを使用して実行でき、実行時のマッピングは関数 `Fldid()` (FML32 では `Fldid32()`) で実行できます。この関数は、ソース・フィールド・テーブル・ファイルを参照して、その引数 (フィールド名) をフィールド識別子にマップするものです。

フィールド・テーブル・ファイル

フィールド・テーブルを格納しているファイルの形式は以下のとおりです。

- # で始まる行および空白行は無視されます。
- $ で始まる行は、マッピング関数では無視されますが、`mkfldhdr` で生成されたヘッダ・ファイルに渡されます。このとき $ は除去されます (FML32 ではコマンド名は `mkfldhdr32()`。`mkfldhdr、mkfldhdr32(1)` を参照)。これにより、たとえば C 言語のコメントや `what` 文字列などを、生成されたヘッダ・ファイルに渡すことができます。
- 文字列 `*base` で始まる行には、後続のフィールド番号をオフセットするためのベース値が含まれています。この機能により、関連するフィールドのセットをグループ化し、簡単に番号を付け直すことができます。
- * および # で始まらない行の形式は次のようになります。

  ```
  name    rel-numb    type
  ```

  各項目の説明は次のとおりです。

  - name は、フィールドの識別子です。`cpp` の制限を越えることはできません。
  - `rel-numb` は、フィールドの相対番号を示す数値です。この数値を現在の基数に加算すると、フィールドのフィールド番号を取得できます。
  - `type` は、フィールドの型を示します。指定できる型は、`char`、`string`、`short`、`long`、`float`、`double`、`carray`、`ptr`、`FML32`、または `VIEW32` のいずれかです。

エントリは空白類 (タブとスペースを任意に組み合わせたもの) で区切ります。

フィールド・テーブルからヘッダ・ファイルへの変換

既に説明したとおり、mkfldhdr (または mkfldhdr32) コマンドを実行すると、フィールド・テーブルが C コンパイラ処理に対応したファイルに変換されます。生成されたヘッダ・ファイルの各行の形式は次のとおりです。

```
#define  name  fldid
```

name はフィールドの名前、fldid はそのフィールド識別子です。このフィールド識別子は、前述したようにフィールド・タイプとフィールド番号から構成されています。フィールド番号は絶対数、つまり base に rel-number を足した数です。生成されたファイルは、C プログラムに組み込むことができます。

環境変数

フィールド・テーブルにアクセスする Fldid() などの関数と、それらを使用する mkfldhdr() および vuform() などのコマンドを使用する場合、メモリ内のフィールド・テーブルを作成するために、シェル変数 FLDTBLDIR と FIELDTBLS (FML32 では FLDTBLDIR32 と FIELDTBLS32) にそれぞれソース・ディレクトリとソース・ファイルを指定しておく必要があります。FIELDTBLS は、フィールド・テーブル・ファイル名の、カンマで区切られたリストを指定します。FIELDTBLS を指定しない場合は、フィールド・テーブル・ファイルの名前として fld.tbl が使用されます。FLDTBLDIR 環境変数は、コロンで区切られたディレクトリのリストで、この中から名前が絶対パス名でないフィールド・テーブルが検索されます (フィールド・テーブルの検索は、PATH 変数を使用する実行可能コマンドの検索とほぼ同じです)。FLDTBLDIR を定義しない場合、カレント・ディレクトリとみなされます。したがって、FIELDTBLS と FLDTBLDIR が設定されていない場合は、カレント・ディレクトリから取得した fld.tbl がデフォルトとなります。

フィールドをグループ (アプリケーションによってのみ使用されるデータベースのフィールドのグループなど) に分けるには、複数のフィールド・テーブルを使用すると便利です。ただし、フィールド名はフィールド・テーブル全体で一意になるようにします。これは、フィールド・テーブルが mkfldhdr コマンドによって C ヘッダ・ファイルに変換される可能性があり、同一のフィールド名があるとコンパイラ名の矛盾が生じるおそれがあるためです。また、Fldid 関数は、名前を FLDID にマップしますが、その際に複数のテーブルを検索します。最初に一致するものが見つかった時点で検索は終了します。

使用例

ベース値を 500 から 700 に変更した場合のフィールド・テーブルの例を以下に示します。

```
# employee ID fields are based at 500
*base 500

#name  rel-numb  type  comment
#----  --------  ----  -------
EMPNAM  1     string emp's name
EMPID   2     long  emp's id
EMPJOB  3     char  job type: D,M,F or T
SRVCDAY  4     carray service date

# address fields are based at 700

*base 700

EMPADDR  1     string street address
EMPCITY  2     string city
EMPSTATE 3     string state
EMPZIP  4     long  zip code
```

関連するヘッダ・ファイルは次のようになります。

```
#define EMPADDR  ((FLDID)41661) /* 番号 : 701 型 : string */
#define EMPCITY  ((FLDID)41662) /* 番号 : 702 型 : string */
#define EMPID   ((FLDID)8694)  /* 番号 : 502 型 : long */
#define EMPJOB  ((FLDID)16887) /* 番号 : 503 型 : char */
#define EMPNAM  ((FLDID)41461) /* 番号 : 501 型 : string */
#define EMPSTATE ((FLDID)41663) /* 番号 : 703 型 : string */
#define EMPZIP  ((FLDID)8896)  /* 番号 : 704 型 : long */
#define SRVCDAY  ((FLDID)49656) /* 番号 : 504 type:carray */
```

関連項目 mkfldhdr、mkfldhdr32(1)

『FML を使用した BEA Tuxedo アプリケーションのプログラミング』

GWADM(5)

名前 GWADM— ドメイン・ゲートウェイ管理サーバ

形式

```
GWADM SRVGRP = "identifier" SRVID = "number" REPLYQ = "N"
 CLOPT = "-A -- [-a {on | off}] [-t {on | off}]"
```

機能説明 ゲートウェイ管理サーバ (GWADM) は、BEA Tuxedo システムに組み込まれているサーバであり、Domains のゲートウェイ・グループ用の管理機能を提供します。

GWADM は、UBBCONFIG ファイルの SERVERS セクションで、特定のゲートウェイ・グループ内で動作するサーバとして定義する必要があります。すなわち、SRVGRP を、GROUPS セクションで指定した GRPNAME タグに設定する必要があります。SVRID も必須パラメータです。このパラメータの値を指定する際は、ゲートウェイ・グループ内で使用できるゲートウェイの最大数を考慮する必要があります。

GWADM のインスタンスは、Domains ゲートウェイ・グループごとに 1 つしか存在できず、そのインスタンスを、グループと関連付けられたゲートウェイに対して定義した MSSQ の一部にすることはできません。また、GWADM では REPLYQ 属性を N に設定する必要があります。

CLOPT オプションは、GWADM の起動時に渡されるコマンド行オプションの文字列です。このオプション文字列のフォーマットは次のとおりです。

```
CLOPT="-A -- gateway group runtime_parameters"
```

次のパラメータは、ゲートウェイ・グループの実行時パラメータとして認識されます。

-a {*on* | *off*}
: このオプションは、このローカル・ドメイン・アクセス・ポイントに対する動作記録ログ機能を off または on に切り替えます。デフォルトは off です。この設定は、ゲートウェイ・グループの実行中に、dmadmin プログラムを使用して変更できます (dmadmin(1) を参照)。

-t {*on* | *off*}
: このオプションは、このローカル・ドメインのドメイン・アクセス・ポイントに対する統計値収集機能を off または on に切り替えます。

デフォルトは off です。この設定は、ゲートウェイ・グループの実行中に、dmadmin プログラムを使用して変更できます (dmadmin(1) を参照)。

GWADM サーバは、対応するゲートウェイを起動する前に起動する必要があります。

移植性 GWADM は、サポートされているすべてのサーバ・プラットフォームで BEA Tuxedo システム提供のサーバとしてサポートされます。

相互運用性 GWADM は、BEA Tuxedo リリース 4.2.1 以降にインストールする必要があります。リリース 4.2.2 のゲートウェイが存在するドメイン内のほかのマシンの場合は、リリース 4.1 以降でも構いません。

使用例 次の例は、UBBCONFIG ファイルで管理サーバを定義する方法を示しています。この例では、GWTDOMAIN ゲートウェイ・プロセスを使用して別の BEA Tuxedo ドメインと通信します。BEA TOP END システムとの相互運用性を実現するには、GWTOPEND ゲートウェイ・プロセスを使用してください。GWTOPEND ゲートウェイ・プロセスの詳細と GWTOPEND の使用例については、GWTOPEND(5) を参照してください。

```
#
*GROUPS
DMADMGRP GRPNO=1
gwgrp  GRPNO=2
#
*SERVERS
DMADM SRVGRP="DMADMGRP" SRVID=1001 REPLYQ=N RESTART=Y GRACE=0
GWADM SRVGRP="gwgrp" SRVID=1002 REPLYQ=N RESTART=Y GRACE=0
  CLOPT="-A -- -a on -t on"
GWTDOMAIN SRVGRP="gwgrp" SRVID=1003 RQADDR="gwgrp" REPLYQ=N
RESTART=Y MIN=1 MAX=1
```

関連項目 dmadmin(1), tmboot(1), DMADM(5), DMCONFIG(5), DMCONFIG for GWTOPEND(5), GWTOPEND(5), servopts(5), UBBCONFIG(5)

『BEA Tuxedo アプリケーション実行時の管理』

『BEA Tuxedo アプリケーションの設定』

『BEA Tuxedo Domains コンポーネント』

GWTDOMAIN(5)

名前 GWTDOMAIN—TDomain ゲートウェイ・プロセス

形式 *GWTDOMAIN* SRVGRP = "*identifier*" SRVID = "*number*" RQADDR = "*queue_name*" REPLYQ = *value* RESTART = Y [MAXGEN = *value*] [GRACE = *value*]

機能説明 GWTDOMAIN は、ドメイン間の通信を実現するドメイン・ゲートウェイ・プロセスです。GWTDOMAIN プロセスは、リモート・ドメインにあるほかの GWTDOMAIN プロセスと通信します。

ドメイン・ゲートウェイは、UBBCONFIG ファイルおよび BDMCONFIG ファイルの SERVERS セクションに記述されます。ドメイン・ゲートウェイは、常に特定のゲートウェイ・グループと関連付ける必要があります。つまり、SRVGRP には、GROUPS セクションで指定された GRPNAME タグに対応する値を設定する必要があります。SVRID も必須パラメータです。このパラメータの値を指定する際は、ドメイン・グループ内で使用できるゲートウェイの最大数を考慮する必要があります。RESTART パラメータは Y に設定します。REPLYQ パラメータは Y または N に設定できます。

GWTDOMAIN プロセスは、GWADM(5) プロセスと同じグループ (先頭は GWADM) に指定する必要があります。1 つのドメインに対して複数の GWTDOMAIN プロセスを設定することもできまが、その場合は各プロセスを異なる BEA Tuxedo グループに設定する必要があります。

使用例 次の例は、UBBCONFIG ファイル内のドメイン・ゲートウェイ・グループの定義を示しています。

```
*GROUPS
DMADMGRP LMID=mach1 GRPNO=1
gwgrp  LMID=mach1 GRPNO=2
*SERVERS
DMADM SRVGRP="DMADMGRP" SRVID=1001 REPLYQ=N RESTART=Y MAXGEN=5
GRACE=3600
GWADM SRVGRP="gwgrp" SRVID=1002 REPLYQ=N RESTART=Y MAXGEN=5
GRACE=3600
GWTDOMAIN SRVGRP="gwgrp" SRVID=1003 RQADDR="gwgrp" REPLYQ=N
RESTART=Y MAXGEN=5 GRACE=3600
```

UBBCONFIG(5) および DMCONFIG(5) の使用例も参照してください。

関連項目 tmadmin(1), tmboot(1), DMADM(5), DMCONFIG(5), GWADM(5), servopts(5), UBBCONFIG(5)

『BEA Tuxedo Domains コンポーネント』

『BEA Tuxedo アプリケーションの設定』

『BEA Tuxedo アプリケーション実行時の管理』

GWTOPEND(5)

名前 GWTOPEND—TOP END ドメイン・ゲートウェイ・プロセス

形式 *GWTOPEND* SRVGRP = *"identifier"* SRVID = *"number"* RQADDR = *"queue_name"* REPLYQ = N RESTART = Y [MAXGEN = *value*] [GRACE = *value*]

機能説明 GWTOPEND は、BEA Tuxedo ドメインと BEA TOP END システム間の通信を提供するドメイン・ゲートウェイ・プロセスです。GWTOPEND ゲートウェイ・プロセスは、単一のシングル BEA TOP END システムの 1 つ以上のノード上のネットワーク・インターフェイス (NI) コンポーネントと通信します。異なる BEA Tuxedo グループで異なる GWTOPEND ゲートウェイを設定して、異なる BEA TOP END システムにアクセスしたり、負荷を分散することができます。GWTOPEND では、要求 / 応答、擬似会話、キュー操作、およびトランザクションがサポートされています。

ドメイン・ゲートウェイは、UBBCONFIG ファイルおよび BDMCONFIG ファイルの SERVERS セクションに記述されます。ドメイン・ゲートウェイは、特定のグループと関連付ける必要があります。つまり、SRVGRP には、GROUPS セクションで指定された GRPNAME タグに対応する値を設定する必要があります。

SVRID も必須パラメータです。このパラメータの値を指定する際は、ドメイン・グループ内で使用できるゲートウェイの最大数を指定する必要があります。RESTART パラメータは Y に設定します。REPLYQ パラメータは N に設定します。

GWTOPEND プロセスは、GWADM(5) プロセスと同じグループ (先頭は GWADM) に指定する必要があります。1 つのドメインに対して複数の GWTOPEND プロセスを設定することもできまが、その場合は各プロセスを異なる BEA Tuxedo グループに設定する必要があります。

BEA TOP END セキュリティをゲートウェイに対して設定する場合、BEA TOP END セキュリティ・サービス製品をノードにインストールし、TP_SYSTEM 名の srvtab ファイルを、下記の「ファイル」セクションに示すノードにコピーする必要があります。long 型のノード名がサポートされる場合は、下記の「ファイル」セクションに示すノードに nodemap ファイルをコピーする必要があります。

ファイル

`$TUXDIR/udataobj/nodemap`

`$APPDIR/srvtab.`*`system`* (*`system`* は BEA TOP END システム名)

`/usr/lib/libtp_krb.so` (BEA TOP END セキュリティがコンフィギュレーションされた UNIX プラットフォーム上にインストールされる)

`%TOPENDDIR%\bin\krb.dll` (BEA TOP END セキュリティがコンフィギュレーションされた Windows プラットフォーム上にインストールされる)

使用例

次の例は、`UBBCONFIG` ファイル内のドメイン・ゲートウェイ・グループの定義を示しています。

```
*GROUPS
DMADMGRP LMID=mach1 GRPNO=1
gwgrp  LMID=mach1 GRPNO=2
*SERVERS
DMADM SRVGRP="DMADMGRP" SRVID=1001 REPLYQ=N RESTART=Y MAXGEN=5
GRACE=3600
GWADM SRVGRP="gwgrp" SRVID=1002 REPLYQ=N RESTART=Y MAXGEN=5
GRACE=3600
GWTOPEND SRVGRP="gwgrp" SRVID=1003 RQADDR="gwgrp" REPLYQ=N
RESTART=Y MAXGEN=5 GRACE=3600
```

詳細については、`UBBCONFIG(5)` および `DMCONFIG for GWTOPEND(5)` の使用例も参照してください。

関連項目

`tmadmin(1)`, `tmboot(1)`, `DMADM(5)`, `DMCONFIG for GWTOPEND(5)`, `GWADM(5)`, `servopts(5)`, `UBBCONFIG(5)`

『BEA TOP END Programmer’s Reference Manual』: `ext_srvtab`(1T)、`nodemap`(5T)

『BEA Tuxedo アプリケーション実行時の管理』

『BEA Tuxedo アプリケーションの設定』

『BEA Tuxedo Domains コンポーネント』

『ATMI アプリケーションでの BEA Tuxedo TOP END Domain Gateway の使用』

GWTUX2TE、GWTE2TUX(5)

名前　GWTUX2TE、GWTE2TUX–BEA Tuxedo / BEA TOP END ゲートウェイ・サーバ

形式

```
GWTUX2TE SRVGRP = "identifier" SRVID = "number"
       CLOPT = "-- -f service_definition_file
       [-c TOPEND_remote_configuration_file]
       [-R sec] [-w wait_time] [[-u username] [-p password_file]]"

GWTE2TUX SRVGRP = "identifier" SRVID = "number"
       CLOPT = "-- -f service_definition_file
       [-c TOPEND_remote_configuration_file]
       [-R sec] [[-u username] [-g groupname]]"
```

機能説明　GWTUX2TE および GWTE2TUX はゲートウェイ・サーバです。GWTUX2TE は、BEA Tuxedo クライアントと BEA TOP END サーバとの接続を実現します。GWTE2TUX は、BEA TOP END クライアントと BEA Tuxedo サーバとの接続を実現します。ドメインに対して、これらのゲートウェイ・サーバの一方または両方をコンフィギュレーションできます。

GWTUX2TE および GWTE2TUX は、特定のサーバ・グループ内で動作するサーバとして、UBBCONFIG ファイルの SERVERS セクションで定義します。したがって、SRVGRP には GRPNAME パラメータ (GROUPS セクションで指定) の値に対応する値を設定する必要があります。SVRID パラメータも必須です。また、GWTUX2TE と GWTE2TUX には、ゲートウェイ・インスタンスの MIN 値と MAX 値を指定できます。これらのゲートウェイ・サーバは同期型ですが、複数のインスタンスを使用してスループットを向上させることも可能です。

CLOPT パラメータは、サーバの起動時にゲートウェイ・サーバにコマンド行オプションのセットを渡すパラメータです。CLOPT を使用してオプションを指定するには、次の形式を使用します。

```
CLOPT="-- gateway_group_run_time_parameters"
```

認識される CLOPT オプションは以下のとおりです。

`-f service_definition_file`
: このファイルには、ゲートウェイ・サーバによって宣言されるサービスと関数がリストされます (ファイル形式については、このリファレンス・ページの「コンフィギュレーション」を参照してください)。

-f を指定しない場合、または指定したファイルに無効な構文が含まれている場合、ゲートウェイ・サーバはエラー情報を記録して終了します。

-c *TOP_END_remote_configuration_file*

このファイルでは、ゲートウェイ・サーバと BEA TOP END システムの接続を定義します。このオプションを指定しない場合、デフォルトで $APPDIR/TOPENDRC.cfg がコンフィギュレーション・ファイルとなります。コンフィギュレーション・ファイルがない場合、または指定したファイルに無効な構文が含まれている場合、ゲートウェイ・サーバはエラー情報を記録して終了します。

-u *username* -p *password_file*

BEA TOP END システムでセキュリティ機能が有効な場合は、GWTUX2TE ゲートウェイに対して -u オプションおよび -p オプションを指定する必要があります。

-u を使用して指定したユーザのパスワードを含むファイルを -p オプションの後に指定します。パスワード・ファイルは ASCII 形式で作成し、パスワードは単一行で指定する必要があります。セキュリティを確保するには、ファイルの読み書きの権限を BEA Tuxedo の管理者に限定しておく必要があります。

詳細については、309 ページの「セキュリティ」を参照してください。

-R *Retry_interval*

ゲートウェイ・サーバが BEA TOP END システムへの接続を確立できない場合、または既存の接続が中断された場合、サーバは接続を 60 秒おき (デフォルト) に再試行します。-R を使用して、再接続の間隔 (秒単位) を変更することもできます。-R を 0 に設定すると、再接続は行われません。この場合、RESTART=Y を指定した状態で接続が失敗したり中断したりすると、ゲートウェイ・サーバは終了して再起動します。

BEA TOP END システムに接続できない場合、ゲートウェイ・サーバはそのシステムのサービスを提供できません。

-w *wait_time*

ゲートウェイ・サーバ GWTUX2TE は、BEA TOP END システムへの要求の送信後 30 秒間 (デフォルト) 応答を待ちます。-w パラメータを使

用すると、この待機時間を変更できます。待機時間 0 は、ゲートウェイ・サーバが応答を無限に待ち続けることを示します。

ゲートウェイ・サーバ GWTE2TUX の応答待機時間を設定するためのパラメータは用意されていないため、TUXCONFIG 内の通常のタイムアウト・パラメータを使用します。

`-u` *`username`* `-g` *`groupname`*

BEA Tuxedo サービスに対してアクセス制御リストを使用している場合は、ゲートウェイ・サーバ GWTE2TUX に対して -u オプションと -g オプションを指定する必要があります。デフォルトではゲスト特権が使用されます。

詳細については、309 ページの「セキュリティ」を参照してください。

プログラミング・パラダイム

ゲートウェイ・サーバ GWTUX2TE および GWTE2TUX では、要求 / 応答メッセージのみがサポートされています。要求の送受信を行う BEA Tuxedo クライアント API 呼び出しとしては以下がサポートされています。

- tpcall()
- tpacall() (TPNOREPLY フラグが設定される場合もある)
- tpgetrply()
- tpforward()

BEA TOP END サーバでは、APPL_CONTEXT フラグは設定できません。このフラグが設定されていると、ゲートウェイ・サーバは BEA TOP END ダイアログを解析し、BEA Tuxedo クライアントにエラー (TPESVCFAIL) を返します。

BEA TOP END クライアント API 呼び出しとしては以下がサポートされています。

- tp_client_send
- tp_client_receive

バッファ型

ゲートウェイ・サーバ GWTUX2TE および GWTE2TUX では、BEA Tuxedo CARRAY (X_OCTET) バッファのみがサポートされています。その他のバッファ・タイプを BEA Tuxedo アプリケーションから送信しようとすると、エラーが生成されてゲートウェイ・サーバによって記録されます。

コンフィギュレーション

ゲートウェイ・サーバ GWTUX2TE および GWTE2TUX は、BEA TOP END のリモート・クライアントとリモート・サーバのサービスを使用します。GWTUX2TE は BEA TOP END クライアントとして機能するため、リモート・クライアントのサービスを使用します。GWTE2TUX は BEA TOP END サーバとして機能するため、リモート・サーバのサービスを使用します。したがって、これらのゲートウェイ・プロセスを実行している BEA Tuxedo ノード上で、BEA TOPEND のリモート・クライアント / サーバ・コンフィギュレーション・ファイルを作成する必要があります。

BEA TOP END のリモート・クライアント / サーバ・コンフィギュレーション・ファイル

BEA TOP END のリモート・クライアント / サーバ・コンフィギュレーション・ファイルについては、『BEATOP END Remote Client Services Guide』で詳しく説明しています。ここでは、このファイルを簡単に説明します。

このコンフィギュレーション・ファイル内のエントリの形式は次のとおりです。

```
[top end configuration file]
[component type] remote server
[system] sysname
[primary node] machine_name        portnum
```

component type には、リモート・サーバ (remote server) を設定します。system には、BEA TOP END システムの名前を設定します。primary node には、BEA TOPEND Network Agent (NA) のマシン名とポート番号を設定します。

セカンダリ・ノードを指定することもできます。セカンダリ・ノードは、プライマリ・ノードへの接続が確立できない場合に使用します。BEA TOP END システムでは、セカンダリ・ノードを複数指定すると、接続のロード・バランシングがラウンド・ロビン方式で行われます。この機能により、ゲートウェイ・サーバの複数のインスタンスを BEA TOP END システムの異なるノードに対して接続できます。次に例を示します。

```
[secondary node] machine    28001
[secondary node] machine2   28001
```

ゲートウェイ・サーバ GWTUX2TE および GWTE2TUX では、オプションの target パラメータもサポートされています。

以下のパラメータは、ゲートウェイ・サーバ GWTUX2TE および GWTE2TUX ではサポートされません。コンフィギュレーション・ファイルでは、これらのパラメータを指定しないようにしてください。

- shutdown
- codeset
- maxconctx

1 つのゲートウェイ・プロセスから接続できるシステムは、TOPENDRC.cfg ファイルの [system] に指定した 1 つの BEA TOP END システムのみです。2 つ目のゲートウェイ・プロセスを使用して、ほかの BEA TOP END システムに接続できるよう設定することも可能です。2 つ目のコンフィギュレーション・ファイルを指すようにするには、CLOPT -c パラメータを使用します。

サービス定義ファイル

サービス定義ファイルの構文は次のとおりです。

```
*TE_LOCAL_SERVICES # TOP END クライアントからアクセス可能な BEA Tuxedo サービス
Servicename PRODUCT=product_name FUNCTION=function_name
    QUALIFIER=function_qualifier

*TE_REMOTE_SERVICES # BEA Tuxedo クライアントからアクセス可能な TOP END サービス
Servicename PRODUCT=product_name FUNCTION=function_name
    QUALIFIER=function_qualifier TARGET=target_name
```

Servicename は、インポート (TE_REMOTE_SERVICE) またはエクスポート (TE_LOCAL_SERVICE) される BEA Tuxedo サービスを示します。

PRODUCT パラメータは必須パラメータで、FUNCTION、QUALIFIER、および TARGET パラメータはオプション・パラメータです。また、TARGET パラメータは、TE_REMOTE_SERVICES に対してのみ有効です。

次の構文を使用すると、サービス定義ファイルのパラメータをデフォルトとして定義できます。

DEFAULT: PRODUCT=*product_name*

TE_LOCAL_SERVICES セクション内のすべてのサービスには、同じ PRODUCT 名を指定する必要があります。

FUNCTION パラメータを指定しない場合、関数名はサービス名とみなされます。サービス・エントリに対して QUALIFIER パラメータおよび TARGET パラメータを指定しない場合、そのサービスに対して関数修飾子およびターゲット名は使用されません。

BEA Tuxedo サービスの有効な名前については、『BEA Tuxedo アプリケーションの設定』を参照してください。PRODUCT パラメータ、FUNCTION パラメータ、QUALIFIER パラメータ、および TARGET パラメータに設定できる有効値については、『BEA TOP ENDAdministrator's Guide』を参照してください。

制限事項 以下の機能は、ゲートウェイではサポートされません。

- トランザクション
- 会話
- イベント
- 任意通知型メッセージ
- キュー (/Q、RTQ)
- 暗号化
- データの圧縮
- 30K 以上のメッセージの送受信
- 移行
- フォーマット処理
- MCC および LMA

セキュリティ 次の表は、さまざまなコンフィギュレーションにおけるセキュリティの設定を示しています。

表 37 ゲートウェイ・サーバのセキュリティ

サーバの種類	セキュリティの状態	セキュリティの設定方法
GWTUX2TE	BEA TOP END システムに認証機能が設定されている。	-u オプションを使用してユーザ名を設定する。 -p オプションを使用してパスワードを設定する。 オペレーティング・システムの保護機能を使用してこのファイルを保護する。
GWTE2TUX	BEA Tuxedo システムに SECURITY=APP_PW、USER_AUTH が設定されている。	不要
GWTE2TUX	BEA Tuxedo システムに SECURITY=ACL、MANDATORY ACL が設定されている。	-u オプションを使用してユーザ名を設定し、-g オプションを使用してグループ名を設定する。

また、CLOPT を使用して指定された username と groupname の組み合わせまたは username と password の組み合わせを、対応する BEA Tuxedo または BEA TOP END のセキュリティ・データベースに入力する必要もあります。BEA Tuxedo のセキュリティ・データベースでは、ユーザ名は tpusradd() を使用して作成するのが一般的です。グループ名は、tpgrpadd() を使用して作成するのが一般的です。

移植性 ゲートウェイ・サーバ GWTUX2TE および GWTE2TUX は、Windows、Sun Solaris、HP-UX、IBM AIX、および NCR MP-RAS でサポートされます。

相互運用性 ゲートウェイ・サーバ GWTUX2TE および GWTE2TUX を実行するには、BEA Tuxedo リリース 6.5 以降が必要です。これらのゲートウェイ・サーバは、BEA TOP END 2.05 以上と相互運用できます。

使用例 次の例では、BEA Tuxedo の UBBCONFIG ファイルおよび BEA TOP END のサービス定義ファイルにおけるゲートウェイ・サーバの定義方法を示します。

この例では、BEA Tuxedo クライアントが RSERVICE サービスに対して tpcall() を発行しています。要求は GWTUX2TE ゲートウェイを介して BEA TOP END システム (pluto) に転送され、BEA TOP END サービス (RPRODUCT:RFUNC) が呼び出されます。

同様に、BEA TOP END クライアントが tp_client_send を発行し、PRODUCT として LPRODUCT を指定し、FUNCTION として LFUNC を指定しています。要求は、GWTE2TUX ゲートウェイを介して BEA Tuxedo システムに転送され、BEA Tuxedo サービス (LSERVICE) が呼び出されます。

コード リスト 37-1　BEA Tuxedo **の** UBBCONFIG **ファイル**

```
###############
#UBBCONFIG
*GROUPS
TOPENDGRP  GRPNO=1

#
*SERVERS
GWTE2TUX SRVGRP="TOPENDGRP" SRVID=1001 RESTART=Y MAXGEN=3 GRACE=10
       CLOPT="-- -f servicedefs -R 30"
GWTUX2TE SRVGRP="TOPENDGRP" SRVID=1002 RESTART=Y MAXGEN=3 GRACE=10
       MIN=5 MAX=5
       CLOPT="-- -f servicedefs"
```

コード リスト 37-2　BEA TOP END **のサービス定義ファイル**

```
############
# サービス定義ファイル
*TE_LOCAL_SERVICES
DEFAULT: PRODUCT=LPRODUCT
LSERVICE FUNCTION=LFUNC

*TE_REMOTE_SERVICES
RSERVICE PRODUCT=RPRODUCT FUNCTION=RFUNC
```

コード リスト 37-3　BEA TOP END **のリモート・コンフィギュレーション・**

ファイル

```
# TOP END リモート・コンフィギュレーション・ファイル
[top end configuration file]
[component type] remote server
[system] pluto
[primary node] topendmach       28001
```

注記　`primary node` エントリの *`port`* 値 (BEA TOP END のリモート・コンフィギュレーション・ファイル の `28001`) は、BEA TOP END Network Agent のポート番号と一致している必要があります。

ソフトウェア要件

次のソフトウェア・コンポーネントが必要です。

- BEA Tuxedo リリース 6.5
- BEA TOP END 2.05

エラー

次のいずれかの状況が発生すると、BEA Tuxedo クライアントに `TPESVCFAIL` が送信されます。

- BEA TOP END サービスにアクセスできない。
- TOP END サービスからエラーが返された。
- BEA TOP END システムへのネットワーク・リンクが確立できない。
- BEA Tuxedo クライアントから `CARRAY` および `X_OCTET` 以外のバッファ・タイプが送信された。

次のいずれかの状況が発生すると、詳細なステータス `TP_EXT_SERVER_APPL` とともにエラー・メッセージ `TP_RESET` が BEA TOP END クライアントに送信されます。

- BEA Tuxedo サービスにアクセスできない (サービスが中断されているなどの理由による)。
- BEA Tuxedo サービスがタイムアウトになった。
- BEA Tuxedo サービスから `TPFAIL` または `TPEXIT` が返された。

上記の例のように、ゲートウェイが対応するシステムでは利用できないサービスを提供している場合、クライアントにエラー (TPESVCFAIL) が送信されます。これは、ローカル・サービスの呼び出し後に返されるエラーとは異なります。ローカル・サービスを呼び出した場合は、クライアントに TPENOENT (BEA Tuxedo システムの場合) または TP_SERVICE (BEA TOP END システムの場合) が送信されます。

関連項目 tmboot(1), servopts(5), UBBCONFIG(5)

『BEA Tuxedo アプリケーションの設定』

『BEA Tuxedo アプリケーション実行時の管理』

『BEA TOP END Remote Client/Server Services Guide』

langinfo(5)

名前 langinfo― 言語情報定数

形式

```
#include <langinfo.h>
```

機能説明 このヘッダ・ファイルには、langinfo データの項目の識別に使用する定数が格納されています。各項目のモードは nl_types(5) で定義されています。

DAY_1
: 週の第 1 日目 (例 : "sunday")

DAY_2
: 週の第 2 日目 (例 : "monday")

DAY_3
: 週の第 3 日目 (例 : "tuesday")

DAY_4
: 週の第 4 日目 (例 : "wednesday")

DAY_5
: 週の第 5 日目 (例 : "thursday")

DAY_6
: 週の第 6 日目 (例 : "friday")

DAY_7
: 週の第 7 日目 (例 : "saturday")

ABDAY_1
: 週の第 1 日目の略称 (例 : "sun")

ABDAY_2
: 週の第 2 日目の略称 (例 : "mon")

ABDAY_3
: 週の第 3 日目の略称 (例 : "tue")

ABDAY_4
　　週の第 4 日目の略称 (例 : "wed")

ABDAY_5
　　週の第 5 日目の略称 (例 : "thur")

ABDAY_6
　　週の第 6 日目の略称 (例 : "fri")

ABDAY_7
　　週の第 7 日目の略称 (例 : "sat")

MON_1
　　年の 1 番目の月 (例 : "january")

MON_2
　　年の 2 番目の月 (例 : "february")

MON_3
　　年の 3 番目の月 (例 : "march")

MON_4
　　年の 4 番目の月 (例 : "april")

MON_5
　　年の 5 番目の月 (例 : "may")

MON_6
　　年の 6 番目の月 (例 : "june")

MON_7
　　年の 7 番目の月 (例 : "july")

MON_8
　　年の 8 番目の月 (例 : "august")

MON_9
　　年の 9 番目の月 (例 : "september")

MON_10
　　年の 10 番目の月 (例 : "october")

MON_11
　年の 11 番目の月 (例 : "november")

MON_12
　年の 12 番目の月 (例 : "december")

ABMON_1
　年の 1 番目の月の略称 (例 : "jan")

ABMON_2
　年の 2 番目の月の略称 (例 : "feb")

ABMON_3
　年の 3 番目の月の略称 (例 : "mar")

ABMON_4
　年の 4 番目の月の略称 (例 : "apr")

ABMON_5
　年の 5 番目の月の略称 (例 : "may")

ABMON_6
　年の 6 番目の月の略称 (例 : "jun")

ABMON_7
　年の 7 番目の月の略称 (例 : "jul")

ABMON_8
　年の 8 番目の月の略称 (例 : "aug")

ABMON_9
　年の 9 番目の月の略称 (例 : "sep")

ABMON_10
　年の 10 番目の月の略称 (例 : "oct")

ABMON_11
　年の 11 番目の月の略称 (例 : "nov")

ABMON_12
　年の 12 番目の月の略称 (例 : "dec")

RADIXCHAR
: 基数文字 (例 : ".")

THOUSEP
: 1000 位の区切り文字 (例 : ",")

YESSTR
: 肯定応答文字 (例 : "yes")

NOSTR
: 否定応答文字 (例 : "no")

CRNCYSTR
: 通貨記号

D_T_FMT
: 日付・時刻の省略時形式

D_FMT
: 日付の省略時形式

T_FMT
: 時刻の省略時形式

AM_STR
: 午前を表す略語 (例 : "AM")

PM_STR
: 午後を表す略語 (例 : "PM")

この情報は、nl_langinfo(3c) を使用して検索します。

項目は、LANGINFO という特別なメッセージ・カタログから検索します。このカタログは、各ロケールごとに生成され、適切なディレクトリにインストールしておきます (mklanginfo(1) を参照)。

関連項目 mklanginfo(1), nl_langinfo(3c), strftime(3c), nl_types(5)

LAUTHSVR

名前 LAUTHSVR—WebLogic Server 組み込み LDAP ベース認証サーバ

形式 `LAUTHSVR SRVGRP="identifier" SRVID=number other_parms CLOPT="-A -- -f filename"`

機能説明 LAUTHSVR は System/T サーバで、ユーザ・セキュリティ情報が WebLogic Server に保持されている場合でも認証サービスを提供します。このサーバを保護されたアプリケーションで使用することにより、クライアントがアプリケーションに参加するときにユーザ単位の認証を行うことができます。このサーバは、パスワードとして TPINIT 型付きバッファを含むサービス要求を受け付け、これを WebLogic Server に格納されたコンフィギュレーション済みパスワードによって検証します。要求が妥当であると認められると、クライアントが使用するためのチケットとしてアプリケーション・キーが返されます。

- ユーザが WebLogic Server の管理者グループに属している場合は、アプリケーション・キーとして TPSYSADM が返されます。
- ユーザが WebLogic Server のオペレータ・グループに属している場合は、アプリケーション・キーとして TPSYSOP が返されます。

注記 tpsysadm と tpsysop に対応するアプリケーション・キーは、それぞれ 0x80000000 と 0xC0000000 でなければなりません。

デフォルトでは、$TUXDIR/udataobj/tpldap ファイルを使用して、LDAP のコンフィギュレーション情報を取得します。このファイルは、ファイル名を指定することでオーバライドできます。ファイル名は、サーバのコマンド行オプションで `"-f filename"` オプションを使用して指定します。たとえば、`CLOPT="-A -- -f/usr/tuxedo/myapp/myldap"` のように指定します。このコンフィギュレーション・ファイルを、マスタ・マシンから Tuxedo UBBCONFIG ファイル内のほかのマシンに自動的に伝達することはできません。複数の LAUTHSVR を使用するには、複数のマシンを別々にコンフィギュレーションする必要があります。

LAUTHSVR の詳細については、321 ページの「LAUTHSVR に関する追加情報」を参照してください。

SECURITY USER_AUTH

SECURITY が USER_AUTH 以上に設定されている場合は、強制的にユーザ単位での認証が実行されます。認証サービスの名前は、アプリケーションに対してコンフィギュレーションできます。これを指定しない場合は、LAUTHSVR 用に宣言されたデフォルト・サービスである AUTHSVC がデフォルトで設定されます。

認証要求は、LDAP データベース内で最初に一致するユーザ名に対してのみ認証されます。複数のエントリに対する認証はサポートされていません。

SECURITY ACL or MANDATORY_ACL

SECURITY が ACL または MANDATORY_ACL に設定されている場合、ユーザ単位の認証が強制的に実行され、サービスや、アプリケーションのキュー、イベントにアクセスするためのアクセス管理リストがサポートされます。認証サービスの名前は AUTHSVC (これらのセキュリティ・レベル用に LAUTHSVR によって宣言されたデフォルト・サービス) である必要があます。

AUTHSVR によって返されるアプリケーション・キーは、下位 17 ビット内のユーザ識別子です。グループ識別子はその次の 14 ビットです。上位ビットは管理キー用に予約されています。

LAUTHSVR に関する追加情報

移植性 LAUTHSVR は、Tuxedo System/T に付属のサービスとして非 Workstation プラットフォームでサポートされます。

使用例

```
# LAUTHSVR の使用例
*RESOURCES
AUTHSVC   "..AUTHSVC"
SECURITY  ACL

*SERVERS
LAUTHSVR SRVGRP="AUTH" SRVID=100
CLOPT="-A -- -f /usr/tuxedo/udataobj/tpldap"
```

MIB(5)

名前 MIB― 管理情報ベース

```
#include <fml32.h>
#include <fml1632.h> /* オプション */
#include <tpadm.h>
#include <cmib.h>  /* コンポーネント MIB ヘッダ */
```

機能説明 BEA Tuxedo システムのアプリケーションは、いくつかの異なるコンポーネント (BEA Tuxedo、Workstation など) で構成され、それぞれのコンポーネントはそのコンポーネント専用に定義された管理情報ベース (MIB) を利用して管理されます。これらのコンポーネントの MIB は、それぞれシステムの特定の部分に対応した MIB 関連のリファレンス・ページで定義されています。たとえば、TM_MIB(5) のリファレンス・ページでは、BEA Tuxedo アプリケーションの基本的な側面の管理に使用する MIB について定義しています。

ただし、これらのコンポーネントの MIB は、必要なアクセスを提供するための関連インターフェイスについて十分に定義したものではありません。この MIB(5) リファレンス・ページでは、管理者、オペレータ、あるいはユーザが、定義済みコンポーネント MIB と相互作用するための汎用的なインターフェイスを記述しています。BEA Tuxedo システムの MIB に対する汎用インターフェイスは、2 つの主要部分から構成されます。

その 1 つでは、BEA Tuxedo システムの既存のインターフェイスが、コンポーネント MIB をサポートする管理サービスへのアクセスを提供する際にどのように使用されるかを記述しています。BEA Tuxedo システムのバッファ・タイプの 1 つである FML32 は、コンポーネント MIB に入力データを渡したり、コンポーネント MIB から出力データを受け取ったりするために使用します。ATMI 要求 / 応答関数は、システム提供のサービスとして組み込まれており、コンポーネント MIB に対するインターフェイスとして使用します。FML32 バッファの ATMI 関数を使用した管理ユーザとコンポーネント MIB との相互作用については、このリファレンス・ページの「「FML32」」および「「ATMI」」で詳しく説明します。

汎用インターフェイスのもう 1 つの部分では、すべてのコンポーネント MIB との相互作用に使用する FML32 の追加の入出力フィールドについて記述しています。FML32 の追加のフィールドを使用すると、要求の機能を拡張したり (操作コードの指定など)、新たな応答属性 (エラー・コード、説明文など) を使用したりできます。FML32 の追加フィールドについては、このマニュアルの「「入力」」および「「出力」」で詳しく説明します。

「「使用方法」」では、管理を目的としたコンポーネント MIB との相互作用に使用できる既存の ATMI 関数や追加の FML32 フィールドの使用例を示します。

また、このリファレンス・ページでは、アプリケーションを管理する際のユーザとコンポーネント MIB とのインターフェイスを定義するのに加え、コンポーネント MIB のリファレンス・ページでクラスの定義に使用する形式を制定しています (「「クラスの説明」」を参照)。

このリファレンス・ページでは、`T_CLASS` および `T_CLASSATT` という 2 つの汎用クラスを定義しています。これら 2 つのクラスは、管理クラスの識別や、クラスまたは属性のパーミッションの調節に使用します。`MIB(5)` のすべてのクラス定義の追加情報については、352 ページの「MIB(5) に関する追加情報」を参照してください。「「診断」」のセクションでは、コンポーネント MIB のシステム・サービスが返す可能性のあるエラー・コードのリストを示します。

認証

ユーザがアプリケーションに結合しようとすると、その権限があるかどうかの認証が行われます (`tpinit(3c)` を参照)。管理者およびオペレータは、`tpinit()` の実行時に `tpsysadm` または `tpsysop` というクライアント名のアプリケーションへの結合を要求できます。2 つの `cltname` 値は予約されており、これらに関連付けることができるのはアプリケーションの管理者およびオペレータのみです。

アプリケーションを最初にコンフィギュレーションする管理者が、特定のセキュリティ・タイプを選択することでセキュリティのレベルを決定します。選択できるセキュリティ・タイプは以下のとおりです。

- セキュリティなし
- アプリケーション・パスワードによる認証
- アプリケーション・パスワードとアプリケーション固有の認証サービスによる認証

セキュリティ・タイプを選択することで、管理者やオペレータが AdminAPI を介してコンポーネント MIB にアクセスする際の柔軟性とセキュリティが決まります。

最も確実で柔軟なセキュリティ・タイプは、アプリケーション・パスワードとアプリケーション固有の認証サーバによる認証 (AUTHSVR(5) を参照) です。この方法では、任意のユーザまたは指定されているユーザが適切なパスワードを認証サーバに提供すると、そのユーザによるアクセスが許可されます。

アプリケーション固有の認証サーバが存在しない場合、クライアントはアプリケーションの認証要求 (「セキュリティなし」または「アプリケーション・パスワードによる認証」のどちらか) を満たし、TPINIT 構造体の cltname フィールドに特別なクライアント名の 1 つを指定した上で、ローカルの UNIX システムの BEA Tuxedo 管理者として実行することで管理者またはオペレータの特別なパーミッションを取得する必要があります。いずれの場合も、正常に結合されたクライアントにはシステムによってキーが割り当てられます。このキーは、クライアントが行うすべての要求に対して与えられます。tpsysadm または tpsysop として正しく認証されたクライアントには、特別な権限を持っていることを示す認証キーが割り当てられます。

管理者用認証を指定した場合、API にアクセスする前にシステムに結合するクライアントに対してのみ適用されます。API を使用するサーバは、このサーバがサービスするクライアントと同様に扱われます。tpsvrinit() または tpsvrdone() から発行されるサービス要求は、管理者からの要求として処理されます。

FML32

BEA Tuxedo システムが定義したコンポーネント MIB を使用するアプリケーション管理は、FML32 バッファ・タイプでのみサポートされています。MIB 情報にアクセスするアプリケーション・プログラムは、FML32 型付きバッファの割り当て、処理、更新を行うように記述する必要があります。ここでは、FML32 を使用する 2 通りの方法について簡単に説明します。詳細については Fintro() を参照してください。

FML32 にインターフェイスする最も直接的な方法は、標準の <fml.h> ヘッダ・ファイルではなく <fml32.h> ヘッダ・ファイルをインクルードし、『BEA Tuxedo FML リファレンス』で指定されている関連の各 FML インターフェイスの FML32 バージョンを使用する方法です。たとえば、Fchg() の代わりに Fchg32() を使用します。

FML32 にインターフェイスするもう 1 つの方法は、<fml32.h> と <fml1632.h> の両方のヘッダ・ファイルをインクルードする方法です。この 2 つのヘッダ・ファイルを組み合わせて使用することで、ベースの FML インターフェイス (たとえば Fchg()) 用にプログラミングしても、実際には各インターフェイスの FML32 バージョンを呼び出すことができます。

ATMI　アプリケーション・プログラムでコンポーネント MIB 固有の属性情報にアクセスしたり更新したりするには、FML32 型付きバッファを割り当て、要求されたデータをそのバッファに格納した上でサービス要求を送出し、サービス要求に対する応答を受け取って結果に関する情報を取り出します。FML32 型付きバッファでの情報の格納および抽出には、前述した FML32 インターフェイスを使用します。バッファの割り当て、要求の送出、および応答の受信は、下記の汎用 ATMI ルーチンを使用して、そのガイドラインと制約の範囲内で行われます。すべてのコンポーネントに対する MIB 要求は、コアの BEA Tuxedo コンポーネント MIB サービスである ".TMIB" に送出する必要があります。このサービスは、TM_MIB(5) 要求を処理するエージェントとしての役割を果たすだけでなく、他のコンポーネント MIB に対する要求を転送します。これにより、ユーザ側でサービス名を MIB やクラスとマッチングする必要がなくなります。

tpalloc()
: BEA Tuxedo システム MIB サービスへの要求の送出や応答の受信に使用する FML32 型付きバッファを割り当てます。FML32 バッファ・タイプにサブタイプはありません。デフォルトの最小サイズは 1024 バイトです。

tprealloc()
: FML32 型付きバッファの再割り当てを行います。

tpcall()
: データが格納された FML32 型付きバッファを入力とし、このサービスが返す出力を格納するバッファとして割り当て済みの FML32 型付きバッファを指定して、BEA Tuxedo システム MIB サービスである ".TMIB" を呼び出します。FML32 は自己記述型バッファ・タイプであるため、入力バッファのバッファ・サイズに 0 を指定することができます。この呼び出しがトランザクション内で発行される場合は TPNOTRAN フラグを使用する必要があります。トランザクション内でない場合、この関数に対して定義されたフラグの使用についての条件や制約は一切ありません。

`tpacall()`
: データが格納された FML32 型付きバッファを入力として、BEA Tuxedo システム MIB サービスである ".TMIB" を非同期で呼び出します。FML32 は自己記述型バッファ・タイプであるため、入力バッファのバッファ・サイズに 0 を指定することができます。この呼び出しがトランザクション内で発行される場合は TPNOTRAN フラグを使用する必要があります。トランザクション内でない場合、この関数に対して定義されたフラグの使用についての条件や制約は一切ありません。

`tpgetrply()`
: これ以前に生成された BEA Tuxedo システム MIB サービスである ".TMIB" の非同期呼び出しに対する応答を受信します。応答は、すでに割り当てられている FML32 型付きバッファに格納されます。この関数に対して定義されたフラグの使用についての条件や制約は一切ありません。

`tpenqueue()`
: BEA Tuxedo システム MIB サービスである ".TMIB" に対する要求を、後で処理するためにキューに登録します。FML32 は自己記述型バッファ・タイプであるため、入力バッファのバッファ・サイズに 0 を指定することができます。この関数に対して定義されたフラグの使用についての条件や制約は一切ありません。ただし、アプリケーションによってこのような要求の転送を処理するようにコンフィギュレーションされた TMQFORWARD(5) サーバを起動する際は、-n (TPNOTRAN フラグが設定された tpcall()) と -d (削除) オプションを指定する必要があります。

`tpdequeue()`
: BEA Tuxedo システム MIB サービスである ".TMIB" に対してこれ以前にキューに登録された要求への応答をキューから取り出します。応答は、すでに割り当てられている FML32 型付きバッファに格納されます。この関数に対して定義されたフラグの使用についての条件や制約は一切ありません。

入力 BEA Tuxedo システム MIB に対する管理要求の特徴付けや制御には、それぞれ特定の FML32 フィールドを使用します。これらのフィールドは、ヘッダ・ファイル <tpadm.h> だけでなく、このリファレンス・ページでも定義されています。対応するフィールド・テーブルは、${TUXDIR}/udataobj/tpadm にあります。これらのフィールドは、管理サー

ビス要求を行う前に必要なコンポーネント MIB 固有のフィールドのほかに、FML32 要求バッファにも追加されます。以下ではこれらのフィールドについて説明し、最後に各フィールドが必須、任意、または未使用となる操作をまとめて表に示します。

TA_OPERATION
: 実行する操作を示す文字列値フィールド。有効な操作は GET、GETNEXT 、および SET です。

TA_CLASS
: アクセスするクラスを示す文字列値フィールド。クラス名はコンポーネント MIB 固有のリファレンス・ページで定義されています。

TA_CURSOR
: 直前の GET または GETNEXT 操作時にシステムが返した文字列値の FML32 フィールド。アプリケーションでは、返された値を以後の要求に転送する必要があります。これにより、システムが現在の検索位置を判別することが可能になります。

TA_OCCURS
: GET または GETNEXT 操作時に検索されたオブジェクトの数を示す long 値の FML32 フィールド。このフィールドを指定しない場合、スペースがあるかぎり、一致するすべてのオブジェクトが返されます。

TA_FLAGS
: 汎用のフラグ値およびコンポーネント MIB 固有のフラグ値を示す long 値の FML32 フィールド。この属性に設定できるコンポーネント MIB 固有の値は、各コンポーネント MIB のリファレンス・ページに定義されています。以下に、汎用のフラグ値と使用方法を示します。

  MIB_LOCAL
  : このフラグは、この MIB に定義されている特定のクラスからの検索方法を変更する場合に使用します。この MIB 内のいくつかのクラスには、グローバル情報 (アクティブ・アプリケーションの任意のサイトで入手可能) とローカル情報 (オブジェクトがアクティブな特定のアプリケーションで入手可能) の両方があります。これらのクラスから情報を検索する要求では、検索を効率的に行うため、デフォルトではローカル情報ではなくグローバル情報のみを検索します。アプリケーション・ユーザが複数のサイトからローカル情報を収集する必要がある場合は、

検索要求時にこのフラグをセットする必要があります。ローカル情報のあるクラスの場合、属性表の最後にローカル属性がリストされています。ローカル属性かどうかは副見出しに示されています。ローカル情報のみのクラスでは、このフラグ値がセットされていなくてもローカル情報が検索されます。

MIB_PREIMAGE
: SET 操作が実行される前にプレイメージ・チェックにパスする必要があることを示します。プレイメージ・チェックでは、MIB 固有のクラス属性のオカレンス 0 が既存のオブジェクトと一致することを確認します。一致した場合、そのオブジェクトは MIB 固有のクラス属性のオカレンス 1 で更新されます。2 回以上発生しない属性は、プレイメージ・チェックの対象にはなりません。複数回出現するフィールドは、その対応するカウント属性が 2 度指定されている場合にチェックされます。

MIB_SELF
: このフラグは、要求元のクライアントやサーバの識別属性を処理前に要求バッファに追加する必要があることを示します。クライアントの場合は TA_CLIENTID を追加し、サーバの場合は TA_GRPNO と TA_SRVID を追加します。

TA_FILTER
: 返す必要のある特定のクラス属性を、最大 32 のオカレンスで指定できる long 値の FML32 フィールド。値 0 のオカレンスを指定してリストを終了することができますが、指定しなくてもかまいません。属性の初期値が 0 のリストでは、クラス固有の属性ではなく、一致したクラス・オブジェクトの個数が返されます。

TA_MIBTIMEOUT
: 要求を満たすために必要なコンポーネント MIB サービス内の時間 (秒数) を示す long 値の FML32 フィールド。0 以下の値を指定した場合、コンポーネント MIB サービスはブロッキング処理を実行できません。この値を指定しない場合、デフォルトで 20 に設定されます。

TA_CURSORHOLD
: 現在の GET または GETNEXT の要求が満たされた後、最初の GET 操作で生成されたシステム・スナップショットを処分せずに保持しておく時間 (秒数) を示す long 値の FML32 フィールド。0 以下の値を指定した

場合、現在の要求が満たされるとスナップショットが処分されます。この値を指定しない場合、デフォルトで 120 に設定されます。

次の表では、R は必須の INPUT 属性、O はオプションの INPUT 属性、— は使用されない INPUT 属性を示します。

表 38 入力表

属性	タイプ	GET	GETNEXT	SET
TA_OPERATION	string	R	R	R
TA_CLASS	string	R	—	R
TA_CURSOR	string	—	R	—
TA_OCCURS	long	O	O	—
TA_FLAGS	long	O	O	O
TA_FILTER	long	O	—	—
TA_MIBTIMEOUT	long	O	O	O
TA_CURSORHOLD	long	O	O	—

出力 正常終了した管理要求からの出力は、1 つまたは複数の MIB 固有オブジェクトと汎用出力フィールドの 1 つのオカレンスからなります。通常、複数の MIB 固有オブジェクトは、返された各クラス属性の複数のオカレンスによって出力に反映されます。各属性のオカレンス 0 は 1 番目のオブジェクトに、オカレンス 1 は 2 番目のオブジェクトに関連します (オカレンス 2 以降も同様)。このガイドラインの例外は、コンポーネント MIB のリファレンス・ページに記載されています。特定の属性値が設定されていない中間オカレンスでは、プレース・ホルダとして FML32 定義の NULL フィールド値が挿入されます。SET 操作が正常終了すると、操作実行後のオブジェクトを反映した単一のオブジェクトが返されます。GET 操作または GETNEXT 操作が正常終了すると、要求されたオカレンス数 (後述の TA_OCCURS を参照) や MIB 固有システム・サービス内の指定されたキー・フィールドおよびスペース制限と一致したオカレンス数に応じて、0 またはそれ以上のオカレンスが返されます。

重要な点は、任意のクラスに対して定義されたすべての属性が、どの要求に対しても返されるわけではないことです。属性が返されるかどうかは、オブジェクトの状態、相互運用のリリース環境、入力要求フィルタによって決まります。管理プログラマは、属性値が出力バッファ内に存在することを前提にするのではなく、属性値が存在するかどうかを明示的に確認する必要があります。

繰り返しになりますが、正常に処理された管理要求には、すべての MIB に適用する汎用のフィールドが含まれています。これらのフィールドは、ヘッダ・ファイル <tpadm.h> に定義されています。対応するフィールド・テーブルは、${TUXDIR}/udataobj/tpadm にあります。汎用応答フィールドは応答バッファに追加され、コンポーネント MIB 固有フィールドで返されます。以下では、各汎用応答フィールドについて説明します。

TA_CLASS
: 応答バッファに表されたクラスを示す文字列値のフィールド。クラス名はコンポーネント MIB 固有のリファレンス・ページで定義されています。

TA_OCCURS
: 応答バッファ内のオブジェクトの数を示す long 値の FML32 フィールド。

TA_MORE
: 要求キー・フィールドが一致する追加オブジェクトが、システム・スナップショットにいくつ保持されているかを示す long 値の FML32 フィールド。このフィールドは SET 操作では返されません。

TA_CURSOR
: システムが保持するスナップ・ショット内での位置を示す文字列値の FML32 フィールド。後続の GETNEXT 操作では、このフィールドを要求バッファに追加する必要があります。このフィールドの値は、アプリケーション・ユーザが解釈したり変更したりすることはできません。このフィールドは SET 操作では返されません。

TA_ERROR
: 正常な終了を示す負でない戻りコードを格納する long 値の FML32 フィールド。以下に、汎用のリターン・コードとその意味を示します。

  TAOK
  : 操作が正常に実行されました。アプリケーションに対する更新は行われていません。

  TAUPDATED
  : アプリケーションに対する更新が正常に終了しました。

TAPARTIAL
: アプリケーションに対する部分的な更新が正常に終了しました。

MIB 固有のシステム・サービス処理において管理要求が失敗すると、アプリケーションに対してアプリケーション・サービス・エラーが返されます。このエラーには、元々の要求とエラーの特徴を示す汎用フィールドが含まれます。アプリケーション・サービスの失敗は、tpcall() または tpgetrply() から返される TPESVCFAIL エラーによって示されます。TMQFORWARD(5) サーバを介して返されたアプリケーション・サービス・エラーは、元の要求で指定されたエラー・キューに登録されます (サーバのコマンド行で -d オプションを指定した場合)。以下に、失敗した管理要求の特徴を示す汎用フィールドを示します。

TA_ERROR
: 発生した特定のエラーを示す long 値の FML32 フィールド。汎用のエラー・コードの場合は、このリファレンス・ページの "DIAGNOSTICS" セクションに記述されます。コンポーネント MIB 固有のエラー・コードの場合は、それぞれのコンポーネント MIB のリファレンス・ページに記述されます。

TA_STATUS
: エラーの説明を格納する文字列値の FML32 フィールド。

TA_BADFLD
: エラーが特定のフィールドの値に起因する場合に、そのフィールドのフィールド識別子を格納する long 値の FML32 フィールド。エラーが複数のフィールドの値の組み合わせに起因する場合は、このフィールドのオカレンスが複数存在することがあります。

使用方法

インクルード・ファイル

コンポーネント MIB とインターフェイスを目的として記述されたアプリケーション・プログラムには、一定のヘッダ・ファイルをインクルードする必要があります。<fml32.h> は、FML32 型付きバッファのアクセスおよび更新に必要なマクロ、構造体、および関数のインターフェイスを定義します。<fml1632.h> は、汎用 FML インターフェイスのマクロ、構造体、および関数から FML32 バージョンへのマッピングを定義します。このヘッダ・ファイルのインクルードは任意です。<tpadm.h> は、このリファレンス・ページに含まれている FML32 フィールド名を定義します。さらに、任意のコンポーネント MIB 固有ヘッダ・ファイルをインクルードして、そのコンポーネント MIB 固有の FML32 フィールド定義にアクセスできるようにする必要があります。

例 :

```
#include <fml32.h>
#include <tpadm.h>
#include <cmib.h>  /* コンポーネント MIB ヘッダ */
```

バッファの割り当て

コンポーネント MIB と相互作用するには、FML32 型付きバッファから該当するサービスに要求を送る必要があります。ATMI 関数 tpalloc() は、FMLTYPE32 (<fml32.h> で定義) を使用してバッファを *type* 引数の値に割り当てます。FML32 バッファにはサブタイプがないため、tpalloc() の *subtype* 引数は NULL にできます。FML32 バッファのデフォルトの最小サイズは 1024 バイトです。tpalloc() の *size* 引数に 0 を指定すると、最小サイズのバッファが割り当てられます。より大きなバッファが必要な場合には、システム最小値より大きな値を *size* に指定して割り当てることができます。

例 :

```
rqbuf = tpalloc(FMLTYPE32, NULL, 0);
```

MIB 要求の作成

FML32 型付きバッファを割り当てたら、ユーザはそのバッファに汎用 MIB フィールドの値とコンポーネント MIB 固有の値を格納する必要があります。要求バッファへの値の追加に使用する最も一般的なインターフェイスは Fadd32() および Fchg32() です。要求バッファがいっぱいでフィールドを追加できない場合は、ATMI 関数 tprealloc() を使用してバッファを再割り当てする必要があります。

例 :

```
/*
 * エラー処理は含まない。bigger_size はシステム側で提供されるのではなく、
 * ユーザ側で指定。バッファを再利用する場合は、Fchg32 を使用して
 * フィールド・オカレンス 0 に設定する。
 */
if (Fchg32(rqbuf, TA_MIBFIELD, 0, "ABC", 0) == -1) {
  if (Ferror32 == FNOSPACE) {
    rqbuf = tprealloc(rqbuf, bigger_size);
    Fchg32(rqbuf, TA_MIBFIELD, 0, "ABC", 0);
  }
}
```

MIB 要求の制御

各コンポーネント MIB に固有の属性のほかに、コンポーネント MIB から要求された操作を制御する必須および任意の属性があります。これらの属性はこのリファレンス・ページに定義されています。

必須の汎用属性は TA_OPERATION と TA_CLASS の 2 つです。

TA_OPERATION は、アクセスする MIB 上で実行する操作を指定します。有効な操作は GET、GETNEXT 、および SET です。

TA_CLASS は、アクセスする MIB クラスを指定します。クラス名はコンポーネント MIB のリファレンス・ページで定義されています。TA_OPERATION が GETNEXT の場合には、TA_CURSOR 属性も指定する必要があります。
TA_CURSOR は、直前の GET または GETNEXT 操作で返されたフィールドです。このフィールドは、以降の要求時に検索位置を調べるために使用します。

任意属性の TA_OCCURS、TA_FLAGS、TA_FILTER、TA_MIBTIMEOUT、および TA_CURSORHOLD は、要求をさらに細かく指定するときに、必須属性に加えて使用することができます。

TA_OCCURS
: GET または GETNEXT 操作時に検索するオブジェクトの数を指定します。この属性を指定しない場合、スペースがあるかぎり、すべてのオカレンスが検索されます。

TA_FLAGS
: フラグ値を指定します。一部の汎用フラグはこのリファレンス・ページに、それ以外の汎用フラグはコンポーネント MIB のリファレンス・ページに定義されています。

TA_FILTER
: GET 操作で返される属性値を限定します。この属性を指定しない場合、使用可能なすべてのクラス属性値用の long 値 FML32 フィールドが返されます。

TA_MIBTIMEOUT
: 要求を満たすために必要なコンポーネント MIB サービス内の時間 (秒数) を指定します。0 以下の値を指定した場合、コンポーネント MIB サービスはブロッキング処理を実行できません。この値を指定しない場合、デフォルトで 20 に設定されます。

TA_CURSORHOLD
: 現在の GET または GETNEXT の要求が満たされた後、最初の GET 操作で生成されたシステム・スナップショットを処分せずに保持しておく時間 (秒数) を指定します。0 以下の値を指定した場合、現在の要求が満たされるとスナップショットが処分されます。この値を指定しない場合、デフォルトで 120 に設定されます。

例 :

```
/* 最初の 5 オブジェクトを取得 (GET) */
Fchg32(rqbuf, TA_OPERATION, 0, "GET", 0);
Fchg32(rqbuf, TA_CLASS, 0, "classname", 0);
n = 5;
Fchg32(rqbuf, TA_OCCURS, 0, n, 0);
/* 要求を作成、後述の「MIB 要求の送信」を参照 */
/* 応答は rpbuf に格納。カーソルも含まれる。*/
/*
 * 次の 5 オブジェクトを取得 (GETNEXT)。rpbuf から TA_CURSOR を転送。
 * すでに生成した rqbuf を再利用。要求後にスナップショットを破棄
 * (TA_CURSORHOLD を 0 に設定 )。
 */
Fchg32(rqbuf, TA_OPERATION, 0, "GETNEXT", 0);
Fchg32(rqbuf, TA_CURSOR, 0, Ffind32(rpbuf, TA_CURSOR, 0, NULL), 0);
n = 0;
Fchg32(rqbuf, TA_CURSORHOLD, 0, n, 0);
/* 要求を作成。後述の「MIB 要求の送信」を参照。*/
```

コンポーネント MIB フィールド

GET または GETNEXT で指定したコンポーネント MIB キー・フィールドは、オブジェクトの集合を選択する際に使用します。キー・フィールド以外のフィールドは、コンポーネント MIB では無視されます。

SET 操作で指定したコンポーネント MIB キー・フィールドは、更新する特定のオブジェクトを識別するために使用します。キー・フィールド以外のフィールドは、キー・フィールドで指定されたオブジェクトの更新値として処理されます。ユーザは、更新 (SET) が許可される前に、現在のオブジェクト・イメージと一致する必要のあるプレイメージを指定することもできます。ユーザは、要求の TA_FLAGS 属性の MIB_PREIMAGE ビットをセットすることでプレイメージを提供することを示します。更新するオブジェクトを指定するキー・フィールドは、プレイメージ (フィールド・オカレンス 0) から取得されます。キー・フィールドがポストイメージにも指定されている場合には、それらのフィールドが正確に一致している必要があります。一致していない場合は要求が失敗します。プレイメージのマッチングで考慮されるのは、クラスの一部であり、かつ入力バッファで指定された 2 つの属性値をもつ属性のみです。1 つの値しかもたない属性は、指定されたクラス・オブジェクト用に設定する新しい値として処理されます。

例 :

```
Fchg32(rqbuf, TA_OPERATION, 0, "GET", 0);
Fchg32(rqbuf, TA_CLASS, 0, "classname", 0);
Fchg32(rqbuf, TA_MIBKEY, 0, "keyvalue", 0);
n = 1;
Fchg32(rqbuf, TA_OCCURS, 0, n, 0); /* 最初に一致したオカレンスを取得
(GET)。*/
/* 要求を作成。後述の「MIB 要求の送信」を参照。応答は rpbuf に格納。*/
/* rpbuf をプレイメージとして使用し、一致する場合は
 * TA_MIBFIELD の値を更新。
 */
Fcpy32(newrq, rpbuf);
Fconcat32(newrq, rpbuf);   /* 2 番目に一致したコピーを追加。*/
Fchg32(newrq, TA_OPERATION, 0, "SET", 0);
n = MIB_PREIMAGE;
Fchg32(newrq, TA_FLAGS, 0, n, 0);
Fchg32(newrq, TA_MIBFIELD, 1, "newval", 0); /* ポストイメージ */
/* 要求を作成。後述の「MIB 要求の送信」を参照。*/
```

MIB 要求の送信

すべてのコンポーネント MIB 要求は、コア BEA Tuxedo コンポーネント MIB サービスである ".TMIB" を通ります。このサービスは、TM_MIB(5) 要求を処理するエージェントとしての役割を果たすだけでなく、他のコンポーネント MIB に対する要求を転送します。これにより、ユーザ側でサービス名を MIB やクラスとマッチングする必要がなくなります。サービス要求は、ATMI 内の任意の要求 / 応答指向サービス (tpcall(), tpacall() and tpenqueue()) を使用して生成できます。ユーザは、これらのインターフェ

イス関数に対して定義されたすべてのフラグと機能にアクセスできます。ここでの唯一の制約は、".TMIB" サービスをトランザクションの範囲外で呼び出す必要がある点です。つまり、トランザクション内で tpcall() や tpacall() を使用して管理要求を送信する場合、TPNOTRAN フラグを使用しないと異常終了 (TPETRAN) してしまいます。tpenqueue() を使用して要求を発行する場合は、TMQFORWARD サーバを -n オプションを指定して起動し、転送されるサービス要求をトランザクション境界の外で行えるようにする必要があります。

例 :

```
/* 上記に従って要求を作成。*/
/* 要求を送信し、応答を待機。*/
flags = TPNOTRAN | TPNOCHANGE | TPSIGRSTRT;
rval = tpcall(".TMIB", rqbuf, 0, rpbuf, rplen, flags);
/* 要求を送信し、記述子を取得。*/
flags = TPNOTRAN | TPSIGRSTRT;
cd = tpacall(".TMIB", rqbuf, 0, flags);
/* キューから要求を取り出す。qctl は設定済みと仮定。*/
flags = TPSIGRSTRT;
rval = tpenqueue("queue", ".TMIB", qctl, rqbuf, 0, flags);
```

MIB 応答の受信

コンポーネント MIB からの応答は、元の要求がどのように生成されたかに応じて 3 通りの方法で受信できます。元の要求が tpcall() で生成された場合、tpcall() が正常に終了すると、応答が受信されたことを示す値を返します。元の要求が tpacall() で生成された場合は、tpgetrply() を使用して応答を受信できます。元の要求が tpenqueue() で生成され、かつキュー制御構造体で応答キューが指定されている場合は、tpdequeue() を使用して応答を受信できます。これらの呼び出しでサポートされているフラグを適宜使用できます。

例 :

```
/* 上記に従って要求を作成。*/
/* 要求を送信し、応答を待機。*/
flags = TPNOTRAN | TPNOCHANGE | TPSIGRSTRT;
rval = tpcall(".TMIB", rqbuf, 0, rpbuf, rplen, flags);
/* 呼び出し記述子を使用して応答を受信。*/
flags = TPNOCHANGE | TPSIGRSTRT;
rval = tpgetrply(cd, rpbuf, rplen, flags);
/* TPGETANY を使用して応答を受信。バッファ・タイプの変更が必要な場合もある。
*/
flags = TPGETANY | TPSIGRSTRT;
```

```
rval = tpgetrply(rd, rpbuf, rplen, flags);
/* キューから要求を取り出す。qctl は設定済みと仮定。*/
flags = TPNOCHANGE | TPSIGRSTRT;
rval = tpdequeue("queue", "replyq", qctl, rpbuf, rplen, flags);
```

MIB 応答の解釈

管理要求に対しては、コンポーネント MIB 固有の属性のほかに、特定の汎用 MIB フィールドが返されることがあります。これらの追加属性は、元の要求の結果の特徴を示すもので、必要に応じて後続の要求で使用できる値を提供します。

GET または GETNEXT 操作が正常に終了すると以下の値が返されます。

- TA_CLASS

  クラス名。

- TA_OCCURS

  検索された一致オブジェクトの数。

- TA_MORE

  まだ検索されていない一致オブジェクトの数。

- TA_CURSOR

  以降の検索で提供されるカーソル。

- TA_ERROR

  負でない戻り値 TAOK に設定されます。

- 使用可能なコンポーネント MIB 固有の属性

  各属性のオカレンス 0 は 1 番目に検索されたオブジェクトを、オカレンス 1 は 2 番目に検索されたオブジェクトを表します (オカレンス 2 以降も同様)。この規則の例外は、コンポーネント MIB のリファレンス・ページに適宜記載されています。

SET 操作が正常に終了すると以下の値が返されます。

- TA_CLASS

  クラス名。

- TA_ERROR

負の値でない戻り値に設定します。TAOK は、要求が正常に終了したにもかかわらず、情報が更新されなかったことを示します。こうした現象は、変更が指定されていない場合や、指定された変更がオブジェクトの現在の状態と一致していない場合に発生します。TAUPDATED は、要求が正常に終了し、情報が更新されたことを示します。TAPARTIAL は、要求が成功したにもかかわらず、システム内の一部しか更新されなかったことを示します。こうした現象は、ネットワークの障害やメッセージの輻湊が原因で発生することがあり、更新されていないサイトはシステムによってできるだけ早く同期されます。

- 使用可能なコンポーネント MIB 固有の属性

  一度に 1 つのオブジェクトしか更新できないため、返されるオブジェクトは 1 つのみです。返された属性には、更新後のオブジェクトが反映されています。

操作が失敗すると、すべてのタイプで以下の値が返されます。

- 元の要求で指定されているフィールド。
- TA_ERROR

  失敗の原因を示す負の戻り値が設定されます。汎用エラー・コードは、このマニュアル・ページの「診断」のセクションに定義されています。コンポーネント MIB 固有の (ほかのエラー・コードや汎用コードと重複していない) エラー・コードは、各 MIB リファレンス・ページに定義されています。

- TA_BADFLD

  エラーの原因となったフィールドのフィールド識別子。

- TA_STATUS

  エラー状態の説明文。

制限事項 フィールドの複数のオカレンスをもつ FML32 バッファでは、オカレンスのシーケンス内に空のフィールドをもつことはできません。たとえば、オカレンス 1 の値をセットし、オカレンス 0 が存在していない場合、FML32 は FML32 定義の NULL 値でオカレンス 0 を自動的に作成します。FML32 定義の NULL 値は、数値フィールドに対しては 0、文字列フィールドに対しては長さゼロの (NULL) 文字列、文字フィールドに対しては文字 '•0' になります。このような制約があるため、異なる属性の集合をもつオブジェクトが返されるこ

とのある GET 操作では、オブジェクトの状態を正確に反映しない NULL の FML32 フィールドが含まれないよう、ユーザに返されたオブジェクトの集合を人為的に分割することがあります。

DOS、Windows、および OS/2 上のワークステーション・クライアントは、64K の FML32 バッファにリンクされています。このため、戻りバッファのサイズは、バッファあたり 64K に制限されています。

COBOL では FML32 バッファ・タイプが限定的にしかサポートされていないため、COBOL バージョンの ATMI では管理 API にアクセスできません。

コンポーネント MIB に対する要求を、アプリケーション・トランザクションの一部にすることはできません。したがって、アクティブ・トランザクション内でコンポーネント MIB に対して発行する tpcall() または tpacall() 呼び出しでは、呼び出し時に TPNOTRAN フラグを設定する必要があります。ただし、コンポーネント MIB への今後の送出に備え、トランザクション内で ATMI 関数 tpenqueue() を使用して要求をキューに登録することができます。この要求のキューへの登録はトランザクション内で実行されますが、コンポーネント MIB 内の処理はトランザクション内では実行されません。このコンテキストで TMQFORWARD(5) サーバを使用するためには、要求が非トランザクション・モードで MIB サービスに送出されるよう、-n コマンド行オプションを指定して TMQFORWARD を起動する必要があります。コンポーネント MIB サービスはトランザクション非対応であるため、TMQFORWARD で -d オプションを指定することもお奨めします。これにより、サービスが失敗しても、要求がリトライされることなく即座に失敗キューに送出されます。

汎用 MIB フィールドとコンポーネント MIB のフィールド識別子は 6,000 ～ 8,000 の範囲で割り当てられます。したがって、管理アクションとユーザ・アクションの両方を行うアプリケーションでは、フィールド識別子を適切に割り当てる必要があります。

クラスの説明

各クラスの説明セクションには、次の 4 つのサブセクションがあります。

概要
: このクラスに関連付けられている属性の概要

属性表
　クラスの各属性に関する名前、タイプ、パーミション、値、およびデフォルト値を示す表。属性表の形式については以下に示してあります。

属性の意味
　各属性の意味の説明

制限事項
　このクラスにアクセスし、このクラスを解釈する場合の制限事項

属性表の形式

前述のように、各クラスは 4 つの部分に分けて定義されています。その 1 つが属性表です。属性表はクラス内の属性のリファレンス・ガイドであり、管理者、オペレータ、一般ユーザがそれらの属性を使用してアプリケーションと対話するための方法を説明しています。属性表の各属性の説明には、5 つの構成要素 (名前、タイプ、パーミッション、値、デフォルト) があります。各要素については、以下を参照してください。

名前 :
　FML32 バッファ内のこの属性値の識別に使用する FML32 フィールド識別子。属性は、密接な関連のある属性が分類されて配置されています。この分類には特別な意味はなく、単に表を使いやすくすることを目的としています。名前や値の後に (r)、(k)、(x)、(*) が付いている場合があります。この表記の意味は以下のとおりです。

　(r)— 新しいオブジェクトを作成する際に必要なフィールド

　(k)— オブジェクトを検索するためのキー・フィールド

　(k)— オブジェクトを検索するための正規表現キー・フィールド

　(*)— オブジェクトを変更するための SET キーとなるフィールド

　クラスに対し、1 つ以上の SET キーで SET 操作を実行する場合 (上記の * を参照) は、SET キーとして定義された属性の値が 1 つ以上含まれている必要があります。指定する SET キーは、クラス内の 1 つのオブジェクトを一意に識別できるキーでなければなりません。SET キーは常にオブジェクト検索用のキー・フィールドであるため、わざわざ (k) と表記することはしていません。ただし、NEW オブジェクトを作成するときに SET キーが必ず必要というわけではなく、(r) は必要に応じて表記されています。

タイプ :

属性値のデータ型。データ型は、C 言語の表記法 (long、char、および string) で定義されます。プログラム内では、FML32 関数 Fldtype32() を使用してデータ型を判別できます。この関数は、データ型を表す FML32 の define (FLD_LONG、FLD_CHAR、および FLD_STRING) を返します (Fldtype、Fldtype32(3fml) を参照)。

パーミッション :

アクセスと更新のパーミッションは、UNIX システムのパーミッションと同様に、それぞれ 3 つのパーミッションからなる 3 つのグループに分けられます。ただし、属性表でこの 3 つのグループが表すのは、UNIX の場合のオーナ、グループ、その他に対するパーミッションではなく、管理者、オペレータ、その他に対するパーミッションです。各グループについて、以下の 3 つのパーミッション位置があります。

位置 1— 検索パーミッション

r	属性を検索できます。
R	オブジェクトの状態が ACTive または ACTive と同等のときにのみ属性を検索できます。どの状態が ACTive と同等か判別するには、各クラスの TA_STATE 属性値の説明を参照してください。この属性は一時的な情報を表し、オブジェクトをアクティブ化するたびに更新されます。
k	検索または更新用のキー・フィールドとして属性を指定できます。
K	属性は、オブジェクトの状態が ACTive または ACTive と同等のときに、検索または更新用のキー・フィールドとしてのみ指定できます。どの状態が ACTive と同等かどうかを判別するには、各クラスの TA_STATE 属性値の説明を参照してください。

位置 2— 非アクティブな更新パーミッション

w	オブジェクトの状態が INActive または INActive と同等のときに属性を更新できます。どの状態が INActive と同等かどうかを判別するには、各クラスの TA_STATE 属性値の説明を参照してください。

u	w パーミッション値と同様に属性を更新できます。ただし、u パーミッション文字で識別されるすべての属性値の組み合わせは、クラス内で一意でなければなりません。
U	w パーミッション値と同様に属性を更新できます。ただし、属性値はクラス内の属性に対して一意でなければなりません。

位置 3— アクティブな更新パーミッション

x	オブジェクトの状態が ACTive または ACTive と同等のときに属性を更新できます。どの状態が ACTive と同等かどうかを判別するには、各クラスの TA_STATE 属性値の説明を参照してください。
X	オブジェクトの状態が ACTive または ACTive と同等のときに属性を更新できます。どの状態が ACTive と同等かどうかを判別するには、各クラスの TA_STATE 属性値の説明を参照してください。この属性は一時的な情報を表し、オブジェクトをアクティブ化するたびに更新されます。
y	オブジェクトの状態が ACTive または ACTive と同等のときに属性を更新できます。ただし、変更がこのクラスまたは他のクラスのオブジェクトに反映されるタイミングに制約があります。詳細については、そのクラスの「属性の意味」のセクションを参照してください。どの状態が ACTive と同等かどうかを判別するには、各クラスの TA_STATE 属性値の説明を参照してください。

値 :

この属性に対して設定または検索 (あるいはその両方) が可能な値。以下に、属性値を表記する際の規則を示します。

LITSTRING	リテラル文字列値。
num	数値。
string[*x..y*]	長さが *x* 以上 *y* 以下の文字列値。末尾の NULL 文字はカウントしません。
LMID	string[1..30] (カンマを入れることはできません) の短縮形。論理マシン識別子を表します。

{*x*|*y*|*z*}	*x*、*y*、または *z* のいずれかを選択します。
{*x*|*y*|*z*}	*x*、*y*、または *z* のいずれかを選択します (選択しなくてもかまいません)。
{*x*|*y*|*z*},*	カンマで区切られたリストから *x*、*y* 、または *z* の 0 以上のオカレンスを選択します。
low = *num*	*low* 以上の数値。
low = *num high*	*low* 以上 *high* 未満の数値。
GET:	検索 (GET) 操作でキー値として返されるか、キー値として指定できる状態属性値。値は常に 3 文字からなる簡略名です。完全名は、該当するクラスの TA_STATE の説明文に示されています。入力指定は簡略名でも完全名でも可能です。大文字 / 小文字は区別されません。出力状態は、常に大文字の完全名で返されます。
SET:	更新 (SET) 操作で設定される状態属性値。GET と同様に簡略名を使用できます。

デフォルト値 :

新しいオブジェクトを作成するとき、つまり状態を INValid から NEW に変更するときに使用するデフォルト値。オブジェクトがアクティブなときにのみ必要になる属性、派生する属性、および使用可能な属性は N/A と表記されています。

TA_STATE の構文

TA_STATE 属性フィールドは、定義された各クラスのメンバです。この属性の意味はクラスごとに定義されています。TA_STATE 値は、多くの場合 3 文字の簡略名で指定できます。TA_STATE 値の完全名を表示する場合は、3 文字の簡略名を大文字で、残りの文字を小文字で示します。TA_STATE 値は、簡略名でも完全名でも入力できます。大文字 / 小文字は区別されません。TA_STATE 値の出力は常に大文字の完全名です。以下に、TA_STATE 属性の使用例を示します。

```
完全名   : ACTive
簡略名   : ACT
出力値  : ACTIVE
有効な入力値  : ACT、act、AcTiVe、active
```

T_CLASS クラスの定義

概要 T_CLASS クラスは、BEA Tuxedo システム・アプリケーション内の管理クラスの属性を表します。このクラスは、主にクラス名の識別に使用します。

属性表

表 39T_CLASS クラス定義の属性表

属性	タイプ	パーミッション	値	デフォルト値
TA_CLASSNAME(k)	string	r--r--r--	*string*	N/A
TA_STATE(k)	string	r--r--r--	GET: VAL SET:N/A	GET:N/A SET:N/A
TA_GETSTATES	string	r--r--r--	*string*	N/A
TA_INASTATES	string	r--r--r--	*string*	N/A
TA_SETSTATES	string	r--r--r--	*string*	N/A

(k)— オブジェクトを検索するためのキー・フィールド

属性の意味

TA_CLASSNAME:*string*
クラス名。

TA_STATE:
GET:
GET 操作は、選択した T_CLASS オブジェクトの情報を検索します。以下に示す状態は、GET 要求への応答で返される TA_STATE の意味を示します。以下に示されていない状態は返されません。

VALid	T_CLASS オブジェクトが定義されています。このクラスのすべてのオブジェクトはこの状態です。この状態は INActive と同等で、パーミッションのチェックに使用します。

SET:
SET 操作は、このクラスでは使用できません。

TA_GETSTATES:*string*
このクラスのオブジェクトに対して、または GET 操作の結果として返される可能性のある状態を '|' で区切ったリスト。状態は大文字の完全名で返されます。

TA_INASTATES:*string*
このクラスのオブジェクトに対して、または GET 操作の結果として返される可能性のある非アクティブと同等な状態を '|' で区切ったリスト。状態は大文字の完全名で返されます。

TA_SETSTATES:*string*
このクラスのオブジェクトに対して SET 操作の一部として設定される可能性のある状態を '|' で区切ったリスト。状態は大文字の完全名で返されます。

制限事項　なし

T_CLASSATT クラスの定義

概要 T_CLASSATT クラスは、管理属性の特性をクラス / 属性ごとに表します。

属性表

表 40T_CLASSATT クラス定義の属性表

属性	タイプ	パーミッション	値	デフォルト値
TA_CLASSNAME(r)(*)	string	ru-r--r--	*string*	N/A
TA_ATTRIBUTE(r)(*)	long	ru-r--r--	0 <= *num*	N/A
TA_STATE(k)	string	rw-r--r--	GET: VAL SET: {NEW | INV}	GET:N/A SET:N/A
TA_PERM(r)	long	rw-r--r--	0000 <= *num* <= 0777	N/A
TA_FACTPERM	long	r--r--r--	0000 <= *num* <= 0777	N/A
TA_MAXPERM	long	r--r--r--	0000 <= *num* <= 0777	N/A
TA_ATTFLAGS	long	r--r--r--	*long*	N/A
TA_DEFAULT	string	r--r--r--	*string*	N/A
TA_VALIDATION	string	r--r--r--	*string*	N/A

(k)—GET キー・フィールド
(r)— オブジェクトの作成に必要なフィールド (SET TA_STATE NEW)
(*)—GET/SET キー、SET 操作では 1 つ以上必要

属性の意味

TA_CLASSNAME:*string*
: クラス名。システムによって認識されているクラス名にのみアクセスできます。

TA_ATTRIBUTE:*long*
: システム提供のヘッダ・ファイル（たとえば tpadm.h）に定義されている属性フィールド識別子。

TA_STATE:

GET: VALid
GET 操作は、選択した T_CLASSATT オブジェクトの情報を検索します。以下に示す状態は、GET 要求への応答で返される TA_STATE の意味を示します。

VALid	T_CLASSATT オブジェクトが定義されています。このクラスのすべてのオブジェクトはこの状態です。この状態は INActive と同等で、パーミッションのチェックに使用します。

SET: {NEW | INValid}
SET 操作は、選択した T_CLASSATT オブジェクトのコンフィギュレーション情報を更新します。以下に示す状態は、SET 要求で設定される TA_STATE の意味を示します。以下に示されていない状態は設定できません。

NEW	アプリケーション用の T_CLASSATT オブジェクトを作成します。状態の変更は INValid 状態でのみ可能です。正常終了すると、オブジェクトの状態は VALid になります。
unset	T_CLASSATT オブジェクトを変更します。変更は、VALid 状態でのみ可能です。正常終了した場合、オブジェクトの状態は変わりません。
INValid	アプリケーション用の T_CLASSATT オブジェクトを削除またはリセットします。状態の変更は VALid 状態でのみ可能です。正常終了すると、オブジェクトの状態は INValid または VALid になります。組み込まれている (システムに明示的に認識されている) このクラスのオブジェクトは、この状態変更でデフォルトのパーミッションに戻り、VALid 状態のままとなります。クラス属性が明示的に認識されていないアドオン・コンポーネントに属するこのクラスのオブジェクトは、この状態変更で削除されて INValid 状態に遷移します。

TA_PERM: 0000 <= *num* <= 0777
: このクラス属性の組み合わせに対するアクセス・パーミッション。パーミッションを設定する際、要求された設定値がその属性に対して許されるパーミッションを超えていると、設定された実際の値が自動的に再設定されます。属性に許容される最大パーミッションは、管理者用に記述されているパーミッションをオペレータとその他の位置で繰り返したものです。たとえば、T_MACHINE クラスの TA_TYPE 属性のパーミッションは rw-r--r-- と記述されており、最大パーミッションは rw-rw-rw- です。

TA_FACTPERM: 0000 <= *num* <= 0777
: BEA Tuxedo システムの出荷時に設定されたこのクラス属性の組み合わせに対するパーミッション。このパーミッションは、オブジェクトの TA_STATE を INValid に変更する SET 操作の後に適用されます。

TA_MAXPERM: 0000 <= *num* <= 0777
: このクラス属性の組み合わせに対する最大パーミッション。

TA_ATTFLAGS:*long*
: この属性の特性を示す以下のフラグの一部または全部のビット単位の論理和。

  MIBATT_KEYFIELD
  : 属性はこのクラスのキー・フィールド。

  MIBATT_LOCAL
  : 属性はローカル情報を表す。

  MIBATT_REGEXKEY
  : 属性はこのクラスの正規表現キー・フィールド。

  MIBATT_REQUIRED
  : 属性はこのクラスの NEW オブジェクトを作成するときに必要。

  MIBATT_SETKEY
  : 属性はこのクラスの SET キー。

  MIBATT_NEWONLY
  : 属性は、TA_STATE を INValid から NEW に変えることによって NEW オブジェクトを作成するときにのみ、このクラスの非アクティブ同等のオブジェクトに対して書き込み可能。

TA_DEFAULT:*string*

このクラスの NEW オブジェクトを作成する際のこの属性のデフォルト。Admin API を使用して NEW オブジェクトを作成できないクラスでは、この属性は必ず長さゼロの文字列として返されます。また、NEW オブジェクトの作成時に SET できない属性も長さゼロの文字列として返されます。*long* 値をもつ属性には、long 値を表す文字列として返されるデフォルト値があります。属性の中には、ここで返される以下のような値で示される特殊な性質を持つものもあります。

Inherited:*Classname*[:*Attribute*]

属性のデフォルトは、指定したクラスの同名の属性から継承されます。*Attribute* を指定した場合は、同名の属性の 1 つからではなく、この属性から継承されます。

Required

属性は NEW オブジェクトを作成するときに必要です。

Special

属性には、デフォルト値を定義するための特別な規則があります。詳細については、該当するコンポーネント MIB のリファレンス・ページを参照してください。

TA_VALIDATION:*string*

新しい値が SET されるときに、このクラス / 属性の組み合わせに適用する検証規則を表す文字列。この文字列の形式は以下のいずれかになります。

CHOICES=*string1*|*string2*|...

示された選択肢の 1 つのみと一致する文字列属性値。

RANGE=*min-max*

min から *max* までの値でなければならない数値属性値。

SIZE=*min-max*

min から *max* までのバイト長でなければならない string または carray 属性値。

READONLY=Y

書き込み操作に対する検証規則を持たない読み取り専用属性。

SPECIAL=Y
: 特別な検証規則。詳細については、該当するコンポーネント MIB リファレンス・ページを参照してください。

UNKNOWN=Y
: 不明な検証規則。通常、詳細がコア・システムで認識されていない追加コンポーネントの属性エントリに関連付けられています。

MIB(5) に関する追加情報

制限事項 なし

診断 コンポーネント MIB への接続時には、2 つの一般的なタイプのエラーがユーザに返される場合があります。1 つは、管理要求に対する応答を検索する 3 つの ATMI 関数 (tpcall()、tpgetrply()、および tpdequeue()) が返すエラーです。これらのエラーは、それぞれの関数のリファレンス・ページに定義されています。

2 つ目は、要求がその内容に対応できるシステム・サービスに正常にルーティングされても、システム・サービス側でその要求を処理できないと判断されると、アプリケーション・レベルのサービス障害として返されるエラーです。このような場合、tpcall() と tpgetrply() は、tperrno() を TPESVCFAIL に設定してエラーを返し、以下のようにエラーの詳細を示す TA_ERROR、TA_STATUS、および TA_BADFLD フィールドと一緒に、元の要求を含む応答メッセージを返します。TMQFORWARD(5) サーバ経由でシステムに転送された要求に対してサービス障害が発生すると、元の要求で識別される異常終了キューに障害応答メッセージが追加されます (TMQFORWARD に対して -d オプションが指定されたと見なされる)。

管理要求の処理中にサービス・エラーが発生すると、TA_STATUS という FML32 フィールドにエラーの内容を説明したテキストが設定され、TA_ERROR という FML32 フィールドにはエラーの原因 (下記参照) を示す値が設定されます。TA_BADFLD に設定される値は、下記の各エラーに関する説明のなかで示します。以下のエラー・コードは、いずれもマイナスであることが保証されています。

[TAEAPP]
要求を正しく処理するにはアプリケーションによる操作が必要ですが、その操作を完了することができませんでした。アプリケーションの操作が必要になるのは、サーバを停止させる場合などです。

[TAECONFIG]
要求された操作に必要なコンポーネント MIB のコンフィギュレーション・ファイルにアクセスできませんでした。

[TAEINVAL]
指定したフィールドが無効です。フィールド識別子が無効であることを示すため、TA_BADFLD が設定されます。

[TAEOS]
要求の処理時にオペレーティング・システムのエラーが発生しました。TA_STATUS の値が、システム・エラー・コード errno の値で更新されます。

[TAEPERM]
ユーザが、書き込みパーミッションのない属性を SET しようとしたか、読み取りパーミッションのないクラスに対して GET を実行しようとしました。フィールド識別子がパーミッション・チェックにパスしなかったことを示すため、TA_BADFLD が設定されます。

[TAEPREIMAGE]
指定したプレイメージと現在のオブジェクトが一致しなかったため、SET 操作が失敗しました。フィールド識別子がプレイメージ・チェックにパスしなかったことを示すため、TA_BADFLD が設定されます。

[TAEPROTO]
管理要求が不正なコンテキストで発行されました。TA_STATUS には追加情報が格納されます。

[TAEREQUIRED]
必要なフィールド値が存在しません。フィールド識別子が見つからないことを示すため、TA_BADFLD が設定されます。

[TAESUPPORT]
現在使用しているバージョンのシステムでは、管理要求がサポートされていません。

[TAESYSTEM]
要求の処理時に BEA Tuxedo システムのエラーが発生しました。TA_STATUS は、エラー条件の詳細を示す値に更新されます。

[TAEUNIQ]
SET 操作で、更新の対象となる一意なオブジェクトを特定するクラス・キーが指定されていません。

[*other*]
それぞれのコンポーネント MIB に固有のその他のエラー・コードは、各コンポーネント MIB のリファレンス・ページに指定されています。これらのエラー・コードは、すべてのコンポーネント MIB 間でも、

ここで定義した汎用コードとの間でも相互に排他的であることが保証されています。

以下の診断コードは TA_ERROR で戻されるもので、管理要求が正常に完了したことを示します。これらのコードはマイナスでないことが保証されています。

[TAOK]
操作が成功しました。コンポーネント MIB のオブジェクトは更新されません。

[TAUPDATED]
操作が成功しました。コンポーネント MIB のオブジェクトは更新されました。

[TAPARTIAL]
操作は部分的に成功しました。コンポーネント MIB のオブジェクトは更新されました。

相互運用性 FML32 インターフェイスへのアクセス、および BEA Tuxedo システムのアプリケーション管理に使用できるコンポーネント MIB へのアクセスは、BEA Tuxedo システム・リリース 4.2.2 以降のバージョンで可能です。汎用 MIB 属性を定義するヘッダ・ファイルおよびフィールド・テーブルは、BEA Tuxedo リリース 5.0 以降で利用できます。各コンポーネント MIB に固有の相互運用性の問題については、それぞれのリファレンス・ページで説明しています。

移植性 BEA Tuxedo システムの MIB を使用した管理作業をサポートするために必要な既存の FML32 および ATMI 関数、さらにこのリファレンス・ページに定義するヘッダ・ファイルとフィールド・テーブルは、すべてのサポート対象ネイティブ・プラットフォームとワークステーション・プラットフォームで使用可能です。

使用例 汎用 MIB 処理とのインターフェイスにおいて既存の API を使用する例については、前述の使用方法のセクションを参照してください。詳しい使用例については、各コンポーネント MIB のリファレンス・ページで実際のコンポーネント MIB のクラスや属性を使用して説明しています。

ファイル
```
${TUXDIR}/include/tpadm.h,
${TUXDIR}/udataobj/tpadm
```

関連項目 tpacall(3c), tpalloc(3c), tpcall(3c), tpdequeue(3c), tpenqueue(3c), tpgetrply(3c), tprealloc(3c), FML 関数の紹介 , Fadd、Fadd32(3fml), Fchg、Fchg32(3fml), Ffind、Ffind32(3fml), AUTHSVR(5), TM_MIB(5), TMQFORWARD(5)

『BEA Tuxedo アプリケーションの設定』

『BEA Tuxedo アプリケーション実行時の管理』

『C 言語を使用した BEA Tuxedo アプリケーションのプログラミング』

『FML を使用した BEA Tuxedo アプリケーションのプログラミング』

nl_types(5)

名前 nl_types― ネイティブ言語データ型

形式

```
#include <nl_types.h>
```

機能説明 nl_types.h ヘッダ・ファイルには、以下の定義が含まれています。

nl_catd
: カタログを識別するために、メッセージ・カタログ関数 catopen()、catgets()、および catclose() で使用します。

nl_item
: langinfo() データの項目を識別するために nl_langinfo() で使用します。タイプ nl_item のオブジェクトの値は langinfo.h で定義されています。

NL_SETD
: メッセージ・テキストのソース・ファイルに $set 宣言が指定されていない場合に gencat() で使用します。この定数は、以降の catgets() への呼び出しの際に、識別子設定パラメータの値として使用できます。

NL_MGSMAX
: セット当たりのメッセージの最大数。

NL_SETMAX
: カタログ当たりのセットの最大数。

NL_TEXTMAX
: メッセージの最大サイズ。

DEF_NLSPATH
: カタログを見つけるためのデフォルトの検索パス。

関連項目 gencat(1)、catgets(3c)、catopen、catclose(3c)、nl_langinfo(3c)、langinfo(5)

servopts(5)

名前 servopts– サーバ・プロセスの実行時オプション

形式

```
AOUT CLOPT= [-A][-s{@filename|service[,service...][:func]}]
[-e stderr_file][-p [L][low_water][,[terminate_time]]
[:[high_water][,create_time]][-h][-l locktype][-n prio]
[-o stdout_file][-r][-t][ -- uargs][-v]
```

機能説明 servopts はコマンドではなく、BEA Tuxedo システムのサーバが認識する実行時オプションのリストです。

これらのオプションを使用するのは BEA Tuxedo システム提供のサーバ、または buildserver(1) コマンドによって作成されたアプリケーション提供のサーバです。

BEA Tuxedo システムでサーバを実行するには、アプリケーションのコンフィギュレーション・ファイルで指定したサーバ (およびその他のリソース) で機能する tmboot(1) コマンドと tmadmin(1) コマンドを使用します。servopts リストで選択したオプションは、コンフィギュレーション・ファイルのサーバに対して指定されます。認識されるオプションは以下のとおりです。

-A
: サーバの構築に使用したすべてのサービスを最初から提供します。-A は、BEA Tuxedo システム提供のサーバでサービスを指定する唯一の方法です。

-s { *@filename* | *service*[,*service*...][:*func*] }
: サーバのブート時に宣言するサービスの名前を指定します。ほとんどの場合、サービスは同じ名前を持つ関数によって実行されます。つまり、x サービスは関数 x によって実行されます。たとえば、次のように指定したとします。

  ```
  -s x,y,z
  ```

  この場合、サービス x、y、および z を最初から提供するサーバが実行され、各サービスは同じ名前を持つ関数によって処理されます。その他のケースでは、異なる名前の関数でサービスが実行されることもあります。たとえば、次のように指定したとします。

```
-s x,y,z:abc
```

この場合、初期サービスが x、y、z であるサーバが実行され、各サービスは関数 abc によって処理されます。

カンマとカンマの間に空白を入れてはいけません。関数名の前にはコロンを付けます。サービス名および暗黙のファンクション名は 15 文字以下でなければなりません。明示的関数名 (コロンの後に指定する名前) は、128 文字まで使用できます。この文字数より長い名前が指定された場合は、警告メッセージが表示されて短縮されます。`tmadmin(1)` または `TM_MIB(5)` によりファイルを取得した場合は、名前の最初の 15 文字だけが表示されます

-s オプションでファイル名を指定するには、ファイル名の前に「@」文字を付けます。このファイルの各行は、`-s` オプションの引数と見なされます。このファイルには、コメントを入れることができます。コメントは必ず「#」または「:」で始めます。`-s` オプションは何度でも指定できます。

サーバ・ロード・モジュール内のサービス名と処理関数を実行時に関連付けることを、動的サービス機能と呼びます。サーバの実行中に提供するサービスのリストを変更するには、`tmadmin advertise` コマンドを使用します。

「.」で始まるサービス名はシステム・サーバ用に予約されています。予約済みのサービスをアプリケーション・サーバで指定すると、サーバを正常に起動できなくなります。

`-e`

サーバの標準エラー出力ファイルとしてオープンするファイルの名前を指定します。このオプションを指定すると、サーバを再起動しても以前と同じ標準エラー出力ファイルが使用されます。このオプションを指定しない場合は、`stderr` というデフォルトのファイルが、`$APPDIR` で指定したディレクトリに作成されます。

`-p [L][`*`low_water`*`][,[`*`terminate_time`*`]][:[`*`high_water`*`][,`*`create_time`*`]]`

このオプションを使用すると、シングル・スレッドの RPC サーバおよび会話型サーバの両方で、サーバの自動生成と自動廃棄をサポートできます。RPC サーバの場合、MAX に 1 より大きい値を指定した MSSQ でこのオプションを使用する必要があります。会話型サーバの場合、MAX は 1 より大きい値でなければなりません。

サーバの生成および消滅は、キュー上の「サーバあたり」の要求の数によって決まります。ただし、RPC サーバでロード [L] 引数を使用している場合は、各要求のロード・ファクタも考慮されます。

注記　UNIX プラットフォームのみ―alarm() システム・コールは、サーバ・プール管理下で実行しているサーバではうまく機能しません。アイドル状態のサーバを終了するコードでは alarm() 呼び出しを使用するため、Usignal() への呼び出しがエラーでない場合でも、ユーザが独自のシグナル・ハンドラを確立するために作成したコードは異常終了します。

-p オプションの引数は、以下のように使用しているサーバの種類によって意味が異なります。

RPC サーバ

L

ロード引数は、RPC サーバでのみ機能します。また、ロード・バランシングをオンにした SHM モードでしか機能しません。サーバを生成するかどうかは、サーバあたりのメッセージ数ではなく、要求のロードに基づいて決定されます。SHM/LDBAL=Y が設定されていない場合、ユーザ・ログ・メッセージ (LIBTUX_CAT:1542) が出力され、生成および廃棄は発生しません。

low_water、*terminate_time*、*high_water*、および *create_time*

これらの引数は、RPC サーバをサーバあたりのメッセージ数に基づいて生成または廃棄する際の制御に使用します。ロードが *create_time* 秒以上にわたって *high_water* を超えると、新しいサーバが生成されます。ロードが *terminate_time* 秒以上にわたって *low_water* を下回ると、1 つのサーバが廃棄されます。*low_water* のデフォルト値は、MSSQ 上のサーバあたり 1 メッセージの平均値、または負荷 50 です。*high_water* のデフォルト値は、サーバあたり 2 メッセージの平均値、または負荷 100 です。*create_time* のデフォルト値は 50 秒、*terminate_time* のデフォルト値は 60 秒です。

会話型サーバ

L
ロード・オプションは会話型サーバには適用されません。

注記　BEA Tuxedo 以降では、マルチスレッドまたは非 MSSQ の会話型サーバの自動生成に制限はありません。ただし、これらのサーバには自動廃棄機能は実装されません。

low_water、*terminate_time*、*high_water*、および *create_time*
これらの引数は、会話型サーバを生成または非アクティブ化する際に使用します。会話型サーバは、一般に RPC サーバより長時間実行するため、現時点で会話に関与しているサーバの *low_water* の最小パーセンテージと *high_water* の最大パーセンテージをチェックします。これらのパーセンテージが、それぞれに関連付けられた時間パラメータ terminate_time または create_time の値を超えると、サーバの数が最小数または最大数に達していない限りサーバが生成または廃棄されます。

また、時間パラメータに値 0 秒を指定すれば、パーセンテージを超えたことが検出されると同時にサーバを生成または廃棄できます。*low_water* パーセンテージのデフォルト値は 0%、*high_water* のパーセンテージは 80% です。terminate_time のデフォルト値は 60 秒、create_time のデフォルト値は 0 秒です。

-h
ハングアップの影響を受けないサーバを実行しないようにします。このオプションを指定しない場合、サーバはハングアップ・シグナルを無視します。

-l *locktype*
サーバをロックします。*locktype* の引数は t、d、または p です。どの引数を使用するかは、ロックの対象がテキスト (TXTLOCK) であるか、データ (DATLOCK) であるか、プロセス全体 (テキストおよびデータ - PROCLOCK) であるかによって決まります。詳細については、plock(2) を参照してください。ルートとして実行されていないサーバはロックできません。また、いったんロックされたサーバのロックを解除することはできません。

-n *prio*

prio 引数に応じ、サーバに対して nice を実行します。プロセスに高い優先順位 (負の引数) を付与するには、そのプロセスを root の UID で実行する必要があります。詳細については nice(2) を参照してください。

-o *stdout_file*

サーバの標準出力ファイルとしてオープンするファイルの名前を指定します。このオプションを指定すると、サーバを再起動しても以前と同じ標準出力ファイルが使用されます。このオプションを指定しない場合は、stdout というデフォルトのファイルが、$APPDIR で指定したディレクトリに作成されます。

-r

実行したサービスのログを、標準エラー出力ファイルに記録します。このログを分析するには、txrpt(1) コマンドを使用します。-r オプションを使用する場合は、ULOGDEBUG 変数が y に設定されていないことを確認してください。ULOGDEBUG 変数が「y」に設定されていると、デバッグ・メッセージが stderr に送信されず、txrpt がファイル内のデバッグ・メッセージを間違って解釈してしまいます。

-t

BEA Tuxedo 7.1 以降のアプリケーションのサーバと、リリース 7.1 より前の BEA Tuxedo ソフトウェアとの相互運用を可能にします。サーバとしては、ワークステーション・リスナ (WSL) プロセス、ドメイン・ゲートウェイ (GWTDOMAIN) プロセス、システム・プロセス、またはアプリケーション・サーバ・プロセスを使用できます。ワークステーション・リスナ・プロセスの場合は、-t オプションを使用して起動すると、すべてのワークステーション・ハンドラ (WSH) プロセスで相互運用が可能になります。

--

システムが認識する引数の最後と、サーバ内のサブルーチンに渡す引数の最初をマークします。このオプションが必要になるのは、ユーザがアプリケーション固有の引数をサーバに渡す必要がある場合のみです。システムが認識するオプションを -- の前に、アプリケーションの引数をその後に指定します。アプリケーションの引数は、ユーザが定義した tpsvrinit() 関数で処理できますが、引数の解析には getopt() を使用します。すべてのシステム引数は tpsvrinit() への呼び出しの前に処理されるため、呼び出しの際には外部整数 optind

がユーザのフラグの開始点を指しています。-- 引数の後であれば、同じオプション文字 (たとえば -A) をアプリケーション固有の意味付けで再使用してもかまいません。

-v

サービス名と関数名のリストを標準出力に出力します。次のようなコメント行から始まります。

```
#
# サーバに組み込まれるサービスおよび対応ハンドラ関数のリスト
#
<servicename>:<functionname><NEWLINE>
<servicename>:<functionname><NEWLINE>
<servicename>:<functionname><NEWLINE>
.. . .
. . . .
```

で始まる最初の 3 行はコメントです。その後の各行は、実行可能ファイルに組み込まれるサービス名とそれに対応する関数名を示しています。buildserver コマンド行に -s: *functionname* が指定されている場合、その行の *servicename* フィールドは空文字列にできます。*functionname* フィールドは必ず指定します。

注記　BEA Tuxedo システムの実行時には、各サーバのそれぞれのコマンド行に次のオプションが自動的に追加されます。

```
-c dom=domainid
```

-c オプションを使用すると、指定したドメイン ID を示すコメント行を、そのドメインに関連付けられたプロセスで通知されるすべてのコマンド出力 (たとえば ps コマンドの出力) に追加できます。複数のドメインを管理する管理者は、このコメントによって、複数のドメインを参照する単一の出力ストリームを理解しやすくなります。

使用例　UBBCONFIG(5) の使用例を参照してください。

関連項目　buildserver(1)、tmadmin(1)、tmboot(1)、txrpt(1)、tpsvrinit(3c)、UBBCONFIG(5)

『BEA Tuxedo アプリケーションの設定』

『BEA Tuxedo アプリケーション実行時の管理』

UNIX システム・リファレンス・マニュアルの nice(2)、plock(2)、
getopt(3)

TM_MIB(5)

名前 TM_MIB—BEA Tuxedo システムの管理情報ベース

形式

```
#include <fml32.h>
#include <tpadm.h>
```

機能説明 BEA Tuxedo システムの MIB は、アプリケーションの基本的な側面を設定および管理するための一連のクラスを定義します。MIB を使用することで、マシン、サーバ、ネットワークなどを管理できます。

管理要求のフォーマットと管理応答の解釈を行うには、TM_MIB(5) を共通 MIB リファレンス・ページ MIB(5) と一緒に使用します。このリファレンス・ページで説明するクラスや属性を使用し、MIB(5) の説明に従ってフォーマットした要求を使用すると、アクティブなアプリケーションの既存の ATMI インターフェイスの 1 つを使用して管理サービスを要求できます。非アクティブなアプリケーションは、tpadmcall() 関数インターフェイスを使用して管理することもできます。TM_MIB(5) のすべてのクラス定義の追加情報については、544 ページの「TM_MIB(5) に関する追加情報」を参照してください。

TM_MIB(5) は、次のクラスで構成されています。

表 41TM_MIB のクラス

クラス名	管理対象
T_BRIDGE	ネットワーク接続
T_CLIENT	クライアント
T_CONN	会話
T_DEVICE	デバイス
T_DOMAIN	グローバル・アプリケーションの属性
T_FACTORY	ファクトリ
T_GROUP	サーバ・グループ

表 41TM_MIB のクラス (続き)

クラス名	管理対象
T_IFQUEUE	サーバ・キュー・インターフェイス
T_INTERFACE	インターフェイス
T_MACHINE	マシン固有の属性
T_MSG	メッセージ・キュー
T_NETGROUP	ネットワーク・グループ
T_NETMAP	ネットグループへのマシン
T_QUEUE	サーバのキュー
T_ROUTING	ルーティング基準
T_SERVER	サーバ
T_SERVERCTXT	サーバ・コンテキスト
T_SERVICE	サービス
T_SVCGRP	サービス・グループ
T_TLISTEN	BEA Tuxedo システム・リスナ
T_TLOG	トランザクション・ログ
T_TRANSACTION	トランザクション
T_ULOG	ユーザ・ログ

各クラスの説明は、以下の 4 つのセクションで構成されています。

- 概要 — このクラスに関連付けられている属性の概要について説明します。
- 属性表 — 属性表の形式を以下に要約します。詳細については MIB(5) で説明します。
- 属性の意味 — クラスに含まれる各属性の意味を定義します。

- 制限事項 — このクラスにアクセスし、このクラスを解釈する場合の制限事項について説明します。

属性表の形式

以降のセクションでは、この MIB に含まれる各クラスが 4 つの部分に分けて定義されています。その 1 つが属性表です。属性表はクラス内の属性のリファレンス・ガイドであり、管理者、オペレータ、一般ユーザがそれらの属性を使用してアプリケーションと対話するための方法を説明しています。

属性表の各属性の説明には、5 つの列 (名前、タイプ、パーミッション、値、デフォルト) があります。各要素については、MIB(5) を参照してください。

TA_FLAGS 値

MIB(5) は、共通 TA_FLAGS 属性を定義します。この属性は long 型で、共通 MIB フラグ値とコンポーネント MIB 固有フラグ値の両方を持ちます。以下は、サポートされる TM_MIB(5) 固有フラグ値です。これらのフラグ値は、共通 MIB フラグと一緒に使用する必要があります。

TMIB_ADMONLY
: T_MACHINE オブジェクトの状態を INActive から ACTive に変える際に、管理プロセスのみをアクティブにすることを示すために使用します。

TMIB_APPONLY
: T_MACHINE オブジェクトをアクティブまたは非アクティブにする際に、アプリケーション・プロセスのみを考慮することを示すために使用します。このフラグは、T_SERVER と T_SERVERCTXT での検索を、アプリケーション・サーバに限定するためにも使用できます。

TMIB_CONFIG
: 要求を満たす際に、設定済みのグループおよびサーバのみを考慮することを示すために使用します。

TMIB_NOTIFY
: T_MACHINE、T_GROUP、または T_SERVER オブジェクトをアクティブまたは非アクティブにする際に、選択した各サーバ・オブジェクトをアクティブ化または非アクティブ化する直前および直後に任意通知型のメッセージが発信元のクライアントに送信されるようにするために使用します。

FML32 フィールド・テーブル

このリファレンス・ページで説明する属性のフィールド・テーブルは、システムにインストールした BEA Tuxedo システム・ソフトウェアのルート・ディレクトリからの相対パスで指定される udataobj/tpadm ファイルにあり

ます。${TUXDIR}/udataobj ディレクトリは、FLDTBLDIR 環境変数で指定されるコロン区切りのリストにアプリケーションによって追加される必要があり、フィールド・テーブル名 tpadm は、FIELDTBLS 環境変数で指定されるカンマ区切りのリストに追加される必要があります。

制限事項

この MIB のヘッダ・ファイルとフィールド・テーブルには、BEA Tuxedo リリース 6.1 以降のサイト (ネイティブとワークステーションの両方) からのみアクセスできます。

ワークステーションによるこの MIB へのアクセスは、実行時のみのアクセスに制限されており、関数 tpadmcall(3c) はワークステーションではサポートされません。

プレイメージ処理 (MIB_PREIMAGE フラグ・ビットのセット) を目的として、グローバル属性を持つクラスのローカル属性が考慮されることはありません。また、インデックス付きのフィールド、およびそれらのフィールドとともに送出されるインデックスも考慮されません (たとえば、T_TLOG クラス、TA_TLOGCOUNT、TA_TLOGINDEX、TA_GRPNO、TA_TLOGDATA など)。

T_BRIDGE クラスの定義

概要 T_BRIDGE クラスは、アプリケーションを構成する論理マシン間の接続性に関する実行時の属性を表します。これらの属性の値は、接続の状態および統計値を表します。

属性表

表 42TM_MIB(5): T_BRIDGE **クラス定義の属性表**

属性 (注 1)	タイプ	パーミッション	値	デフォルト値
TA_LMID(*)(注 2)	string	r--r--r--	"*LMID1*[,*LMID2*]"	N/A
TA_NETGROUP(k)(注 3)	string	R--R--R--	*string*[1..30]	"DEFAULTNET"
TA_STATE(k)	string	rwxrwxr--	GET: "{ACT | INA | SUS | PEN}" SET: "{ACT | INA | SUS | PEN}"	N/A N/A
TA_CURTIME	long	R--R--R--	0 <= *num*	N/A
TA_CONTIME	long	R-XR-XR--	0 <= *num*	N/A
TA_SUSPTIME	long	rwxrwxr--	0 <= *num*	300 (注 4)
TA_RCVDBYT	long	R-XR-XR--	0 <= *num*	N/A
TA_SENTBYT	long	R-XR-XR--	0 <= *num*	N/A
TA_RCVDNUM	long	R-XR-XR--	0 <= *num*	N/A
TA_SENTNUM	long	R-XR-XR--	0 <= *num*	N/A
TA_FLOWCNT	long	R-XR-XR--	0 <= *num*	N/A
TA_CURENCRYPTBIT	string	R--R-----	"{0 | 40 | 56 | 128}" (注 5)	N/A

(k)—GET キー・フィールド
(*)—GET/SET キー。SET 操作では 1 つ以上必要

注 1 T_BRIDGE クラスのすべての属性はローカル属性です。
注 2 TA_LMID 属性は、SET 操作に対してはすべて (*LMID1*、*LMID2*) を指定する必要があります。

注3 SET 操作は、BEA Tuxedo リリース 6.4 では TA_NETGROUP DEFAULTNET しか使用できません。GET 操作は、両方の LMID 値用に定義された TA_NETGROUP を使用できます。
注4 TA_SUSPTIME を SET できるのは、TA_STATE がすでに SUSPENDED になっている場合か、これから SUSPENDED に SET しようとしている場合に限られます。
注5 リンク・レベルの暗号化値の 40 ビットは、下位互換性を維持するために提供されています。

属性の意味

TA_LMID: "*LMID1*[,*LMID2*]"

ネットワーク接続の接続元の論理マシン識別子 (*LMID1*) と接続先の論理マシン識別子 (*LMID2*)。

TA_NETGROUP: *string*[1 . .30]

ネットワーク・グループの論理名。接続先と接続元の TA_LMID マシン識別子が同じ TA_NETGROUP である場合、T_BRIDGE クラスは TA_NETGROUP ごとに関連しているフィールドのすべてのインスタンスを提供します。TA_NETGROUP は、GET 要求のキー・フィールドとして使用できます。BEA Tuxedo リリース (リリース 6.4) の SET 操作では、DEFAULTNET 以外の TA_NETGROUP 値は使用できません。

TA_STATE:

GET: "{ACTive | INActive | SUSpended | PENding}"

GET 操作は、選択した T_BRIDGE オブジェクトの実行時情報を検索します。論理マシン識別子を 1 つしか持たない TA_LMID 属性値は、アプリケーションのほかのマシンに対する *LMID1* からのすべてのアクティブな接続と一致します。この場合、取得した各レコードには、接続先の LMID が書き込まれて拡張された TA_LMID 属性値が含まれます。以下に示す状態は、GET 要求への応答で返される TA_STATE の意味を示します。

ACTive	接続が確立され、アクティブな状態です。
INActive	接続は非アクティブな状態です。この状態が返されるのは、特定の接続で状態を要求した場合、つまり、TA_LMID 属性で指定した 2 つの LMID および接続元の論理マシンにアクセス可能な場合のみです。

SUSpended	確立された接続がエラー条件の発生によって終了し、再接続が少なくとも TA_SUSPTIME 属性値に指定した時間だけ中断されていることを示します。この状態は、パーミッションの決定においては ACTive と同等です。
PENding	非同期接続が要求されていますが、まだ完了していません。接続要求の最終的な結果はまだ確定していません。

SET: "{ACTive | INActive | SUSpended | PENding}"

SET 操作は、選択した T_BRIDGE オブジェクトの実行時情報を更新します。以下に示す状態は、SET 要求で設定される TA_STATE の意味を示します。以下に示されていない状態は設定できません。

unset	既存の T_BRIDGE オブジェクトを変更します。この組み合わせは、ACTive 状態または SUSpended 状態でのみ可能です。正常終了した場合、オブジェクトの状態は変わりません。
ACTive	指定した論理マシン間の接続を確立することで、T_BRIDGE オブジェクトをアクティブにします。論理マシンを 1 つしか指定しなかった場合、どちらかのマシンがアクティブでない場合、および接続元の論理マシンにアクセスできない場合、この操作は異常終了します。T_BRIDGE オブジェクトが非同期接続を確立している間、ブリッジ・プロセスは別の処理を行います。状態を PENding に変更することを推奨します。状態の変更は、状態が INActive または SUSpended である場合のみ可能です。この状態遷移に対するパーミッションの決定に際しては、アクティブなオブジェクトのパーミッション (--x--x--x) が考慮されます。正常に終了すると、オブジェクトの状態は PENding になります。

INActive	指定した論理マシン間の接続をクローズすることで、T_BRIDGE オブジェクトを非アクティブにします。この操作は、マシンを 1 つしか指定しなかった場合、および 2 つのマシンが接続されていない場合は異常終了します。状態の変更は、ACTive 状態でのみ可能です。正常に終了すると、オブジェクトの状態は INActive になります。
SUSpended	指定した論理マシン間の接続を切断し、指定した値を TA_SUSPTIME 属性に設定することで、T_BRIDGE オブジェクトを中断します。状態の変更は、ACTive 状態でのみ可能です。正常に終了すると、オブジェクトの状態は SUSpended になります。制限事項 : レポートされる統計値は、接続元の論理マシン側の統計値です。これらの統計値をリセットすると、同じ接続について接続先の論理マシンからレポートされる統計値との同期が失われます。
PENding	指定した論理マシン間の非同期接続を確立することで、T_BRIDGE オブジェクトをアクティブにします。論理マシンを 1 つしか指定しなかった場合、どちらかのマシンがアクティブでない場合、および接続元のマシンにアクセスできない場合、この操作は異常終了します。PENding 状態では、接続要求が成功したか失敗したかは判別されません。ただし、接続が未処理でも、ブリッジ・プロセスはほかのイベントやデータの処理を継続します。状態の変更は、状態が INActive または SUSpended である場合のみ可能です。この状態遷移に対するパーミッションの決定に際しては、アクティブなオブジェクトのパーミッション (--x--x--x) が考慮されます。正常に終了すると、オブジェクトの状態は PENding になります。

TA_CURTIME: 0 <= *num*

T_BRIDGE:TA_LMID で time(2) システム・コールから返される 1970 年 1 月 1 日の 00:00:00 UTC から現在までの時間 (単位は秒)。この属性は、以下の属性値からの経過時間を算出するために使用できます。

TA_CONTIME: 0 <= *num*

T_BRIDGE:TA_LMID で time(2) システム・コールから返される 1970 年 1 月 1 日の 00:00:00 UTC から、この接続が初めて確立された時点までの時間 (単位は秒)。オープンされている経過時間 (秒) は、TA_CURTIME - TA_CONTIME を使用して計算できます。

TA_SUSPTIME: 0 <= *num*

この接続の保留の残り時間 (単位は秒)。この時間が経過すると、接続の TA_STATE は自動的に INACTIVE に変わり、通常のアプリケーション・トラフィックによってアクティブにできます。

TA_RCVDBYT: 0 <= *num*

接続先の論理マシンから接続元の論理マシンに送信されたバイト数。

TA_SENTBYT: 0 <= *num*

接続元の論理マシンから接続先の論理マシンに送信されたバイト数。

TA_RCVDNUM: 0 <= *num*

接続先の論理マシンから接続元の論理マシンに送信されたメッセージの数。

TA_SENTNUM: 0 <= *num*

接続元の論理マシンから接続先の論理マシンに送信されたメッセージの数。

TA_FLOWCNT: 0 <= *num*

接続に対してフロー制御が発生した回数。

TA_CURENCRYPTBITS: "{0 | 40 | 56 | 128}"

このリンクの現在の暗号化レベル。レベルは、リンクの確立時にマシン間で調整されます。

注記　リンク・レベルの暗号化の値 40 は、下位互換性を維持するために提供されています。

制限事項　なし

T_CLIENT クラスの定義

概要 T_CLIENT クラスは、アプリケーション内のアクティブなクライアントの実行時属性を表します。これらの属性値により、実行中のアプリケーション内のクライアントのアクティビティを識別してトラッキングできます。

属性表

表 43TM_MIB(5): T_CLIENT クラス定義の属性表

属性 (注 1)	タイプ	パーミッション	値	デフォルト値
TA_STATE(k)	string	R-XR-XR--	GET: "{ACT | SUS | DEA}" SET: "{ACT | SUS | DEA}"	N/A N/A
TA_CLIENTID(*)	string	R--R--R--	*string*[1..78]	N/A
TA_CLTNAME(k)	string	R--R--R--	*string*[0..30]	N/A
TA_IDLETIME(k)	long	R--R--R--	0 <= *num*	N/A
TA_LMID(k)	string	R--R--R--	*LMID*	N/A
TA_PID(k)	long	R--R--R--	1 <= *num*	N/A
TA_CONTEXTID	long	R--R--R--	-2 <= *num* < 30,000	N/A
TA_SRVGRP(k)	string	R--R--R--	*string*[0..30]	N/A
TA_USRNAME(k)	string	R--R--R--	*string*[0..30]	N/A
TA_WSC(k)	string	R--R--R--	"{Y | N}"	N/A
TA_WSH(k)	string	R--R--R--	"{Y | N}"	N/A
TA_WSHCLIENTID(k)	string	R--R--R--	*string*[1..78]	N/A
TA_RELEASE	long	R--R--R--	0 <= *num*	N/A
TA_WSPROTO	long	R--R--R--	0 <= *num*	N/A
TA_NUMCONV	long	R-XR-XR--	0 <= *num*	N/A
TA_NUMDEQUEUE	long	R-XR-XR--	0 <= *num*	N/A

表 43TM_MIB(5): T_CLIENT **クラス定義の属性表 (続き)**

属性 (注 1)	**タイプ**	**パーミッション**	**値**	**デフォルト値**
TA_NUMENQUEUE	long	R-XR-XR--	0 <= *num*	N/A
TA_NUMPOST	long	R-XR-XR--	0 <= *num*	N/A
TA_NUMREQ	long	R-XR-XR--	0 <= *num*	N/A
TA_NUMSUBSCRIBE	long	R-XR-XR--	0 <= *num*	N/A
TA_NUMTRAN	long	R-XR-XR--	0 <= *num*	N/A
TA_NUMTRANABT	long	R-XR-XR--	0 <= *num*	N/A
TA_NUMTRANCMT	long	R-XR-XR--	0 <= *num*	N/A
TA_CMTRET	string	R--R--R--	"{COMPLETE | LOGGED}"	N/A
TA_CURCONV	long	R--R--R--	0 <= *num*	N/A
TA_CURENCRYPTBIT	string	R--R-----	"{0 | 40 | 56 | 128}" (注 2)	N/A
TA_CURREQ	long	R--R--R--	0 <= *num*	N/A
TA_CURTIME	long	R--R--R--	1 <= *num*	N/A
TA_LASTGRP	long	R--R--R--	1 <= *num* < 30,000	N/A
TA_NADDR	string	R--R--R--	*string*[1..256] (注 3)	N/A
TA_NOTIFY	string	R--R--R--	"{DIPIN | SIGNAL | THREAD | IGNORE}"	N/A
TA_NUMUNSOL	long	R--R--R--	0 <= *num*	N/A
TA_RPID	long	R--R--R--	1 <= *num*	N/A
TA_TIMELEFT	long	R--R--R--	0 <= *num*	N/A
TA_TIMESTART	long	R--R--R--	1 <= *num*	N/A
TA_TRANLEV	long	R--R--R--	0 <= *num*	N/A

表 43TM_MIB(5): T_CLIENT **クラス定義の属性表 (続き)**

属性 (注 1)	**タイプ**	**パーミッション**	**値**	**デフォルト値**
(k)—GET キー・フィールド (*)—GET/SET キー。SET 操作では 1 つ以上必要				

注 1 T_CLIENT クラスのすべての属性はローカル属性です。
注 2 リンク・レベルの暗号化値の 40 ビットは、下位互換性を維持するために提供されています。
注 3 BEA Tuxedo 8.0 以前では、この属性の文字列の最大長は 78 バイトです。

属性の意味 TA_STATE:

GET: "{ACTive | SUSpended | DEAd}"

GET 操作は、選択した T_CLIENT オブジェクトの実行時情報を検索します。クライアント情報は、ローカルの掲示板テーブルにしか記録されません。したがって、パフォーマンスを最大にするには、クライアントの状態の照会にはできる限りキー・フィールドを使用する必要があります。以下に示す状態は、GET 要求への応答で返される TA_STATE の意味を示します。

ACTive	T_CLIENT オブジェクトの状態はアクティブです。これは、クライアントがビジーかアイドルかを示すものではありません。TA_CURCONV 属性または TA_CURREQ 属性の取得値がゼロでない場合は、クライアントがビジーであることを示します。
SUSpended	T_CLIENT オブジェクトの状態がアクティブで、次のサービス要求 (tpcall() または tpacall()) の実行および新たな会話の開始 (tpconnect()) が中断されていることを示します。詳細については、後述の SET SUSpended を参照してください。この状態は、パーミッションの決定においては ACTive と同等です。

DEAd	T_CLIENT オブジェクトが掲示板ではアクティブと識別されているにもかかわらず、異常終了が原因で現在は実行されていないことを示します。この状態が保持されるのは、クライアントのローカル BBL が異常終了を検知し、クライアントの掲示板のリソースをクリーン・アップするまでです。この状態は、パーミッションの決定においては ACTive と同等です。

SET: "{ACTive | SUSpended | DEAd}"

SET 操作は、選択した T_CLIENT オブジェクトの実行時情報を更新します。以下に示す状態は、SET 要求で設定される TA_STATE の意味を示します。以下に示されていない状態は設定できません。

ACTive	SUSpended 状態の T_CLIENT オブジェクトをアクティブにします。状態の変更は、SUSpended 状態でのみ可能です。正常に終了すると、オブジェクトの状態は ACTive になります。
unset	既存の T_CLIENT オブジェクトを変更します。この組み合わせは、ACTive 状態または SUSpended 状態でのみ可能です。正常終了した場合、オブジェクトの状態は変わりません。
SUSpended	T_CLIENT オブジェクトを中断し、サービス要求 (tpcall() または tpacall())、会話の開始 (tpconnect())、トランザクションの開始 (tpbegin())、および新たな要求のキューへの登録 (tpenqueue()) が実行できないようにします。トランザクション内のクライアントはこれらの呼び出しを実行できますが、現在のトランザクションをアボートまたはコミットすると中断されます。これらのルーチンを呼び出すと、TPESYSTEM エラーが返され、エラーを示すシステム・ログ・メッセージが生成されます。状態の変更は、ACTive 状態でのみ可能です。正常に終了すると、オブジェクトの状態は SUSpended になります。

DEAd	T_CLIENT オブジェクトをアボートの形で非アクティブにします。状態の変更は、ACTive 状態または SUSpended 状態でのみ可能です。クライアントをアボートの形で非アクティブにする方法としては、まず警告メッセージをブロードキャストで送出し (tpbroadcast())、クライアントを中断してから (前述の SET SUSpended を参照)、状態を DEAd に設定することをお勧めします。正常に終了すると、オブジェクトの状態は DEAd になります。 制限事項 : ワークステーション・ハンドラ (T_CLIENT:TA_WSH == Y) は、DEAd 状態には設定できません。 プラットフォームやシグナルの制約により、システムがクライアントを *kill* できない場合があります。この場合、ネイティブ・クライアントは次回の ATMI へのアクセス時にアボートの形で終了し、ワークステーション・クライアントから WSH への接続は直ちに切断されます。

TA_CLIENTID: *string*[1..78]

クライアント識別子。このフィールドのデータは、等号比較の場合を除いて、エンド・ユーザが直接解釈することはできません。

TA_CLTNAME: *string*[0..30]

tpinit() 実行時に TPINIT 構造体の cltname 要素を使用してクライアントに関連付けられるクライアント名。

TA_IDLETIME: 0 <= *num*

このクライアントが、ATMI 呼び出しで最後にシステムと対話してから経過したおおよその時間 (単位は秒)。この値の誤差は、TA_SCANUNIT (T_DOMAIN クラスを参照) の秒数以内です。キー・フィールドとして指定した場合、正の値であればアイドル時間が指定値以上のすべてのクライアントが一致し、負の値であればアイドル時間が指定値以下のすべてのクライアントが一致します。ゼロのときは、すべてのクライアントが一致します。

TA_LMID:*LMID*
: クライアントを実行している論理マシン (ネイティブ・クライアント)、またはクライアントが接続されている論理マシン (ワークステーション・クライアント)。

TA_PID: 1 <= *num*
: クライアントのプロセス識別子。ワークステーション・クライアントの場合、この識別子はクライアントの接続に使用しているワークステーション・ハンドラを示します。GET 操作で負の値を指定すると、呼び出し元のプロセスのクライアント情報を検索できます。呼び出し元のプロセスがクライアントでない場合はエラーが返されます。

TA_CONTEXTID: -2 <= *num* < 30,000
: この特定のアプリケーション関連の識別子。

TA_SRVGRP: *string*[0..30]
: クライアントが関連付けられたサーバ・グループ。この情報は、tpinit() 実行時に TPINIT 構造体の grpname 要素を使用して設定できます。

TA_USRNAME: *string*[0..30]
: tpinit() 実行時に TPINIT 構造体の usrname 要素を使用してクライアントに関連付けられるユーザ名。

TA_WSC: "{Y | N}"
: ワークステーション・クライアント。この属性が "Y" に設定されている場合、指定したクライアントはリモート・ワークステーションからアプリケーションにログインしています。

TA_WSH: "{Y | N}"

: ワークステーション・ハンドラ。この属性が "Y" に設定されている場合、指定したクライアントはワークステーション・ハンドラ・プロセスです。

TA_WSHCLIENTID: *string*[1..78]
: このクライアントがワークステーション・クライアントである場合 (TA_WSH == Y) は、関連付けられたワークステーション・ハンドラ (WSH) のクライアント識別子が返されます。それ以外の場合は、長さゼロの文字列が返されます。

TA_RELEASE: 0 <= *num*
: クライアントを実行しているマシンの BEA Tuxedo システム・プロトコルのメジャー・リリース番号。この番号は、同じマシンの TA_SWRELEASE とは異なる場合があります。ワークステーション・クライアントの場合 (TA_WSC == Y) 、アプリケーションへのアクセスに使用するアプリケーション管理のマシンのメジャー・リリースがこの値とは異なる場合があります。

TA_WSPROTO: 0 <= *num*
: ワークステーション・クライアントの BEA Tuxedo システム・ワークステーション・プロトコルのバージョン番号。この値は、ワークステーション・プロトコルを更新するたびに変更されます。この属性がワークステーション以外のクライアント (TA_WSC == N) に関連付けられている場合は、値としてゼロが返されます。

TA_NUMCONV: 0 <= *num*
: このクライアントが tpconnect() を使用して開始した会話の数。

TA_NUMDEQUEUE: 0 <= *num*
: このクライアントが tpdequeue() を使用してキューからの取り出し操作を開始した回数。

TA_NUMENQUEUE: 0 <= *num*
: このクライアントが tpenqueue() を使用してキューへの登録操作を開始した回数。

TA_NUMPOST: 0 <= *num*
: このクライアントが tppost() を使用して開始したポストの数。

TA_NUMREQ: 0 <= *num*
: このクライアントが tpcall() または tpacall() を使用して開始した要求の数。

TA_NUMSUBSCRIBE: 0 <= *num*
: このクライアントが tpsubscribe() を使用して行ったサブスクリプションの数。

TA_NUMTRAN: 0 <= *num*
: このクライアントが開始したトランザクションの数。

TA_NUMTRANABT: 0 <= *num*
: このクライアントがアボートしたトランザクションの数。

TA_NUMTRANCMT: 0 <= *num*
: このクライアントがコミットしたトランザクションの数。

TA_CMTRET: "{COMPLETE | LOGGED}"
: このクライアントの TP_COMMIT_CONTROL 特性の設定。この特性の詳細については、BEA Tuxedo System ATMI 関数 tpscmt() の説明を参照してください。

TA_CURCONV: 0 <= *num*
: このクライアントが tpconnect() を使用して開始し、現在もアクティブな会話の数。

TA_CURENCRYPTBITS: "{0 | 40 | 56 | 128}"
: このクライアントの現在の暗号化レベル。レベルは、リンクの確立時に調整されます。

  注記　リンク・レベルの暗号化の値 40 は、下位互換性を維持するために提供されています。

TA_CURREQ: 0 <= *num*
: このクライアントが tpcall() または tpacall() を使用して開始し、現在もアクティブな要求の数。

TA_CURTIME: 1 <= *num*
: T_CLIENT:TA_LMID で time(2) システム・コールから返される 1970 年 1 月 1 日の 00:00:00 UTC から現在までの時間 (単位は秒)。この属性は、T_CLIENT:TA_TIMESTART 属性値からの経過時間を算出するために使用できます。

TA_LASTGRP: 1 <= *num* < 30,000
: 最後に開始されたサービス要求またはこのクライアントから開始された会話のサーバ・グループ番号 (T_GROUP:TA_GRPNO)。

TA_NADDR: *string*[1..256] (BEA Tuxedo 8.0 以前は最大で 78 バイト)
: ワークステーション・クライアントである場合、この属性はクライアントのネットワーク・アドレスを示します。ネットワーク・アドレスに表示不能な文字が含まれている場合は、以下のいずれかの形式に変換されます。

  - "0x*hex-digits*"
  - "\\x*hex-digits*"

どちらの形式の文字列でも、文字数が偶数の有効な 16 進数値が含まれている必要があります。このような文字列は、指定された文字列の 16 進数表現を含む文字配列に内部変換されます。

TCP/IP アドレスの場合は、以下のいずれかの形式になります。

- "*//hostname*:*port*"
- "*//#.#.#.#*:*port_number*"

記号は、0 から 255 の範囲の 10 進数を表しています。*port_number* の値は、0 から 65535 の範囲の 10 進数です。

注記　一部のポート番号は、お使いのシステムで使用されるトランスポート・プロトコル (TCP/IP など) のために予約されている場合があります。予約されているポート番号を確認するには、トランスポート・プロトコルのマニュアルを調べてください。

ワークステーション・クライアントでない場合、この属性の値は長さがゼロの文字列になります。

制限事項 : システムがこの情報を提供できるかどうかは、使用するトランスポート・プロバイダによって決まります。プロバイダがこの情報を提供しない場合、ワークステーション・クライアントにアドレスを関連付けることができないこともあります。

TA_NOTIFY: "{DIPIN | SIGNAL | THREAD | IGNORE}"
: このクライアントの通知特性の設定。詳細については、T_DOMAIN クラスのこの属性に関する説明を参照してください。

TA_NUMUNSOL: 0 <= *num*
: このクライアントのキューに登録され、処理待ちになっている任意通知型メッセージの数。

TA_RPID: 1 <= *num*
: クライアントの応答キューに対する UNIX システムのメッセージ・キューの識別子。制限事項 : UNIX システム固有の属性です。アプリケーションを実行しているプラットフォームが UNIX ベースでない場合、この属性は返されないことがあります。

TA_TIMELEFT: 0 <= *num*

このクライアントが現時点で待っている応答がタイムアウトするまでの残り時間 (単位は秒)。タイムアウトは、トランザクション・タイムアウトまたはブロック・タイムアウトです。

TA_TIMESTART: 1 <= *num*

クライアントがアプリケーションに参加した時点までの経過時間 (単位は秒)。T_CLIENT:TA_LMID で time(2) システム・コールから返される 1970 年 1 月 1 日の 00:00:00 UTC から起算されます。

TA_TRANLEV: 0 <= *num*

このクライアントの現在のトランザクション・レベル。値がゼロの場合は、クライアントが現在トランザクションに関与していないことを示します。

制限事項 なし

T_CONN クラスの定義

概要 T_CONN クラスは、アプリケーション内のアクティブな会話の実行時属性を表します。

属性表

表 44TM_MIB(5): T_CONN クラス定義の属性表

属性 (注 1)	タイプ	パーミッション	値	デフォルト値
TA_LMID(k)	string	R--R--R--	*LMID*	N/A
TA_STATE(k)	string	R--R--R--	GET: "ACT"	N/A
			SET:N/A	N/A
TA_SERVICENAME	string	R--R--R--	*string*[1..15]	N/A
TA_CLIENTID(k)	string	R--R--R--	*string*[1..78]	N/A
TA_CONNOGRPNO	long	R--R--R--	1 <= *num* < 30,001	N/A
TA_CONNOLMID	string	R--R--R--	*LMID*	N/A
TA_CONNOPID	long	R--R--R--	1 <= *num*	N/A
TA_CONNOSNDCNT	long	R--R--R--	0 <= *num*	N/A
TA_CONNOSRVID	long	R--R--R--	1 <= *num* < 30,001	N/A
TA_CONNSGRPNO	long	R--R--R--	1 <= *num* < 30,001	N/A
TA_CONNSLMID	string	R--R--R--	*LMID*	N/A
TA_CONNSPID	long	R--R--R--	1 <= *num*	N/A
TA_CONNSSNDCNT	long	R--R--R--	0 <= *num*	N/A
TA_CONNSSRVID	long	R--R--R--	1 <= *num* < 30,001	N/A

(k)—GET キー・フィールド

注 1 T_CONN クラスのすべての属性はローカル属性です。

属性の意味 TA_LMID:*LMID*

検索マシンの論理マシン識別子。

TA_STATE:

GET: "{ACTive}"

GET 操作は、選択した T_CONN オブジェクトの実行時情報を検索します。以下に示す状態は、GET 要求への応答で返される TA_STATE の意味を示します。

ACTive	返されるオブジェクトは、アプリケーション内のアクティブな会話の片方または両方の状態を反映します。

SET:

SET 操作は、このクラスでは使用できません。

TA_SERVICENAME: *string*[1..15]

会話の起動元によって呼び出され、会話の従属側によって処理される会話サービスのサービス名。

TA_CLIENTID: *string*[1..78]

クライアント識別子。このフィールドのデータは、等号比較の場合を除いて、エンド・ユーザが直接解釈することはできません。

TA_CONNOGRPNO: 1 <= *num* < 30,001

会話の起動元のサーバ・グループ番号。起動元がクライアントである場合は、この属性の値として 30,000 が返されます。

TA_CONNOLMID:*LMID*

会話の起動元を実行している場所、または起動元がアプリケーションにアクセスしている場所 (ワークステーション・クライアントの場合) を示す論理マシン識別子。

TA_CONNOPID: 1 <= *num*

会話の起動元のプロセス識別子。

TA_CONNOSNDCNT: 0 <= *num*

起動元が tpsend() を呼び出した回数。

TA_CONNOSRVID: 1 <= *num* < 30,001

会話の起動元のサーバ識別子。

TA_CONNSGRPNO: 1 <= *num* < 30,001

会話の従属側のサーバ・グループ番号。

TA_CONNSLMID:*LMID*
: 会話の従属側を実行している場所、または従属側がアプリケーションにアクセスしている場所 (ワークステーション・クライアントの場合) を示す論理マシン識別子。

TA_CONNSPID: 1 <= *num*
: 会話の従属側のプロセス識別子。

TA_CONNSSNDCNT: 0 <= *num*
: 従属側が tpsend() を呼び出した回数。

TA_CONNSSRVID: 1 <= *num* < 30,001
: 会話の従属側のサーバ識別子。

制限事項 なし

T_DEVICE クラスの定義

概要 T_DEVICE クラスは、BEA Tuxedo システムのデバイス・リストの格納に使用する raw ディスク・スライスまたは UNIX システム・ファイルのコンフィギュレーション属性と実行時属性を表します。このクラスを使用すると、raw ディスク・スライスまたは UNIX システム・ファイルに格納するデバイス・リストのエントリを作成および削除できます。

属性表

表 45TM_MIB(5): T_DEVICE クラス定義の属性表

属性 (注 1)	タイプ	パーミッション	値	デフォルト値
TA_LMID(*)	string	ru-r--r--	*LMID*	"local_lmid"
TA_CFGDEVICE(r)(*)	string	ru-r--r--	*string*[2..64]	N/A
TA_DEVICE(*)	string	ru-r--r--	*string*[2..64]	"TA_CFGDEVICE"
TA_DEVOFFSET(*)	long	ru-r--r--	0 <= *num*	0
TA_DEVSIZE(r)	long	rw-r--r--	0 <= *num*	1000 (注 3)
TA_DEVINDEX(*)(注 2)	long	r--r--r--	0 <= *num*	N/A
TA_STATE(k)	string	rwxr--r--	GET: "VAL" SET: "{NEW | INV}"	N/A N/A

(k)—GET キー・フィールド
(r)— オブジェクトの作成に必要なフィールド (SET TA_STATE NEW)
(*)—GET/SET キー、SET 操作では 1 つ以上必要

注 1 T_DEVICE クラスのすべての属性はローカル属性です。
注 2 SET 操作では、デバイス・リストの特定のエントリを識別するために TA_DEVINDEX が必要となります。ただし、新たなデバイス・リスト・エントリを作成する目的で状態を NEW にセットする場合は例外です。この場合、TA_DEVINDEX は設定しないでください。値は、システムによって割り当てられ、エントリが正常に作成された時点で返されます。

注 3 TA_DEVSIZE は、オブジェクトの作成時にのみ SET されます。

属性の意味

TA_LMID:*LMID*

デバイスが存在する論理マシンの識別子。この属性は、コンフィギュレーションが済んでいる (少なくとも 1 つの T_MACHINE エントリが定義されている) アプリケーションでは、そのアプリケーションがブートされていてもブートされていなくてもキー・フィールドとして使用できます。SET 操作では、ブートされたアプリケーションにアクセスする際にこの属性がキー・フィールドとして必要になります。コンフィギュレーションの済んでいないアプリケーションで T_DEVICE クラスにアクセスする場合、この属性は指定されていても無視されます。

TA_CFGDEVICE: *string*[2..64]

BEA Tuxedo のファイル・システムが格納されている、または格納するためのファイルやデバイスの絶対パス名。

TA_DEVICE: *string*[2..64]

デバイス・リスト・エントリの絶対パス名。

TA_DEVOFFSET: 0 <= *num*

TA_CFGDEVICE で指定した BEA Tuxedo システム VTOC で使用する TA_DEVICE の領域の開始点を示すオフセット (単位はブロック)。制限事項 : BEA Tuxedo ファイル・システム (TA_CFGDEVICE) 上の最初のデバイス・リスト・エントリ (TA_DEVICE) では、この属性をゼロに設定する必要があります。

TA_DEVSIZE: 0 <= *num*

デバイス・リスト・エントリとして使用するディスク領域のサイズ (単位はページ)。制限事項 : この属性は、状態を NEW に変更する場合にのみ設定できます。

TA_DEVINDEX: 0 <= *num*

TA_CFGDEVICE が指すデバイス・リスト内の TA_DEVICE のデバイス・インデックス。この属性の値は、BEA Tuxedo ファイル・システムの特定のデバイスに関係のある属性を取得または設定する場合に限り、識別子として使用できます。

TA_STATE:

GET: "{VALid}"

GET 操作は、選択した T_DEVICE オブジェクトの実行時情報を検索します。以下に示す状態は、GET 要求への応答で返される TA_STATE の意味を示します。

VALid	TA_CFGDEVICE が示す BEA Tuxedo ファイル・システムが存在し、有効なデバイス・リストが格納されていることを示します。TA_DEVICE は、デバイス・インデックス telnet lchome3 を持つファイル・システム内の有効なデバイスです。

SET: "{NEW | INValid}"

SET 操作は、選択した T_DEVICE オブジェクトの情報の更新、または指定したオブジェクトの追加を実行します。以下に示す状態は、SET 要求で設定される TA_STATE の意味を示します。以下に示されていない状態は設定できません。

NEW	アプリケーションの T_DEVICE オブジェクトを作成または再初期化します。状態の変更は、INValid 状態または VALid 状態でのみ可能です。正常終了すると、オブジェクトの状態は VALid になります。INValid 状態でこの状態遷移が呼び出された場合はオブジェクトが作成されます。それ以外の場合はオブジェクトが再初期化されます。TA_CFGDEVICE BEA Tuxedo ファイル・システムで最初の TA_DEVICE デバイス・リスト・エントリを作成すると、TA_CFGDEVICE で必要となる VTOC 構造体と UDL 構造体が自動的に作成されて初期化されます。特定の TA_CFGDEVICE 用に作成した最初のデバイス・リスト・エントリの値は、TA_DEVICE 属性の値と等しくなければなりません。
INValid	アプリケーションの T_DEVICE オブジェクトを削除します。状態の変更は VALid 状態でのみ可能です。正常終了すると、オブジェクトの状態は INValid になります。TA_DEVINDEX 0 は特別で、最後に削除する必要があります。

制限事項　なし

T_DOMAIN クラスの定義

概要 T_DOMAIN クラスは、グローバルなアプリケーションの属性を表します。これらの属性値は、BEA Tuxedo システム・アプリケーションの識別、カスタマイズ、サイズの指定、セキュリティ保護に使用します。以下で示す属性値の多くは、この MIB で示すほかのクラスでアプリケーションのデフォルト値として使用します。

個々のアプリケーションには、T_DOMAIN クラスのオブジェクトが 1 つだけ存在します。したがって、このクラスにはキー・フィールドは定義されません。このクラスに対する GET 操作は、常にこの唯一のオブジェクトに関する情報を返します。同様に、SET 操作は常にその唯一のオブジェクトを更新します。GETNEXT は、このクラスでは使用できません。

属性表

表 46TM_MIB(5): T_DOMAIN **クラス定義の属性表**

属性	タイプ	パーミッション	値	デフォルト値
TA_IPCKEY(r)	long	rw-r--r--	32,769 <= *num* < 262,144	N/A
TA_MASTER(r)	string	rwxr-xr--	"*LMID1*[,*LMID2*]"	N/A
TA_MODEL(r)	string	rw-r--r--	"{SHM | MP}"	N/A
TA_STATE	string	rwxr--r--	GET: "{ACT | INA}"	N/A
			SET: "{NEW | INV | ACT | INA | FIN}"	N/A
TA_DOMAINID	string	rwxr--r--	*string*[0..30]	""
TA_PREFERENCES	string	rwxr--r--	*string*[0..1023]	""
TA_UID	long	rwyr--r--	0 <= *num*	(注 1)
TA_GID	long	rwyr--r--	0 <= *num*	(注 1)
TA_PERM	long	rwyr--r--	0001 <= *num* <= 0777	0666
TA_LICEXPIRE	long	R--R--R--	*string*[0..78]	N/A

表 46TM_MIB(5): T_DOMAIN **クラス定義の属性表 (続き)**

属性	タイプ	パーミッション	値	デフォルト値
TA_LICMAXUSERS	long	R--R--R--	0 <= *num* < 32,768	N/A
TA_LICSERIAL	string	R--R--R--	*string*[0..78]	N/A
TA_MIBMASK	long	rwx------	0 <= *num* <= 0777	0000
TA_MAXACCESSERS	long	rwyr--r--	1 <= *num* < 32,768	50
TA_MAXCONV	long	rwyr--r--	0 <= *num* < 32,768	64
TA_MAXGTT	long	rwyr--r--	0 <= *num* < 32,768	100
TA_MAXBUFSTYPE	long	rw-r--r--	1 <= *num* < 32,768	32
TA_MAXBUFTYPE	long	rw-r--r--	1 <= *num* < 32,768	16
TA_MAXDRT	long	rw-r--r--	0 <= *num* < 32,768	0
TA_MAXGROUPS	long	rw-r--r--	100 <= *num* < 32,766	100
TA_MAXNETGROUPS	long	rw-r--r--	1 <= *num* < 8,192	8
TA_MAXMACHINES	long	rw-r--r--	256 <= *num* < 8,191	256
TA_MAXQUEUES	long	rw-r--r--	1 <= *num* < 8,192	50
TA_MAXRFT	long	rw-r--r--	0 <= *num* < 32,766	0
TA_MAXRTDATA	long	rw-r--r--	0 <= *num* < 32,761	0
TA_MAXSPDATA	long	rw-r--r--	1 <= *num* <= 2147483640	0
TA_MAXTRANTIME	long	rwyr--r--	1 <= *num* <= 2147483647	0
TA_MAXSERVERS	long	rw-r--r--	1 <= *num* < 8,192	50
TA_MAXSERVICES	long	rw-r--r--	1 <= *num* < 32,766	100
TA_MAXACLGROUPS	long	rw-r--r--	1 <= *num* < 16,384	16,384
TA_CMTRET	string	rwyr--r--	"{COMPLETE | LOGGED}"	"COMPLETE"
TA_LDBAL	string	rwyr--r--	"{Y | N}"	"Y"
TA_NOTIFY	string	rwyr--r--	"{DIPIN | SIGNAL | THREAD | IGNORE}"	"DIPIN"

表 46TM_MIB(5): T_DOMAIN **クラス定義の属性表 (続き)**

属性	タイプ	パーミッション	値	デフォルト値
TA_SYSTEM_ACCESS	string	rwyr--r--	"{FASTPATH | PROTECTED} [,NO_OVERRIDE]"	"FASTPATH"
TA_OPTIONS	string	rwyr--r--	"{[LAN | MIGRATE | ACCSTATS | NO_XA | NO_AA],*}"	""
TA_USIGNAL	string	rw-r--r--	"{SIGUSR1 | SIGUSR2}"	"SIGUSR2"
TA_SECURITY	string	rw-r--r--	"{NONE | APP_PW | USER_AUTH | ACL | MANDATORY_ACL}"	"NONE"
TA_PASSWORD	string	-wx------	*string*[0..30]	N/A
TA_AUTHSVC	string	rwxr--r--	*string*[0..15]	""
TA_SCANUNIT	long	rwxr-xr--	0 <= *num* <= 60	10(注 2)
TA_BBLQUERY	long	rwxr-xr--	0 <= *num* <32,768	300(注 3)
TA_BLOCKTIME	long	rwxr-xr--	0 <= *num* < 32,768	60(注 3)
TA_DBBLWAIT	long	rwxr-xr--	0 <= *num* < 32,768	20(注 3)
TA_SANITYSCAN	long	rwxr-xr--	0 <= *num* < 32,768	120(注 3)

表 46TM_MIB(5): T_DOMAIN **クラス定義の属性表 (続き)**

属性	タイプ	パーミッション	値	デフォルト値
TA_CURDRT	long	r--r--r--	0 <= *num* < 32,768	N/A
TA_CURGROUPS	long	r--r--r--	0 <= *num* < 32,768	N/A
TA_CURMACHINES	long	r--r--r--	0 <= *num* < 32,768	N/A
TA_CURQUEUES	long	r--r--r--	0 <= *num* < 32,768	N/A
TA_CURRFT	long	r--r--r--	0 <= *num* < 32,768	N/A
TA_CURRTDATA	long	r--r--r--	0 <= *num* < 32,768	N/A
TA_CURSERVERS	long	r--r--r--	0 <= *num* < 32,768	N/A
TA_CURSERVICES	long	r--r--r--	0 <= *num* < 32,768	N/A
TA_CURSTYPE	long	r--r--r--	0 <= *num* < 32,768	N/A
TA_CURTYPE	long	r--r--r--	0 <= *num* < 32,768	N/A
TA_HWDRT	long	r--r--r--	0 <= *num* < 32,768	N/A
TA_HWGROUPS	long	r--r--r--	0 <= *num* < 32,768	N/A
TA_HWMACHINES	long	r--r--r--	0 <= *num* < 32,768	N/A
TA_HWQUEUES	long	r--r--r--	0 <= *num* < 32,768	N/A
TA_HWRFT	long	r--r--r--	0 <= *num* < 32,768	N/A
TA_HWRTDATA	long	r--r--r--	0 <= *num* < 32,768	N/A
TA_HWSERVERS	long	r--r--r--	0 <= *num* < 32,768	N/A
TA_HWSERVICES	long	r--r--r--	0 <= *num* < 32,768	N/A
TA_SEC_PRINCIPAL_NAME	string	rwxr--r--	*string*[0..511]	""
TA_SEC_PRINCIPAL_LOCATION	string	rwxr--r--	*string*[0..511]	""
TA_SEC_PRINCIPAL_PASSVAR	string	rwxr--r--	*string*[0..511]	""
TA_SIGNATURE_AHEAD	long	rwxr--r--	1 <= *num* <= 2147483647	3600
TA_SIGNATURE_BEHIND	long	rwxr--r--	1 <= *num* <= 2147483647	604800
TA_SIGNATURE_REQUIRED	string	rwxr--r--	"{Y | N}"	"N"
TA_ENCRYPTION_REQUIRED	string	rwxr--r--	"{Y | N}"	"N"

表 46TM_MIB(5): T_DOMAIN **クラス定義の属性表 (続き)**

属性	タイプ	パーミッション	値	デフォルト値
(r)― オブジェクトの作成に必要なフィールド (SET TA_STATE NEW)				

注 1 UNIX システムが認識できる UID と GID
注 2 *num* は 5 の倍数であること
注 3 *num* は、*num* と TA_SCANUNIT の積がほぼデフォルト値と同じになるように設定します。

属性の意味

TA_IPCKEY: 32,769 <= *num* < 262,144

BEA Tuxedo システムの掲示板の既知のアドレスに対する数値キー。単一プロセッサ環境では、このキーにより掲示板の名前が指定されます。マルチプロセッサ環境または LAN 環境では、このキーにより DBBL のメッセージ・キューが指定されます。また、このキーは、既知のアドレスのほか、アプリケーション全体の掲示板などのリソース名の基準としても使用されます。

TA_MASTER: "*LMID1*[,*LMID2*]"

マスタ論理マシン (*LMID1*) とバックアップ論理マシン (*LMID2*) の識別子。INActive 状態のアプリケーションでは、マスタの識別子 (*LMID1*) がローカル・マシンと一致する必要があります。SHM モードのアプリケーション (後述の TA_MODEL を参照) では、マスタ論理マシンの識別子のみ設定できます。ACTive MP 状態のアプリケーション (後述の TA_MODEL を参照) では、この属性値の変更は次のような意味を持ちます。

現在アクティブなマスタ LMID を A、現在のバックアップ・マスタ LMID を B、セカンダリ LMID をそれぞれ C、D、. . . とすると、MP モードで実行中のアプリケーションの TA_MASTER 属性で可能な変更は、次のようなシナリオによって定義されます。

A,B -> B,A - マスタを A から B に移行。

A,B -> A,C - バックアップ・マスタ LMID を C に変更。

マスタの移行には、順序立てた方式と分断方式があります。順序立てた移行は、マスタ・マシンが ACTive 状態でアクセス可能な場合に行われます。それ以外の場合には分断方式で移行します。ネットワーク

への接続を新たに確立する場合や確立しなおす際は、接続する 2 つのサイトがマスタ・マシンの場所に関する情報を共有しているかどうかが確認されます。この点が確認できない場合、接続は拒否され、適切なログ・メッセージが生成されます。ACTive 状態のアプリケーションのマスタ・マシンとバックアップ・マシンの BEA Tuxedo リリース番号は、アプリケーション内のほかのすべてのマシンのリリース番号と同じかそれ以上である必要があります。また、マスタ・マシンとバックアップ・マシンのリリース番号は同じでなければなりません。TA_MASTER 属性を変更する際は、この関係を維持する必要があります。

TA_MODEL: "{SHM | MP}"

コンフィギュレーションの種類。SHM は単一のマシンのコンフィギュレーションを設定します。指定できるのは、T_MACHINE オブジェクトのみです。MP は、複数のマシンを使用する環境、すなわちネットワーク・コンフィギュレーションを設定します。ネットワーク化されたアプリケーションを定義する場合は、MP を指定する必要があります。

TA_STATE:

GET: "{ACTive | INActive}"

GET 操作は、T_DOMAIN オブジェクトのコンフィギュレーション情報および実行時情報を検索します。以下に示す状態は、GET 要求への応答で返される TA_STATE の意味を示します。

ACTive	T_DOMAIN オブジェクトは定義済みで、マスタ・マシンがアクティブであることを示します。
INActive	T_DOMAIN オブジェクトは定義済みで、アプリケーションがアクティブでないことを示します。

SET: "{NEW | INValid | ACTive | INActive | FINactive}"

SET 操作は、T_DOMAIN オブジェクトのコンフィギュレーション情報および実行時情報を更新します。以下に示す状態は、SET 要求で設定される TA_STATE の意味を示します。以下に示されていない状態は設定できません。

NEW	アプリケーションの T_DOMAIN オブジェクトを作成します。状態の変更は INValid 状態でのみ可能です。正常に終了すると、オブジェクトの状態は INActive になります。この状態の変更により、TA_MASTER から推定された TA_LMID、ローカル・システム名に基づく TA_PMID、および環境変数 TUXCONFIG と TUXDIR に基づいて決定された TA_TUXCONFIG と TA_TUXDIR により、新しい T_MACHINE オブジェクトも作成されます。T_MACHINE クラスのほかのコンフィギュレーション可能な属性は、T_DOMAIN NEW 要求に値を設定することにより、この時点で設定できます。TA_APPDIR の値を指定しない場合は、カレント・ディレクトリがデフォルト・ディレクトリとなります。
unset	T_DOMAIN オブジェクトを変更します。変更は、ACTive 状態または INActive 状態でのみ可能です。正常終了した場合、オブジェクトの状態は変わりません。
INValid	アプリケーションの T_DOMAIN オブジェクトを削除します。状態の変更は、INActive 状態でのみ可能です。正常終了すると、オブジェクトの状態は INValid になります。
ACTive	マスタ・マシンの管理プロセス (DBBL、BBL など) をアクティブにします。この状態遷移に対するパーミッションの決定に際しては、アクティブなオブジェクトのパーミッション (--x--x--x) が考慮されます。状態の変更は、INActive 状態でのみ可能です。正常に終了すると、オブジェクトの状態は INActive になります。
INActive	マスタ・マシンの管理プロセス (DBBL、BBL など) を非アクティブにします。状態の変更は、ACTive 状態でのみ可能です。正常に終了すると、オブジェクトの状態は INActive になります。

FINactive	マスタ・マシンの管理プロセス (DBBL、BBL など) を強制的に非アクティブにします。シャットダウンを許可するかどうかの決定においては、アタッチされているクライアントは無視されます。状態の変更は、ACTive 状態でのみ可能です。正常に終了すると、オブジェクトの状態は INActive になります。

TA_DOMAINID: *string*[0..30]
ドメインの識別文字列。

TA_PREFERENCES: *string*[0..1023]
アプリケーションが定義するフィールド。このフィールドは、BEA Tuxedo システム /Admin GUI 製品で GUI 表示の環境設定を格納、保管するために使用します。

TA_UID: 0 <= *num*
T_MACHINE クラスで新たにコンフィギュレーションするオブジェクトの属性のデフォルト設定。制限事項 : この属性を変更しても、コンフィギュレーション済みの T_MACHINE オブジェクトおよびアクティブなオブジェクトには影響しません。

TA_GID: 0 <= *num*
T_MACHINE クラスで新たにコンフィギュレーションするオブジェクトの属性のデフォルト設定。制限事項 : この属性を変更しても、コンフィギュレーション済みの T_MACHINE オブジェクトおよびアクティブなオブジェクトには影響しません。

TA_PERM: 0001 <= *num* <= 0777
T_MACHINE クラスで新たにコンフィギュレーションするオブジェクトの属性のデフォルト設定。制限事項 : この属性を変更しても、コンフィギュレーション済みの T_MACHINE オブジェクトおよびアクティブなオブジェクトには影響しません。

TA_LICEXPIRE: *string*[0..78]
マシンのバイナリの失効期日。バイナリが BEA Tuxedo システム・マスタ・バイナリでない場合は長さゼロの文字列になります。

TA_LICMAXUSERS: 0 <= *num* < 32,768
: マシンにライセンスされたユーザの最大数。バイナリが BEA Tuxedo システム・マスタ・バイナリでない場合は -1 になります。

TA_LICSERIAL: *string* [0..78]
: マシンのシリアル番号。

TA_MIBMASK: 0 <= *num* <= 0777
: 属性のアクセス・マスク。この属性値に指定したユーザ・タイプとアクセス・モードの組み合わせは、このマニュアルで定義されているクラスと属性のすべての組み合わせに使用できるわけではありません。たとえば、0003 と設定した場合は、管理者およびオペレータ以外のユーザに対する更新が禁止されます。

TA_MAXACCESSERS: 1 <= *num* < 32,768
: このアプリケーション内の特定のマシン上の掲示板に、同時にアクセスできるクライアントおよびサーバの最大数のデフォルト値。指定しない場合のデフォルトの最大数は 50 です。この属性の T_DOMAIN の値は、マシンごとに T_MACHINE クラスでオーバライドできます。

  この数には、BBL、restartsrv、cleanupsrv、tmshutdown()、tmadmin() などのシステム管理プロセスを含める必要はありません。ただし、DBBL、すべてのブリッジ・プロセス、すべてのシステム提供サーバ・プロセスとアプリケーション・サーバ・プロセス、および特定のサイトで使用する可能性があるクライアント・プロセスはこの数に含めますシステム提供のサーバには、AUTHSVR、TMQUEUE、TMQFORWARD、TMUSREVT、TMSYSEVT、TMS (T_GROUP:TA_TMSNAME 属性を参照)、TMS_QM、GWTDOMAIN、WSL などがあります。アプリケーションが特定のサイトでワークステーション・リスナ (WSL) を起動する場合は、起動する WSL と使用する可能性があるワークステーション・ハンドラ (WSH) の両方をこの数に含める必要があります。

  BEA Tuxedo リリース 7.1 より前 (6.5 以前) では、ユーザ・ライセンス数をチェックする仕組みにおいて、アプリケーションの TA_MAXACCESSERS 属性と TA_MAXSERVERS 属性が使用されていました。つまり、アプリケーションで実行中の 1 台以上のマシンの TA_MAXACCESSERS の数と、特定のマシンの TA_MAXACCESSERS の数の合計が、TA_MAXSERVERS の数とユーザ・ライセンス数の合計より大きい場合、マシンを起動することはできませんでした。したがって、アプリケーションの TA_MAXACCESSERS パラメータには、TA_MAXSERVERS の

数とユーザ・ライセンス数の合計か、またはそれより小さい値を指定しなければなりませんでした。

BEA Tuxedo のリリース 7.1 以降では、アプリケーションに設定されているユーザ・ライセンスの数と、現在使用されているユーザ・ライセンスの数に基づいて、ライセンスのチェックが行われます。すべてのユーザ・ライセンスが使用中になると、アプリケーションに新しいクライアントが参加することはできなくなります。

制限事項 : この属性を変更しても、コンフィギュレーション済みの T_MACHINE オブジェクトおよびアクティブなオブジェクトには影響しません。

TA_MAXCONV: 0 <= *num* < 32,768

このアプリケーションの特定のマシン上のクライアントとサーバが同時に関与できる会話の最大数。この値を指定しない場合のデフォルト値は、T_SERVER クラスに会話サーバが定義されている場合は 64、そうでない場合は 1 になります。1 つのサーバで、最大 64 個の会話を同時に行うことができます。この属性の T_DOMAIN の値は、マシンごとに T_MACHINE クラスでオーバライドできます。

制限事項 : この属性を変更しても、コンフィギュレーション済みの T_MACHINE オブジェクトおよびアクティブなオブジェクトには影響しません。

TA_MAXGTT: 0 <= *num* < 32,768

このアプリケーションの特定のマシンが同時に関与できるグローバル・トランザクションの最大数。指定しない場合のデフォルト値は 100 です。この属性の T_DOMAIN の値は、マシンごとに T_MACHINE クラスでオーバライドできます。

制限事項 : この属性を変更しても、コンフィギュレーション済みの T_MACHINE オブジェクトおよびアクティブなオブジェクトには影響しません。

TA_MAXBUFSTYPE: 1 <= *num* < 32,768

掲示板のバッファ・サブタイプ・テーブルに対応するバッファ・サブタイプの最大数。

TA_MAXBUFTYPE: 1 <= *num* < 32,768

掲示板のバッファ・タイプ・テーブルに対応するバッファ・タイプの最大数。

TA_MAXDRT: 0 <= *num* < 32,768

掲示板のルーティング・テーブルに対応するルーティング・テーブル・エントリの最大数。T_ROUTING クラス・オブジェクトごとに、エントリが 1 つずつ必要です。実行時にテーブルを拡張できるようにするには、追加のエントリを割り当てる必要があります。

TA_MAXGROUPS: 100 <= *num* < 32,766

掲示板のサーバ・グループ・テーブルに対応するサーバ・グループの最大数。制限事項 : BEA Tuxedo リリース 4.2.2 以前のサイトでは、この属性の値が 100 に固定されています。こうしたサイトとの相互運用性を確保するには、使用中のサーバ・グループ・エントリの数が常に 100 以下になるようにする必要があります。リリース 4.2.2 以前のサイトは、定義されているサーバ・グループの数が 100 を超えるアプリケーションには結合できません。また、リリース 4.2.2 以前のサイトがすでに含まれているアプリケーションでは、100 を超えるサーバ・グループを追加することはできません。

TA_MAXNETGROUPS: 1 <= *num* < 8,192

TUXCONFIG ファイルの NETWORK セクションに対応するコンフィギュレーション済みのネットワーク・グループの最大数を指定します。この値には、1 以上 8192 未満を指定します。指定しないは、デフォルト値の 8 が設定されます。

TA_MAXMACHINES: 256 <= *num* < 8,191

掲示板のマシン・テーブルに対応するマシンの最大数。制限事項 : BEA Tuxedo リリース 4.2.2 では、この属性の値が 256 に固定されています。4.2.2 より前のリリースでは、この属性の値が 50 に固定されています。リリース 4.2.2 以前のサイトとの相互運用性を確保するには、使用しているマシン・エントリの数が、最も低い固定値を超えないようにする必要があります。リリース 4.2.2 のサイトは、定義されているマシンの数が 256 を超えるアプリケーションには結合できません。4.2.2 より前のリリースのサイトは、定義されているマシンの数が 50 を超えるアプリケーションには結合できません。また、リリース 4.2.2 以前のサイトがすでに含まれているアプリケーションでは、最も低い制限値を超えるマシンを追加することはできません。

TA_MAXQUEUES: 1 <= *num* < 8,192

掲示板のキュー・テーブルに対応するキューの最大数。制限事項 : リリース 4.2.2 以前のサイトは、TA_MAXQUEUES の設定が TA_MAXSERVERS

の設定に等しい場合に限り、アクティブなアプリケーションに結合することができます。

TA_MAXRFT: 0 <= *num* < 32,768
: 掲示板の範囲基準テーブルに対応するルーティング基準範囲テーブル・エントリの最大数。TA_RANGES の設定に含まれる範囲ごとに 1 つのエントリが必要です。これに加え、T_ROUTING クラス・オブジェクトごとに 1 つの追加エントリが必要です。実行時にテーブルを拡張できるようにするには、追加のエントリを割り当てる必要があります。

TA_MAXRTDATA: 0 <= *num* < 32,761
: 掲示板の文字列プール・テーブルに対応する文字列プール領域の最大数。文字列プールには、TA_RANGES の値で指定した文字列と CARRAY が格納されます。実行時にテーブルを拡張できるようにするには、追加の領域を割り当てる必要があります。

TA_MAXSPDATA 0 <= *num* <= 2147483640
: 掲示板の共通文字列プールに対応する文字列プール領域の最大数。この値には、0 以上 2147483640 以下を指定します。指定しない場合は、デフォルト値の 0 が設定されます。この属性は、BEA Tuxedo 8.1 以降が動作するアプリケーションにのみ適用されます。

  ほとんどの場合、この属性をデフォルト値に設定しておけば、BEA Tuxedo システムによって TUXCONFIG パラメータ文字列 (TUXCONFIG、TUXDIR、APPDIR、TLOGDEVICE、ULOGPFX、ENVFILE、TMSNAME、RCMD、NADDR、NLSADDR、FADDR、および SERVERS セクションの AOUT) に必要な文字列プール領域が割り当てられます。BEA Tuxedo 8.1 では、これらのパラメータ文字列の最大長が 256 バイトまで拡張されています。

  大規模な動的コンフィギュレーション (たとえば、BEA Tuxedo アプリケーションにさらに 6 つのマシンを追加するなど) が予想されるアプリケーションの場合、管理者は TA_MAXSPDATA 属性を使用して共通文字列プールのサイズを増やすことができます。共通文字列プールのサイズを調整しても、TA_MAXRTDATA 属性で制御するルーティング文字列プールのサイズには影響しません。これら 2 つの文字列プールは別々に制御されます。

  TA_MAXSPDATA にどのような値を指定しても、BEA Tuxedo システムが計算した範囲外の文字列プール領域への割り当ては行われません。この範囲の計算は、(1) TUXCONFIG ファイルに実際に指定した文字列、お

よび (2) 指定された文字列が最大長の 256 バイトである場合に必要となる領域の長さ、の 2 つに基づいて行われます。ユーザが指定した値がこの範囲の外である場合、tmloadcf(1) コマンドによって警告が通知され、それに最も近い許容値が設定されます。

TUXCONFIG パラメータの最大長が 256 バイトに拡張されたため、実際に掲示板に格納されるのは、GROUPS セクションの TMSNAME パラメータ、および SERVERS セクションの AOUT および RCMD パラメータのみになっています。その他のパラメータは、プロセスの起動時に読み込まれ、プロセス・メモリに格納されます。

TA_MAXTRANTIME 0 <= *num* <= 2147483647

この BEA Tuxedo アプリケーションが開始または受信するトランザクションのタイムアウトの最大値 (単位は秒)。この値には、0 以上 2147483647 以下の値を指定します。デフォルト値は 0 で、グローバル・トランザクションのタイムアウトを設定しないことを示します。この属性は、BEA Tuxedo 8.1 以降が動作するアプリケーションにのみ適用されます。

TA_MAXTRANTIME タイムアウト値が、AUTOTRAN サービスに対して指定した TRANTIME タイムアウト値またはトランザクションを開始する際に tpbegin(3c) 呼び出しで渡されたタイムアウト値よりも小さい場合、トランザクションのタイムアウトは TA_MAXTRANTIME 値まで減少します。TA_MAXTRANTIME は BEA Tuxedo 8.0 以前を実行するマシン上で開始されるトランザクションには影響を与えません。ただし、BEA Tuxedo 8.1 以降が動作するマシンがトランザクションの影響を受ける場合は、そのマシンに対して設定されている TA_MAXTRANTIME 値までトランザクション・タイムアウト値が制限 (必要に応じて減少) されます。

UBBCONFIG ファイルの SERVICES セクションで指定した TRANTIME 値が TA_MAXTRANTIME の値より大きい場合は、tmloadcf(1) コマンドによってコンフィギュレーションがエラーなしでロードされます。BEA Tuxedo 8.1 以降のマシンが AUTOTRAN トランザクションの影響を受ける場合は、そのマシンに対して設定されている TA_MAXTRANTIME 値までトランザクション・タイムアウト値が自動的に減少されます。

制限事項 : この属性を実行時に変更しても、それ以前に開始されたトランザクションには反映されません。

TA_MAXSERVERS: 1 <= *num* < 8,192

このアプリケーションの掲示板のサーバ・テーブルに対応するサーバの最大数。指定しない場合のデフォルト値は 50 です。

アプリケーションで使用可能なシステム提供のサーバおよびアプリケーション・サーバのすべてのインスタンスを、掲示板のサーバ・テーブルに指定する必要があります。このテーブルはグローバル・テーブルであるため、同じサーバ・テーブルがアプリケーションの各マシン上に存在します。システム提供のサーバには、AUTHSVR、TMQUEUE、TMQFORWARD、TMUSREVT、TMSYSEVT、TMS (T_GROUP:TA_TMSNAME 属性を参照)、TMS_QM、GWTDOMAIN、WSL などがあります。

BEA Tuxedo システムを使用するサイトを管理するには、1 サイトあたりほぼ 1 つのシステム提供サーバが必要です。さらに、DBBL プロセス、BBL プロセス、ブリッジ・プロセス、および WSH プロセスもすべて TA_MAXSERVERS の数に含めます。

TA_MAXSERVICES: 1 <= *num* < 32,766

掲示板のサービス・テーブルに対応するサービスの最大数。この値には、0 より大きく 32,766 未満の値を指定します。指定しない場合は、デフォルト値の 100 が設定されます。

この属性を設定する場合、アプリケーション・サーバおよびシステム・サーバで使用するサービスの数を計算に入れる必要があります。考慮すべきサーバとしては、BBL、DBBL、BRIDGE、TMS、システム提供の管理サーバなどがあります。BEA Tuxedo システムのサイトを管理するには、サイト 1 つにつき 5 つ程度のサービスが必要になります。また、ワークステーション、/Q、ドメインといった管理コンポーネントをサポートする管理サービスも計算に入れる必要があります。

TA_MAXACLGROUPS: 1 <= *num* < 16, 384

ACL パーミッションのチェックに使用できるグループ識別子の最大数。定義可能なグループ ID の最大数は、TA_MAXACLGROUPS - 1 です。

TA_CMTRET: "{COMPLETE | LOGGED}"

BEA Tuxedo システム・アプリケーションのすべてのクライアント・プロセスおよびサーバ・プロセスの TP_COMMIT_CONTROL 特性の初期設定。LOGGED の場合、TP_COMMIT_CONTROL 特性は TP_CMT_LOGGED に初期

設定され、それ以外の場合は TP_CMT_COMPLETE に初期設定されます。この特性の設定の詳細については、BEA Tuxedo System ATMI 関数 tpscmt() の説明を参照してください。

制限事項 : この属性を実行時に変更しても、アクティブなクライアントやサーバには反映されません。

TA_LDBAL: "{Y | N}"

ロード・バランシング機能のオン (「Y」) / オフ (「N」) を切り替えます。

制限事項 : この属性を実行時に変更しても、アクティブなクライアントやサーバには反映されません。

TA_NOTIFY: "{DIPIN | SIGNAL | THREAD | IGNORE}"

クライアント・プロセスに対して送出される任意通知型メッセージに使用する通知検出方式のデフォルト設定。このデフォルト設定は、適切な tpinit() フラグ値を使用してクライアントごとにオーバライドできます。いったん検出された任意通知型メッセージをアプリケーションで使用するには、tpsetunsol() 関数で指定したアプリケーション定義の任意通知型メッセージ処理ルーチンを使用します。

DIPIN は、ディップ・イン方式で通知検出を行うことを示します。この方式では、クライアント・プロセスに代わってシステムが ATMI 呼び出しの間に通知メッセージのみを検出します。特定の ATMI 呼び出しでの検出ポイントは、システムによって定義されるものではありません。したがって、システムによるブロッキング呼び出しがディップ・イン検出によって割り込みされることはありません。DIPIN は、デフォルトの通知検出方式です。

SIGNAL は、シグナル・ベース方式で通知検出を行うことを示します。この方式では、通知メッセージが使用可能になると、システムがターゲットのクライアント・プロセスにシグナルを送出します。システムは、通知方式を選択したクライアントに代わって、シグナル検出ルーチンをインストールします。

THREAD は、THREAD 方式で通知を行うことを示します。この方式では、任意通知型メッセージを受け取るための専用のスレッドが作成され、そのスレッドに任意通知型メッセージ・ハンドラがディスパッチされます。1 つの BEA Tuxedo アプリケーションで同時に実行できる任意通知型メッセージ・ハンドラは 1 つのみです。この値は、マルチス

レッドをサポートするプラットフォーム専用です。COBOL クライアントでは、THREAD 通知は使用できません。THREAD を指定した場合は、デフォルトで DIPIN に設定されます。

IGNORE は、アプリケーション・クライアントがデフォルトで通知メッセージを無視することを示します。この設定は、tpinit() 時の通知を要求するクライアントのみが任意通知型メッセージを受信するアプリケーションに適しています。

制限事項 : この属性を実行時に変更しても、アクティブなクライアントには反映されません。ネイティブ・クライアント・プロセスのすべてのシグナルは、アプリケーション・プロセスではなく管理システム・プロセスによって処理されます。したがって、SIGNAL 方式を使用して通知できるのは、アプリケーション管理者と同じ UNIX システム・ユーザ識別子で実行されているネイティブ・クライアントのみです。ワークステーション・クライアントの場合は、どのユーザ識別子で実行されているかに関係なく、SIGNAL 方式を使用できます。

注記　SIGNAL 通知方式は、MS-DOS クライアントでは使用できません。

TA_SYSTEM_ACCESS: {FASTPATH | PROTECTED}[,NO_OVERRIDE]

BEA Tuxedo システム・ライブラリが、アプリケーションのプロセス内で BEA Tuxedo システムの内部テーブルにアクセスするために使用するデフォルト・モード。FASTPATH は、BEA Tuxedo システム・ライブラリが、高速アクセス用の保護されていない共有メモリを使用して BEA Tuxedo システムの内部テーブルにアクセスできることを示します。PROTECTED は、BEA Tuxedo システム・ライブラリが、アプリケーションのコードによって破壊されないよう保護された共有メモリを使用して BEA Tuxedo システムの内部テーブルにアクセスできることを示します。NO_OVERRIDE は、アプリケーション・プロセスが tpinit(3c) または TPINITIALIZE(3cbl) で使用可能なフラグを使用して選択モードを変更できないことを示します。

制限事項 : (1) 実行中のアプリケーションでこの属性を変更しても、新たに起動したクライアントおよび新たにコンフィギュレーションした T_SERVER オブジェクトにしか反映されません。
(2) TA_SYSTEM_ACCESS を PROTECTED に設定しても、マルチスレッド・サーバには反映されない場合があります。これは、あるスレッドが BEA Tuxedo コードを実行している間、すなわちスレッドが掲示板に

アタッチされている間は、別のスレッドがユーザ・コードを実行している可能性があるためです。BEA Tuxedo システムでは、このような状況を回避することはできません。

TA_OPTIONS: "{[LAN | MIGRATE | ACCSTATS | NO_XA | NO_AA],*}"
有効なアプリケーション・オプションのカンマ区切りリスト。以下に、有効なオプションを定義します。

LAN— ネットワーク対応のアプリケーション

MIGRATE— サーバ・グループの移行を許可する

ACCSTATS— 正確な統計値 (SHM モードの場合のみ)

NO_XA—XA トランザクションの使用を許可しない

NO_AA— 監査および認証のプラグイン関数を呼び出さない

制限事項 : アクティブなアプリケーションでは、ACCSTATS のみを設定または再設定できます。

TA_USIGNAL: "{SIGUSR1 | SIGUSR2}"
シグナル・ベース方式の通知に使用するシグナル (前述の TA_NOTIFY を参照)。

TA_SECURITY: "{NONE | APP_PW | USER_AUTH | ACL | MANDATORY_ACL}"
アプリケーション・セキュリティの種類。長さゼロの文字列および NONE は、セキュリティ機能をオフにすることを示します。識別子 APP_PW は、アプリケーション・パスワードによるセキュリティを有効にすることを示します。クライアントは、初期化時にアプリケーション・パスワードを提示する必要があります。この属性の設定には、長さがゼロでない TA_PASSWORD 属性が必要です。識別子 USER_AUTH は APP_PW とほぼ同じですが、クライアントの初期化時にユーザごとの認証を実行する点が異なります。識別子 ACL は USER_AUTH とほぼ同じですが、サービス名、キュー名、およびイベント名に対してアクセス制御チェックを実行する点が異なります。ある名前に対応する ACL が見つからなかった場合は、パーミッションが付与されているものとみなされます。識別子 MANDATORY_ACL は、ACL とほぼ同じですが、名前に対応する ACL が見つからない場合にパーミッションを付与しない点が異なります。

注記　TA_OPTIONS パラメータで NO_AA 値が有効になっている場合、セキュリティ値 NONE、APP_PW、および USER_AUTH は正しく機能しますが、認証や監査は実行されません。また、ACL および MANDATORY_ACL パラメータも正常に機能しますが、デフォルトの BEA セキュリティ・メカニズムのみが使用されます。

TA_PASSWORD: *string*[0 . . 30]
平文のアプリケーション・パスワード。この属性は、TA_SECURITY 属性の値が設定されていない場合は無視されます。アプリケーション・パスワードは、システムによって自動的に暗号化されます。

TA_AUTHSVC: *string*[0..15]
システムに関与する各クライアントに対して呼び出されるアプリケーション認証サービス。TA_SECURITY 属性の値が設定されていない場合または APP_PW に設定されている場合、この属性は無視されます。

TA_SCANUNIT: 0 <= *num* <= 60 (5 の倍数)
システムが定期的に実行するスキャンの間隔 (単位は秒)。定期的なスキャンは、サービス要求内の古いトランザクションやタイムアウトになったブロッキング呼び出しを検出するために使用します。TA_BBLQUERY 属性、TA_BLOCKTIME 属性、TA_DBBLWAIT 属性、および TA_SANITYSCAN 属性は、この値の乗数で指定します。SET 操作においてこの属性値としてゼロを渡すと、属性値がデフォルト値にリセットされます。

TA_BBLQUERY: 0 <= *num* < 32,768
TA_SCANUNIT 属性の乗数。登録された BBL に対する DBBL 状態チェックの間隔を示します。DBBL は、すべての BBL の状態が TA_BBLQUERY で指定した期間内に報告されるようにします。BBL からの報告がない場合、DBBL はその BBL にメッセージを送信し、状態を照会します。応答がない場合、BBL は分断されます。SET 操作においてこの属性値としてゼロを渡すと、属性値がデフォルト値にリセットされます。この属性は、TA_SANITYSCAN 属性 (後述) の値の 2 倍以上の値に設定する必要があります。

TA_BLOCKTIME: 0 <= *num* < 32,768
TA_SCANUNIT 属性の乗数。ATMI のブロッキング呼び出しがタイムアウトする前にブロックする最短時間を示します。SET 操作においてこの属性値としてゼロを渡すと、属性値がデフォルト値にリセットされます。

TA_DBBLWAIT: 0 <= *num* < 32,768
: TA_SCANUNIT 属性の乗数。DBBL がタイムアウトする前に BBL からの応答を待機する最長時間を示します。SET 操作においてこの属性値としてゼロを渡すと、属性値がデフォルト値にリセットされます。

TA_SANITYSCAN: 0 <= *num* < 32,768
: TA_SCANUNIT 属性の乗数。システムに対する基本的な正常性チェックの間隔を示します。正常性チェックには、BBL ステータス・チェック・イン (MP モードの場合のみ) だけでなく、ローカル・マシンで実行しているクライアントまたはサーバの各 BBL によって行われるクライアントまたはサーバの実行可能状態のチェックも含まれます。SET 操作においてこの属性値としてゼロを渡すと、属性値がデフォルト値にリセットされます。

TA_CURDRT: 0 <= *num* < 32,768
: 掲示板のルーティング・テーブル・エントリの現在の使用数。

TA_CURGROUPS: 0 <= *num* < 32,768
: 掲示板のサーバ・グループ・テーブル・エントリの現在の使用数。

TA_CURMACHINES: 0 <= *num* < 32,768
: 現時点でコンフィギュレーションが済んでいるマシンの数。

TA_CURQUEUES: 0 <= *num* < 32,768
: 掲示板のキュー・テーブル・エントリの現在の使用数。

TA_CURRFT: 0 <= *num* < 32,768
: 掲示板のルーティング基準範囲テーブル・エントリの現在の使用数。

TA_CURRTDATA: 0 <= *num* < 32,768
: ルーティング・テーブルの文字列プールの現在のサイズ。

TA_CURSERVERS: 0 <= *num* < 32,768
: 掲示板のサーバ・テーブル・エントリの現在の使用数。

TA_CURSERVICES: 0 <= *num* < 32,768
: 掲示板のサービス・テーブル・エントリの現在の使用数。

TA_CURSTYPE: 0 <= *num* < 32,768
: 掲示板のサブタイプ・テーブル・エントリの現在の使用数。

TA_CURTYPE: 0 <= *num* < 32,768
: 掲示板のタイプ・テーブル・エントリの現在の使用数。

TA_HWDRT: 0 <= *num* < 32,768
掲示板のルーティング・テーブル・エントリの使用数の上限。

TA_HWGROUPS: 0 <= *num* < 32,768
掲示板のサーバ・グループ・テーブル・エントリの使用数の上限。

TA_HWMACHINES: 0 <= *num* < 32,768
コンフィギュレーションが済んでいるマシンの最大数。

TA_HWQUEUES: 0 <= *num* < 32,768
掲示板のキュー・テーブル・エントリの使用数の上限。

TA_HWRFT: 0 <= *num* < 32,768
掲示板のルーティング基準範囲テーブル・エントリの使用数の上限。

TA_HWRTDATA: 0 <= *num* < 32,768
ルーティング・テーブルの文字列プールのサイズの上限。

TA_HWSERVERS: 0 <= *num* < 32,768
掲示板のサーバ・テーブル・エントリの使用数の上限。

TA_HWSERVICES: 0 <= *num* < 32,768
掲示板のサービス・テーブル・エントリの使用数の上限。

TA_SEC_PRINCIPAL_NAME: *string*[0..511]
BEA Tuxedo 7.1 以降が動作するアプリケーションで認証用に使用されるセキュリティ・プリンシパル名。この属性の最大文字数は、文字列の最後を表す NULL 文字列を除いて 511 文字です。この属性に指定するプリンシパル名は、このドメインで実行される 1 つ以上のシステム・プロセスの識別子として使用されます。

TA_SEC_PRINCIPAL_NAME は、コンフィギュレーションの階層のうち、T_DOMAIN クラス、T_MACHINE クラス、T_GROUP クラス、および T_SERVER クラスの 4 つのレベルのどこでも指定できます。特定のコンフィギュレーション・レベルでのプリンシパル名は、下位レベルでオーバライドできます。TA_SEC_PRINCIPAL_NAME がどのレベルでも指定されていない場合、アプリケーションのプリンシパル名にはこのドメインの TA_DOMAINID 文字列がデフォルトで設定されます。

TA_SEC_PRINCIPAL_NAME のほかにも、TA_SEC_PRINCIPAL_LOCATION と TA_SEC_PRINCIPAL_PASSVAR という属性があります。後の 2 つの属性は、アプリケーション起動時に、BEA Tuxedo 7.1 以降で動作するシス

テム・プロセスに対して復号化キーのオープンする処理に関係する属性です。特定のレベルで TA_SEC_PRINCIPAL_NAME のみが指定されている場合には、それ以外の 2 つの属性に長さゼロの NULL 文字列が設定されます。

TA_SEC_PRINCIPAL_LOCATION: *string*[0..511]

TA_SEC_PRINCIPAL_NAME に指定したプリンシパルの復号化 (秘密) キーを格納するファイルまたはデバイスのロケーション。この属性の最大文字数は、文字列の最後を表す NULL 文字列を除いて 511 文字です。

TA_SEC_PRINCIPAL_LOCATION は、コンフィギュレーションの階層のうち、T_DOMAIN クラス、T_MACHINE クラス、T_GROUP クラス、および T_SERVER クラスの 4 つのレベルのどこでも指定できます。この属性は、どのレベルで指定する場合でも TA_SEC_PRINCIPAL_NAME 属性と対になっている必要があり、それ以外の場合には無視されます (TA_SEC_PRINCIPAL_PASSVAR はオプションです。この属性が指定されていない場合、システムによって長さゼロの NULL 文字列が設定されます)。

TA_SEC_PRINCIPAL_PASSVAR: *string*[0..511]

TA_SEC_PRINCIPAL_NAME に指定したプリンシパルのパスワードを格納する変数。この属性の最大文字数は、文字列の最後を表す NULL 文字列を除いて 511 文字です。

TA_SEC_PRINCIPAL_PASSVAR は、コンフィギュレーションの階層のうち、T_DOMAIN クラス、T_MACHINE クラス、T_GROUP クラス、および T_SERVER クラスの 4 つのレベルのどこでも指定できます。この属性は、どのレベルで指定する場合でも TA_SEC_PRINCIPAL_NAME 属性と対になっている必要があり、それ以外の場合には無視されます (TA_SEC_PRINCIPAL_LOCATION はオプションです。この属性が指定されていない場合、システムによって長さゼロの NULL 文字列が設定されます)。

初期化時は、TA_SEC_PRINCIPAL_PASSVAR に設定した復号化キーの各パスワードを管理者が入力する必要があります。管理者が入力したパスワードはシステム側で自動的に暗号化され、それぞれが対応するパスワード変数に割り当てられます。

TA_SIGNATURE_AHEAD: 1 <= *num* <= 2147483647
: デジタル署名のタイムスタンプとして許容する時刻の範囲 (秒)。ローカル・マシンのシステム・クロックを基準とし、それより何秒後まで許容するかを指定します。指定しない場合、デフォルト値の 3600 秒 (1 時間) が設定されます。この属性は、BEA Tuxedo 7.1 以降が動作するアプリケーションにのみ適用されます。

TA_SIGNATURE_BEHIND: 1 <= *num* <= 2147483647
: デジタル署名のタイムスタンプとして許容する時刻の範囲 (秒)。ローカル・マシンのシステム・クロックを基準とし、それより何秒前まで許容するかを指定します。指定しない場合、デフォルト値の 604800 秒 (1 週間) が設定されます。この属性は、BEA Tuxedo 7.1 以降が動作するアプリケーションにのみ適用されます。

TA_SIGNATURE_REQUIRED: "{Y | N}"
: "Y" に設定すると、このドメインで実行するすべてのプロセスで、その入力メッセージ・バッファのデジタル署名が必要となります。指定しない場合、デフォルト値の "N" が設定されます。この属性は、BEA Tuxedo 7.1 以降が動作するアプリケーションにのみ適用されます。

  TA_SIGNATURE_REQUIRED は、コンフィギュレーションの階層のうち、T_DOMAIN クラス、T_MACHINE クラス、T_GROUP クラス、および T_SERVICE クラスの 4 つのレベルのどこでも指定できます。特定のレベルで SIGNATURE_REQUIRED に "Y" を設定すると、下位レベルで動作するすべてのプロセスに署名が必要となります。

TA_ENCRYPTION_REQUIRED: "{Y | N}"
: "Y" に設定すると、このドメインで実行するすべてのプロセスで暗号化された入力メッセージ・バッファが必要となります。指定しない場合、デフォルト値の "N" が設定されます。この属性は、BEA Tuxedo 7.1 以降が動作するアプリケーションにのみ適用されます。

  TA_ENCRYPTION_REQUIRED は、コンフィギュレーションの階層のうち、T_DOMAIN クラス、T_MACHINE クラス、T_GROUP クラス、および T_SERVICE クラスの 4 つのレベルのどこでも指定できます。特定のレベルで TA_ENCRYPTION_REQUIRED に "Y" を設定すると、下位レベルで動作するすべてのプロセスに暗号化が必要となります。

制限事項　このクラスの属性の多くは、アプリケーションが非アクティブなときにしか調節できません。つまり、ATMI インターフェイス・ルーチンを使用してアプリケーションを管理することはできません。そのため、起動されていないアプリケーションをコンフィギュレーションしたりコンフィギュレーションしなおしたりするための手段として、tpadmcall() という関数が用意されています。このインターフェイスは、アプリケーションのマスタ・サイトとして設定されたサイトのみで、非アクティブなアプリケーションのコンフィギュレーション (SET 操作) のためだけに使用します。いったん初期のコンフィギュレーションを作成してアクティブにすると、MIB(5) で説明した標準の ATMI インターフェイスを使用してアプリケーションを管理することが可能になります。

T_FACTORY MIB

概要 T_FACTORY MIB クラスは、FactoryFinder に登録されているファクトリのオカレンスを表します。アプリケーションで使用可能なファクトリがこの MIB に反映され、管理者は Administration Console またはコマンド行ツールを使用してそれを確認できます。スコープはグローバルです。

属性表

表 47TM_MIB(5): T_FACTORY **属性**

属性	使用法	タイプ	パーミッション	値	デフォルト値
TA_STATE	k	string	R--R--R--	GET:"{ACT}"	N/A
TA_FACTORYID	k	string	R--R--R--	*string*[1..25]	N/A
TA_INTERFACENAME	k	string	R--R--R--	*string*[1..128]	N/A
(k)GET キー・フィールド					

属性の意味 TA_STATE

GET: {ACTive }

GET 操作は、選択した T_FACTORY オブジェクトのコンフィギュレーション情報および実行時情報を取得します。
以下に示す状態は、GET 要求への応答で返される TA_STATE の意味を示します。

ACTive	T_FACTORY オブジェクトは、FactoryFinder に登録されています。

TA_FACTORY

ファクトリの登録 ID。

TA_INTERFACENAME

ファクトリの完全修飾インターフェイス名。ファクトリのインターフェイス・リポジトリ ID です。この名前の形式は、インターフェイスのインプリメンテーションを生成する IDL に指定されたオプション

によって異なります。詳細については、CORBA 2.1 仕様のセクション 7.6 を参照してください。

T_GROUP クラスの定義

概要 T_GROUP クラスは、特定のサーバ・グループに関係のあるアプリケーション属性を表します。これらの属性値は、グループの識別、ロケーション、DTP に関する情報を表します。

属性表

表 48TM_MIB(5): T_GROUP **クラス定義の属性表**

属性	タイプ	パーミッション	値	デフォルト値
TA_SRVGRP(r)(*)	string	rU-r--r--	*string*[1. .30]	N/A
TA_GRPNO(k)(r)	long	rU-r--r--	1 <= *num* < 30,000	N/A
TA_LMID(k)(r) (注 1)	string	rwyr--r--	"*LMID1*[,*LMID2*]"	N/A
TA_STATE(k)	string	rwxr-xr--	GET: "{ACT | INA | MIG}"	N/A
			SET: "{NEW | INV | ACT | RAC |INA | MIG}"	N/A
TA_CURLMID(k)	string	R--R--R--	*LMID*	N/A
TA_ENVFILE	string	rwyr--r--	*string*[0..256] (注 2)	""
TA_OPENINFO	string	rwyr--r--	*string*[0..256]	""
TA_CLOSEINFO	string	rwyr--r--	*string*[0..256]	""
TA_TMSCOUNT	long	rw-r--r--	0 or 2 <= *num* < 11	3
TA_TMSNAME(k)	string	rw-r--r--	*string*[0..256] (注 2)	""
TA_SEC_PRINCIPAL_NAME	string	rwxr--r--	*string*[0..511]	""
TA_SEC_PRINCIPAL_LOCATION	string	rwxr--r--	*string*[0..511]	""
TA_SEC_PRINCIPAL_PASSVAR	string	rwxr--r--	*string*[0..511]	""
TA_SIGNATURE_REQUIRED	string	rwxr--r--	"{Y|N}"	"N"
TA_ENCRYPTION_REQUIRED	string	rwxr--r--	"{Y|N}"	"N"

(k)—GET キー・フィールド
(r)— オブジェクトの作成に必要なフィールド (SET TA_STATE NEW)
(*)—GET/SET キー、SET 操作では 1 つ以上必要

注 1 TA_LMID は、このクラス内で一意である必要があります。
注 2 BEA Tuxedo 8.0 以前では、この属性の文字列の最大長は 78 バイトです。

属性の意味

TA_SRVGRP: *string*[1..30]
サーバ・グループの論理名。グループ名は、T_GROUP クラスのすべてのグループ名、および T_MACHINE クラスの TA_LMID の値と重複しない一意な名前である必要があります。また、サーバ・グループ名にはアスタリスク (*)、カンマ (,)、コロン (:) は使用できません。

TA_GRPNO: 1 <= *num* < 30,000
このサーバ・グループに関連付けられたグループ番号。

TA_LMID: "*LMID1*[,*LMID2*]"
このサーバ・グループのプライマリ・マシンの論理マシン識別子 (*LMID1*) と、省略可能なセカンダリの論理マシンの識別子 (*LMID2*)。セカンダリの LMID は、サーバ・グループの移行先のマシンを示します (T_DOMAIN:TA_OPTIONS 属性の MIGRATE オプションが指定されている場合)。GET 操作で指定する LMID は、プライマリ、セカンダリのどちらかの LMID と一致します。アクティブなグループのロケーションは、TA_CURLMID 属性で使用できます。TA_LMID 属性で指定する論理マシン識別子はコンフィギュレーション済みである必要があります。制限事項 : アクティブなオブジェクトでこの属性を変更しても、グループのバックアップ LMID しか変更されません。

TA_STATE:

GET: "{ACTive | INActive | MIGrating}"
GET 操作は、選択した T_GROUP オブジェクトのコンフィギュレーション情報および実行時情報を取得します。以下に示す状態は、GET 要求への応答で返される TA_STATE の意味を示します。

ACTive	定義済みでアクティブな状態にある T_GROUP オブジェクト (TMS またはアプリケーション・サーバ)。TA_TMSNAME 属性が NULL 以外の文字列に設定されているサーバ・グループは、そのグループに関連付けられた TMS がアクティブであればグループ自体もアクティブとみなされます。それ以外の場合は、グループ内のいずれかのサーバがアクティブであればアクティブとみなされます。

INActive	定義済みでアクティブでない状態にある T_GROUP オブジェクト。
MIGrating	定義済みで、現時点でセカンダリ論理マシンへの移行状態にある T_GROUP オブジェクト。セカンダリ論理マシンとは、TA_LMID に登録された論理マシンのうち、TA_CURLMID に該当しない論理マシンです。この状態は、パーミッションの決定においては ACTive と同等です。

SET: "{NEW | INValid | ACTive | ReACtivate | INActive | MIGrating}"

SET 操作は、選択した T_GROUP オブジェクトのコンフィギュレーション情報および実行時情報を更新します。以下に示す状態は、SET 要求で設定される TA_STATE の意味を示します。以下に示されていない状態は設定できません。

NEW	アプリケーションの T_GROUP オブジェクトを作成します。状態の変更は INValid 状態でのみ可能です。正常に終了すると、オブジェクトの状態は INActive になります。
unset	既存の T_GROUP オブジェクトを変更します。この組み合わせは、ACTive 状態または INActive 状態でのみ可能です。正常終了した場合、オブジェクトの状態は変わりません。
INValid	アプリケーションの T_GROUP オブジェクトを削除します。状態の変更は、INActive 状態でのみ可能です。正常終了すると、オブジェクトの状態は INValid になります。

ACTive	T_GROUP オブジェクトをアクティブにします。状態の変更は、INActive 状態または MIGrating 状態でのみ可能です。この状態遷移に対するパーミッションの決定に際しては、アクティブなオブジェクトのパーミッション (--x--x--x) が考慮されます。 グループの現在の状態が INActive で、プライマリ論理マシンがアクティブな場合は、TMS とアプリケーション・サーバ (TA_FLAGS の設定による制約が適用される) はプライマリ論理マシンで起動されます。それ以外の場合は、セカンダリ論理マシン上で起動されます (ただし、セカンダリ論理マシンがアクティブな場合)。どちらのマシンもアクティブでない場合、要求は実行されません。 グループの現在の状態が MIGrating である場合は、TMS とアプリケーション・サーバの実行マシンとして、アクティブなセカンダリ論理マシン (TA_LMID リストで TA_CURLMID の代替マシンとして指定されたマシン) が使用されます。これ以外の場合は要求は実行されません。サーバ・グループをアクティブにする際に個々のサーバの状態が必要な場合には、TMIB_NOTIFY TA_FLAG 値を使用する必要があります。 正常に終了すると、オブジェクトの状態は ACTive になります。
ReACtivate	ACTive 状態への変更とほぼ同じですが、状態が INActive や MIGrating である場合だけでなく、ACTive である場合にも状態の変更が可能である点が異なります。 サーバ・グループを再びアクティブにする際に個々のサーバの状態が必要な場合には、TMIB_NOTIFY TA_FLAG 値を使用する必要があります。

INActive	T_GROUP オブジェクトを非アクティブにします。TMS とアプリケーション・サーバ (TA_FLAGS の設定による制約が適用される) も非アクティブになります。状態の変更は、ACTive 状態または MIGrating 状態でのみ可能です。正常に終了すると、オブジェクトの状態は INActive になります。 サーバ・グループを非アクティブにする際に個々のサーバの状態が必要な場合には、TMIB_NOTIFY TA_FLAG 値を使用する必要があります。
MIGrating	アクティブな論理マシン (TA_CURLMID) の T_GROUP オブジェクトを非アクティブにし、グループがセカンダリ論理マシンに移行できるようにします。状態の変更は、ACTive 状態でのみ可能です。正常に終了すると、オブジェクトの状態は MIGrating になります。
UnAVailable	グループ内のすべてのアプリケーション・サービスを中断します。(注：アプリケーション・サービスを個別に中断するには T_SVCGROUP クラスを使用します)。この状態に対する SET 操作は、グループが ACTive である場合にのみ実行可能です。この操作を実行すると、グループを ACTive 状態にしたまま、すべてのアプリケーションを中断状態にできます。制限事項 : リリース 6.4 以前のアクティブなマシンが混在したアプリケーションでは操作が異常終了します。
AVaiLable	グループ内で中断とマークされているすべてのアプリケーションの中断を解除します。この状態に対する SET 操作は、グループが ACTive である場合にのみ実行可能です。この操作を実行すると、グループの状態は ACTive のままとなります。

制限事項 : リリース 6.4 以前のアクティブなマシンが混在したアプリケーションでは操作が異常終了します。

TA_CURLMID:*LMID*
: サーバ・グループを実行している現在の論理マシン。この属性は、非アクティブなサーバ・グループでは返されません。

TA_ENVFILE: *string*[0..256] (BEA Tuxedo 8.0 以前では最大 78 バイト)
: このグループ内で実行するサーバの環境ファイル。無効なファイル名を指定すると環境に追加されません。*string* の値は環境内に配置されます。

  起動時は、ローカル・サーバが tmboot(1) の環境を継承し、MASTER 上にないリモート・サーバが tlisten(1) の環境を継承します。また、対応する T_GROUP オブジェクトの情報に基づいてサーバが起動されると、TUXCONFIG、TUXDIR、および APPDIR も環境に配置されます。

  PATH は環境内で次のように設定されます。

  APPDIR:TUXDIR/bin:/bin:/usr/bin:*path*

  path は、マシンの環境ファイルの最初の PATH= 行の値です。これ以降の PATH= 行はすべて無視されます。この PATH の値は、サーバの検索パスとして使用され、単純なパス名または相対パス名で指定されます。したがって、先頭はスラッシュではありません。

  LD_LIBRARY_PATH は環境内で次のように設定されます。

  APPDIR:TUXDIR/lib:/lib:/usr/lib:*lib*

  lib は、マシンの環境ファイルの最初の LD_LIBRARY_PATH= 行の値です。これ以降の LD_LIBRARY_PATH= 行はすべて無視されます。

  サーバの初期化時 (tpsvrinit(3c) を呼び出す前) には、サーバがマシンとサーバの両方の ENVFILE ファイルの変数を読み取ってエクスポートします。変数がマシンとサーバの両方の ENVFILE ファイルに設定されている場合は、サーバの ENVFILE ファイルの値によってマシンの ENVFILE ファイルの値がオーバライドされます。ただし、PATH はオーバライドではなく追加されます。クライアントはマシンの ENVFILE ファイルのみを処理します。マシンとサーバの ENVFILE ファイルの処理では、*ident*= の形式でない行は無視されます。*ident* に含めることができるのはアンダースコアまたは英数字のみです。

  PATH= 行が見つかると、PATH が次のように設定されます。

APPDIR:TUXDIR/bin:/bin:/usr/bin:*path*

path は、マシンの環境ファイルの最初の PATH= 行の値です。これ以降の PATH= 行はすべて無視されます。マシンとサーバの両方の環境ファイルに PATH が存在する場合、*path* は *path1:path2* となります。*path1* はマシンの ENVFILE から読み取ったパス、*path2* はサーバの ENVFILE から読み取ったパスです。LD_LIBRARY_PATH= 行が見つかると、LD_LIBRARY_PATH が次のように設定されます。

APPDIR:TUXDIR/lib:/lib:/usr/lib:*lib*

lib は、マシンの環境ファイルの最初の LD_LIBRARY_PATH= 行の値です。これ以降の LD_LIBRARY_PATH= 行はすべて無視されます。TUXDIR、APPDIR、または TUXCONFIG をリセットしようとしても、対応する T_GROUP 属性値と値が一致していない場合には無視されて警告メッセージが表示されます。制限事項 : アクティブなオブジェクトでこの属性を変更しても、実行中のサーバやクライアントには反映されません。

TA_OPENINFO: *string*[0..256]

このグループのリソース・マネージャをオープンするときに必要な、リソース・マネージャ・インスタンスに依存する情報。この値は二重引用符で囲む必要があり、その長さは 256 文字以下でなければなりません。

TA_TMSNAME 属性の値に TMS 以外の NULL でない文字列を指定した場合、TA_OPENINFO 属性の値は、リソース・マネージャへのアクセスを開始する際に必要なリソース・マネージャに依存する情報を提供します。それ以外の場合、TA_OPENINFO 属性の値は無視されます。

TA_OPENINFO 属性の値が NULL 文字列である場合、このグループのリソース・マネージャがリソースへの open アクセスにアプリケーション固有の情報を必要としないことを意味します。

TA_OPENINFO 文字列の形式は、基となるリソース・マネージャのベンダごとに異なります。ベンダ固有の情報の先頭には、トランザクション・インターフェイス (XA インターフェイス) の公開名とコロン (:) が付きます。

BEA Tuxedo /Q データベースでは、次のような形式になります。

```
# UNIX の場合 #
OPENINFO = "TUXEDO/QM:qmconfig:qspace"

# Windows の場合 #
OPENINFO = "TUXEDO/QM:qmconfig;qspace"

# AS/400 環境の場合 #
OPENINFO = "TUXEDO/QM:qmconfig;qspace"

# OpenVMS 環境の場合 #
OPENINFO = "TUXEDO/QM,[a.b.c]qmconfig,qspace"
```

TUXEDO/QM は、BEA Tuxedo /Q XA インターフェイスの公開名です。*qmconfig* は、キュー・スペースを設定する QMCONFIG (BEA Tuxedo の qmadmin(1) 参照) の名前です。*qspace* はキュー・スペースの名前です。Windows および AS/400 では、*qmconfig* の後に指定する区切り文字として、セミコロン (;) を使用します。OpenVMS では、TUXEDO/QM および *qmconfig* の後ろの区切りはカンマ (,) でなければなりません。

その他のベンダのデータベースでは、TA_OPENINFO 文字列の形式は、基となるリソース・マネージャのベンダごとに異なります。

制限事項 : この属性を実行時に変更しても、グループ内のアクティブなサーバには反映されません。

TA_CLOSEINFO: *string*[0..256]

このグループのリソース・マネージャをクローズするときに必要な、リソース・マネージャ・インスタンスに依存する情報。この値は二重引用符で囲む必要があり、その長さは 256 文字以下でなければなりません。BEA Tuxedo /Q データベースでは、TA_CLOSEINFO 文字列は使用しません。

TA_TMSNAME 属性の値に TMS 以外の NULL でない文字列を指定した場合、TA_CLOSEINFO 属性の値は、リソース・マネージャへのアクセスを終了する際に必要なリソース・マネージャに依存する情報を提供します。それ以外の場合、TA_CLOSEINFO 属性の値は無視されます。

TA_CLOSEINFO 属性の値が NULL 文字列である場合、このグループのリソース・マネージャがリソースへの close アクセスにアプリケーション固有の情報を必要としないことを意味します。

TA_CLOSEINFO 文字列の形式は、基となるリソース・マネージャのベンダごとに異なります。ベンダ固有の情報の先頭には、トランザクション・インターフェイス (XA インターフェイス) の公開名とコロン (:) が付きます。

制限事項 : この属性を実行時に変更しても、グループ内のアクティブなサーバには反映されません。

TA_TMSCOUNT:0 または 2 <= *num* < 11

TA_TMSNAME 属性の値に NULL でない文字列を指定した場合、TA_TMSCOUNT 属性値は関連付けられたグループ用に起動するトランザクション・マネージャ・サーバの数を示します。それ以外の場合、この属性の値は無視されます。

TA_TMSNAME: *string*[0..256] (BEA Tuxedo 8.0 以前では最大 78 バイト)

このグループに関連付けられているトランザクション・マネージャ・サーバの a.out。分散トランザクション (tpbegin() で開始し、tpcommit()/tpabort() で終了する、複数のリソース・マネージャやマシンの間で処理されるトランザクション) に参加するサーバを持つグループ・エントリに対しては、この属性を必ず指定する必要があります。

値 TMS は、NULL XA インターフェイスの使用を示すために予約されています。TMS 以外の空でない値を指定した場合は、このオブジェクトのプライマリおよびセカンダリ論理マシンに関連付けられたマシンに対して TLOGDEVICE を指定する必要があります。

各 TM サーバに対して一意のサーバ識別子が自動的に選択されるため、サーバは何度でも再起動できます。

TA_SEC_PRINCIPAL_NAME: *string*[0..511]

BEA Tuxedo 7.1 以降が動作するアプリケーションで認証用に使用されるセキュリティ・プリンシパル名。この属性の最大文字数は、文字列の最後を表す NULL 文字列を除いて 511 文字です。この属性に指定するプリンシパル名は、このグループで実行される 1 つ以上のシステム・プロセスの識別子として使用されます。

TA_SEC_PRINCIPAL_NAME は、コンフィギュレーションの階層のうち、T_DOMAIN クラス、T_MACHINE クラス、T_GROUP クラス、および T_SERVER クラスの 4 つのレベルのどこでも指定できます。特定のコン

フィギュレーション・レベルでのプリンシパル名は、下位レベルでオーバライドできます。TA_SEC_PRINCIPAL_NAME がどのレベルでも指定されていない場合、アプリケーションのプリンシパル名にはこのドメインの TA_DOMAINID 文字列がデフォルトで設定されます。

TA_SEC_PRINCIPAL_NAME のほかにも、TA_SEC_PRINCIPAL_LOCATION と TA_SEC_PRINCIPAL_PASSVAR という属性があります。後の 2 つの属性は、アプリケーション起動時に、BEA Tuxedo 7.1 以降で動作するシステム・プロセスに対して復号化キーのオープンする処理に関係する属性です。特定のレベルで TA_SEC_PRINCIPAL_NAME のみが指定されている場合には、それ以外の 2 つの属性に長さゼロの NULL 文字列が設定されます。

TA_SEC_PRINCIPAL_LOCATION: *string*[0..511]

TA_SEC_PRINCIPAL_NAME に指定したプリンシパルの復号化 (秘密) キーを格納するファイルまたはデバイスの位置。この属性の最大文字数は、文字列の最後を表す NULL 文字列を除いて 511 文字です。

TA_SEC_PRINCIPAL_LOCATION は、コンフィギュレーションの階層のうち、T_DOMAIN クラス、T_MACHINE クラス、T_GROUP クラス、および T_SERVER クラスの 4 つのレベルのどこでも指定できます。この属性は、どのレベルで指定する場合でも TA_SEC_PRINCIPAL_NAME 属性と対になっている必要があり、それ以外の場合には無視されます (TA_SEC_PRINCIPAL_PASSVAR はオプションです。この属性が指定されていない場合、システムによって長さゼロの NULL 文字列が設定されます)。

TA_SEC_PRINCIPAL_PASSVAR: *string*[0..511]

TA_SEC_PRINCIPAL_NAME に指定したプリンシパルのパスワードを格納する変数。この属性の最大文字数は、文字列の最後を表す NULL 文字列を除いて 511 文字です。

TA_SEC_PRINCIPAL_PASSVAR は、コンフィギュレーションの階層のうち、T_DOMAIN クラス、T_MACHINE クラス、T_GROUP クラス、および T_SERVER クラスの 4 つのレベルのどこでも指定できます。この属性は、どのレベルで指定する場合でも TA_SEC_PRINCIPAL_NAME 属性と対になっている必要があり、それ以外の場合には無視されます (TA_SEC_PRINCIPAL_LOCATION はオプションです。この属性が指定さ

れていない場合、システムによって長さゼロの NULL 文字列が設定されます)。

初期化時は、TA_SEC_PRINCIPAL_PASSVAR に設定した復号化キーの各パスワードを管理者が入力する必要があります。管理者が入力したパスワードはシステム側で自動的に暗号化され、それぞれが対応するパスワード変数に割り当てられます。

TA_SIGNATURE_REQUIRED: "{Y | N}"

"Y" に設定すると、このグループで実行するすべてのプロセスで、その入力メッセージ・バッファのデジタル署名が必要となります。指定しない場合、デフォルト値の "N" が設定されます。この属性は、BEA Tuxedo 7.1 以降が動作するアプリケーションにのみ適用されます。

TA_SIGNATURE_REQUIRED は、コンフィギュレーションの階層のうち、T_DOMAIN クラス、T_MACHINE クラス、T_GROUP クラス、および T_SERVICE クラスの 4 つのレベルのどこでも指定できます。特定のレベルで SIGNATURE_REQUIRED に "Y" を設定すると、下位レベルで動作するすべてのプロセスに署名が必要となります。

TA_ENCRYPTION_REQUIRED: "{Y | N}"

"Y" に設定すると、このグループで実行するすべてのプロセスで暗号化された入力メッセージ・バッファが必要となります。指定しない場合、デフォルト値の "N" が設定されます。この属性は、BEA Tuxedo 7.1 以降が動作するアプリケーションにのみ適用されます。

TA_ENCRYPTION_REQUIRED は、コンフィギュレーションの階層のうち、T_DOMAIN クラス、T_MACHINE クラス、T_GROUP クラス、および T_SERVICE クラスの 4 つのレベルのどこでも指定できます。特定のレベルで TA_ENCRYPTION_REQUIRED に "Y" を設定すると、下位レベルで動作するすべてのプロセスに暗号化が必要となります。

制限事項　なし

T_IFQUEUE Class

概要 T_IFQUEUE クラスは、CORBA 環境における特定のサーバ・キュー (T_QUEUE) に関係するクラスで、インターフェイスの実行時属性を表します。基本的に読み取り専用で、インターフェイスが継承するコンフィギュレーション属性へのアクセスを提供するだけでなく、キュー内のインターフェイスに関連する統計情報を提供します。また、管理者はこのクラスを使用して、インターフェイスをきめ細かく中断したりアクティブにしたりできます。このクラスは、インターフェイス名と、インターフェイス上のメソッド呼び出しを処理できるサーバ・プロセスとの間のリンクを提供します。つまり、TA_RQADDR を T_SERVER クラス上のキー検索フィールドとして使用することができます。

属性表

表 49TM_MIB(5): T_IFQUEUE クラス定義の属性表

属性	使用法	タイプ	パーミッション	値	デフォルト値
TA_INTERFACENAME	*	string	R--R--R--	*string*[1..128]	N/A
TA_SRVGRP	*	string	R--R--R--	*string*[1..30]	N/A
TA_RQADDR	*	string	R--R--R--	*string*[1..30]	N/A
TA_STATE	k	string	R-XR-XR--	GET: "{ACT | SUS | PAR}" SET: "{ACT | SUS}"	N/A
TA_AUTOTRAN		string	R--R--R--	"{Y | N}"	N/A
TA_LOAD		long	R--R--R--	1 <= *num* < 32K	N/A
TA_PRIO		long	R--R--R--	1 <= *num* < 101	N/A
TA_TIMEOUT		long	R--R--R--	0 <= *num*	N/A
TA_TRANTIME		long	R--R--R--	0 <= *num*	N/A
TA_FBROUTINGNAME		string	R--R--R--	*string*[1..15]	N/A
TA_LMID	k	string	R--R--R--	*LMID*	N/A
TA_NUMSERVERS		long	R--R--R--	0 <= *num*	N/A

表 49TM_MIB(5): T_IFQUEUE **クラス定義の属性表 (続き)**

TA_TPPOLICY		string	R--R--R--	"{method|transaction |process}"	N/A
TA_TXPOLICY		string	R--R--R--	"{always|never| optional|ignore}"	N/A
TA_NCOMPLETED	l	long	R-XR-XR--	0 <= *num*	N/A
TA_NQUEUED	l	long	R--R--R--	0 <= *num*	N/A
TA_CUROBJECTS	l	long	R--R--R--	0 <= *num*	N/A
TA_CURTRANSACTIONS	l	long	R--R--R--	0 <= *num*	N/A

(k)GET キー・フィールド
(l) - ローカル・フィールド
(*) - GET/SET キー。SET 操作では 1 つ以上必要

属性の意味

TA_INTERFACENAME: *string*[1..128]
完全修飾されたインターフェイス名。インターフェイスのインターフェイス・リポジトリ ID です。この名前の形式は、インターフェイスのインプリメンテーションを生成する IDL に指定されたオプションによって異なります。詳細については、CORBA 2.1 仕様のセクション 7.6 を参照してください。

TA_SRVGRP: *string*[0..30]
サーバ・グループ名。サーバ・グループ名にはアスタリスク、カンマ、コロンは使用できません。

TA_RQADDR: *string*[1..30]
このインターフェイスを提供するアクティブなサーバの要求キューのシンボリック・アドレス。この属性の詳細については、T_SERVER:TA_RQADDR を参照してください。

TA_STATE:

GET: "{ACTive | SUSpended | PARtitioned}"
GET 操作は、選択した T_IFQUEUE オブジェクトのコンフィギュレーション情報を検索します。以下に示す状態は、GET 要求へ

の応答で返される TA_STATE の意味を示します。これら以外の状態は返されません。

ACTive	T_IFQUEUE オブジェクトは、実行中のシステムで使用できるインターフェイスを表します。
SUSpended	T_IFQUEUE オブジェクトは、実行中のシステムで中断されているインターフェイスを表します。
PARtitioned	T_IFQUEUE オブジェクトは、実行中のシステムで分断されているインターフェイスを表します。

SET: "{ACTive | SUSpended}"

以下に示す状態は、SET 要求で設定される TA_STATE の意味を示します。以下に示されていない状態は設定できません。

ACTive	T_IFQUEUE オブジェクトをアクティブにします。状態の変更は、SUSpended 状態でのみ可能です。正常に終了すると、オブジェクトの状態は ACTive になります。
SUSpended	T_IFQUEUE オブジェクトを中断します。状態の変更は、ACTive 状態でのみ可能です。正常に終了すると、オブジェクトの状態は SUSpended になります。

制限事項 : インターフェイスの動的な宣言 (INActive または INValid から ACTive への状態変更) はサポートされていません。また、非宣言 (ACTive から INActive への状態変更) もサポートされていません。

TA_AUTOTRAN: "{Y | N}"

トランザクション・コンテキストの範囲外で実行された呼び出しに対して、自動的にトランザクションを開始するかどうかを示します。制限事項については、この属性の T_INTERFACE の説明を参照してください。

TA_LOAD: 1 <= *num* <= 32K
: このT_INTERFACEオブジェクトは、システムに対する負荷を設定します。インターフェイスの負荷は、ロード・バランシングのために使用します。つまり、すでに負荷が大きいキューは、新規の要求ではあまり選択されません。

TA_PRIO: 1 <= *num* <= 101
: このT_INTERFACEオブジェクトは、指定された優先順位でキューから取り出されます。複数のインターフェイス要求がサービス・キューで待機している場合、優先順位の高い要求から処理されます。

TA_TIMEOUT: 0 <= *num*
: このインターフェイスの個々のメソッド呼び出しを処理する際の時間制限 (単位は秒)。このインターフェイスのメソッドの呼び出しを処理するサーバは、要求の処理が指定した時間制限値を超えると異常終了します。この属性を 0 に設定すると、サーバは異常終了しません。

TA_TRANTIME: 0 <= *num*
: このT_INTERFACEオブジェクト用に自動的に開始されたトランザクションのトランザクション・タイムアウト値 (単位は秒)。インターフェイスのT_INTERFACE:TA_AUTOTRAN属性値がYである場合に、トランザクション・モードでない要求を受信すると、トランザクションが自動的に開始されます。

TA_FBROUTINGNAME: *string*[1..15]
: このインターフェイスに関連付けられたファクトリ・ベースのルーティング基準。

TA_LMID: *LMID*
: このインターフェイスを提供するキューが配置されている現在の論理マシン。

TA_NUMSERVERS: 0 <= *num*
: このキューでこのインターフェイスを提供する対応サーバの数。

TA_TPPOLICY: "{method | transaction | process}"
: TP フレームワークの非アクティブ化方針。サーバの起動時にフレームワークに登録される方針を反映します。インターフェイスを登録する最初のサーバが、T_INTERFACEの値を設定します。この値は変更できません。

TA_TXPOLICY: "{optional | always | never | ignore}"
インターフェイスのトランザクション方針。この属性の設定は、TA_AUTOTRAN 属性の効果に影響します。詳細については、TA_AUTOTRAN を参照してください。この属性は常に読み取り専用です。この属性は、開発者がサーバの構築時に設定し、サーバの起動時に登録されます。

TA_NCOMPLETED: 0 <= *num*
インターフェイスの最初の提供より後に完了したインターフェイスのメソッド呼び出しの数。

TA_NQUEUED: 0 <= *num*
このインターフェイスのキューに現時点で登録されている要求の数。

TA_CUROBJECTS: 0 <= *num*
関連付けられたキューに対する、このインターフェイスのアクティブなオブジェクトの数。この数値は、関連付けられたマシン上のキューにおけるアクティブなオブジェクト・テーブル内のエントリの数を表します。この数には、メモリ内にはないものの、アクティブなトランザクション内で呼び出されたオブジェクトが含まれています。

TA_CURTRANSACTIONS: 0 <= *num*
関連付けられたキューに対し、このインターフェイスに関連付けられたアクティブなグローバル・トランザクションの数。

T_INTERFACE クラス

概要 T_INTERFACE MIB クラスは、ドメインおよびサーバ・グループの両方のレベルで CORBA インターフェイスのコンフィギュレーション属性と実行時属性を表します。

ドメイン・レベルの T_INTERFACE オブジェクトは、サーバ・グループに関連付けられていないオブジェクトです。その TA_SRVGRP 属性には、NULL 文字列 (長さ 0 の文字列、"") が格納されます。

サーバ・グループ・レベルの T_INTERFACE オブジェクトは、関連付けられたサーバ・グループを持つオブジェクトです。つまり、その TA_SRVGRP 属性には、ドメインに対する有効なサーバ・グループ名が格納されます。インターフェイスのサーバ・グループ・レベル表現は、インターフェイスの状態 (TA_STATE) の管理や、蓄積された統計情報の収集に使用するコンテナも提供します。

サーバ内でアクティブ化されたすべての CORBA インターフェイスに対して、対応するサーバ・グループ・レベルの T_INTERFACE オブジェクトが存在する必要があります。サーバ内のインターフェイスのアクティブ化は、このインターフェイスの T_IFQUEUE オブジェクトの状態によって制御されます。T_IFQUEUE オブジェクトをアクティブ化すると、その属性が、対応するサーバ・グループ・レベルの T_INTERFACE オブジェクトに指定された値によって初期化されます。このようなオブジェクトが存在しない場合は動的に作成されます。動的に作成されたサーバ・グループ・レベルの T_INTERFACE オブジェクトは、インターフェイスにドメイン・レベルの T_INTERFACE オブジェクトが存在する場合はその属性によって初期化されます。対応するドメイン・レベルの T_INTERFACE オブジェクトが存在しない場合は、システムが指定するデフォルトの設定値が適用されます。アクティブ化されたインターフェイスには、常にサーバ・グループ・レベルの T_INTERFACE オブジェクトが関連付けられます。

インターフェイスに対する各レベルのコンフィギュレーション属性の指定はすべてオプションです。省略した場合は、システム定義のデフォルト値が設定され、実行時のサーバ・グループ・レベルの T_INTERFACE オブジェクトが作成されます。サーバが提供するインターフェイスは、サーバ・スケルトンのアクティブ化に使用する ICF ファイルで識別され、サーバの起動時にシステムによって自動的に宣言されます。

属性表

表 50TM_MIB(5): T_INTERFACE **クラス定義の属性表**

属性	使用法	タイプ	パーミッション	値	デフォルト値
TA_INTERFACENAME	r*	string	ru-r--r--	*string*[1..12]	N/A
TA_SRVGRP	r*	string	ru-r--r--	*string*[0..30]	N/A
TA_STATE	k	string	rwxr-xr--	GET: "{ACT | INA | SUS | PAR}" SET: "{NEW | INV | ACT | REA | SUS}"	N/A
TA_AUTOTRAN		string	rwxr-xr--	"{Y | N}"	"N"
TA_LOAD		long	rwxr-xr--	1 <= *num* < 32K	50 [1]
TA_PRIO		long	rwxr-xr--	1<= *num* < 101	50
TA_TIMEOUT		long	rwxr-xr--	0 <= *num*	0
TA_TRANTIME		long	rwxr-xr--	0 <= *num*	30
TA_FBROUTINGNAME		string	rwyr-yr--	*string*[1...15]	([2])
TA_LMID	k	string	R--R--R--	*LMID*	N/A
TA_NUMSERVERS		long	R--R--R---	0 <= *num*	N/A
TA_TPPOLICY		string	R--R--R--	"{method | transaction | process}"	N/A
TA_TXPOLICY		string	R--R--R--	"{always | never | optional | ignore}"	N/A
TA_NCOMPLETED	l	long	R-XR-XR--	0 <= *num*	N/A [3]
TA_NQUEUED	l	long	R--R--R--	0 <= *num*	N/A

表 50TM_MIB(5): T_INTERFACE **クラス定義の属性表 (続き)**

属性	使用法	タイプ	パーミッション	値	デフォルト値
(k)GET キー・フィールド (l) - ローカル・フィールド (r) - オブジェクトの作成に必要なフィールド (SET TA_STATE NEW) (*) - GET/SET キー、SET 操作では 1 つ以上必要					

1. グループ・レベルの T_INTERFACE オブジェクト (TA_SRVGRP != "") は、TA_INTERFACENAME 設定が一致するドメイン・レベルの T_INTERFACE オブジェクトが存在する場合には、そのオブジェクトからデフォルト値を決定します。ドメイン・レベルのオブジェクトが存在しない場合、またはドメイン・レベルのオブジェクトが作成中の場合は、一覧表示されているデフォルト値が適用されます。
2. 同じ TA_INTERFACENAME のすべての T_INTERFACE オブジェクトは、TA_FBROUTINGNAME 値が一致している必要があります。したがって、同じ TA_INTERFACENAME で一致するオブジェクトがない場合、新しく設定されるオブジェクトのデフォルト値は長さゼロの文字列 ("") になります。一致するオブジェクトがある場合は、デフォルト値 (唯一の有効値) は、既存の一致するオブジェクトに対して設定されている TA_FBROUTINGNAME の値になります。
3. TA_NCOMPLETED および TA_IMPLID (ローカル) では、T_DOMAIN MIB クラスで TA_LDBAL="Y" となっている必要があります。

属性の意味

TA_INTERFACENAME: *string*[1..128]

完全修飾されたインターフェイス名。インターフェイスのインターフェイス・リポジトリ ID です。この名前の形式は、インターフェイスのインプリメンテーションを生成する IDL に指定されたオプションによって異なります。詳細については、CORBA 2.1 仕様のセクション 7.6 を参照してください。

TA_SRVGRP: *string*[0..30]

サーバ・グループ名。サーバ・グループ名にはアスタリスク、カンマ、コロンは使用できません。この属性に対して明示的に指定された長さゼロの文字列を使用して、インターフェイスのドメイン・レベルのコンフィギュレーション情報および実行時情報を指定し照会します。このクラスのドメインおよびグループ・レベルのオブジェクトに関しては、その他の属性で説明した内容とは異なる制限事項および実行、指定方法がいくつかあります。

TA_STATE:

以下は、T_INTERFACE クラスの GET 値および SET TA_STATE 値の実行、指定方法です。グループおよびドメイン・レベルのオブジェクト間で実行、指定方法が異なる場合は、その違いを説明しています。

GET: "{ACTive | INActive | SUSpended | PARtitioned}"

GET 操作は、選択した T_INTERFACE オブジェクトのコンフィギュレーション情報を検索します。以下に示す状態は、GET 要求への応答で返される TA_STATE の意味を示します。これら以外の状態は返されません。

ACTive	T_INTERFACE オブジェクトは定義済みで、対応する T_IFQUEUE エントリの少なくとも 1 つは ACTive 状態です。 **注意：**グループ・レベル T_INTERFACE オブジェクトの場合、対応する T_IFQUEUE エントリは TA_INTERFACENAME および TA_SRVGRP 属性が一致するエントリです。ドメイン・レベルの T_INTERFACE オブジェクトの場合、対応する T_IFQUEUE エントリは TA_SRVGRP 値に関係なく TA_INTERFACENAME 属性が一致するエントリです。
INActive	T_INTERFACE オブジェクトが定義されていますが、ACTive と同等の状態の対応する T_IFQUEUE エントリは存在しません。
SUSpended	T_INTERFACE オブジェクトは定義済みで、対応するすべての T_IFQUEUE エントリのうち、ACTive 状態のものはなく、少なくとも 1 つが SUSpended 状態です。この状態は、パーミッションの決定においては ACTive と同等です。
PARtitioned	T_INTERFACE が定義済みで、対応するすべての T_IFQUEUE エントリのうち、 1. ACTive 状態のものはなく 2. SUSpended 状態のものはなく 3. 少なくとも 1 つは PARtitioned 状態にあります。この状態は、パーミッションの決定においては ACTive と同等です。

SET: "{NEW | INValid | ACTive | REActivate | SUSpended}"

SET 操作は、選択した T_INTERFACE オブジェクトのコンフィギュレーション情報および実行時情報を更新します。ドメイン・レベルの変更を行うと、複数のサーバ・グループに影響することがあります。また、複数のサーバがインターフェイスを提供している場合は、実行時の変更が複数のサーバに影響することがあります。以下に示す状態は、SET 要求で設定される TA_STATE の意味を示します。以下に示されていない状態は設定できません。

NEW	アプリケーションに対する T_INTERFACE オブジェクトを作成します。状態の変更は INValid 状態でのみ可能です。正常に終了すると、オブジェクトの状態は INActive になります。ドメイン・レベルの T_INTERFACE オブジェクトを作成すると、新しい値が明示的に指定されている場合は TA_FBROUTINGNAME 値がすべてリセットされ、同じ TA_INTERFACENAME 値を持つ既存のグループ・レベルのオブジェクトに影響します。その他のコンフィギュレーション属性の設定は、既存のグループ・レベル T_INTERFACE オブジェクトには影響しません。
INValid	アプリケーションに対する T_INTERFACE オブジェクトを削除します。状態の変更は、INActive 状態でのみ可能です。正常に終了すると、オブジェクトの状態は INValid になります。
ACTive	T_INTERFACE オブジェクトをアクティブにします。この状態をドメイン・レベルのオブジェクトに設定すると、ドメイン内で SUSpended 状態の対応するすべての T_IFQUEUE エントリがアクティブになります。この状態をグループ・レベルのオブジェクトに設定すると、インターフェイスを提供しているグループ内のサーバにのみ影響します。状態の変更は、SUSpended 状態でのみ可能です。正常に終了すると、オブジェクトの状態は ACTive になります。

REActivate	T_INTERFACE オブジェクトを再びアクティブにします。この状態をドメイン・レベルのオブジェクトに設定すると、ドメイン内で SUSpended 状態の対応するすべての T_IFQUEUE エントリがアクティブになります。この状態をグループ・レベルのオブジェクトに設定すると、インターフェイスを提供しているグループ内のサーバにのみ影響します。状態の変更は、ACTive 状態または SUSpended 状態でのみ可能です。正常に終了すると、オブジェクトの状態は ACTive になります。この状態では、グループ・レベルの T_INTERFACE オブジェクトを個別にアクティブ化しなくても、グループ・レベルで中断されている T_IFQUEUE エントリをグローバルにアクティブ化できます。
SUSpended	T_INTERFACE オブジェクトを中断します。この状態をドメイン・レベルのオブジェクトに設定すると、ドメイン内で ACTive 状態の対応するすべての T_IFQUEUE エントリが中断されます。この状態をグループ・レベルのオブジェクトに設定すると、インターフェイスを提供しているグループ内のサーバにのみ影響します。状態の変更は、ACTive 状態でのみ可能です。正常に終了すると、オブジェクトの状態は SUSpended になります。

制限事項 : インターフェイスの動的な宣言 (INActive または INValid から ACTive への状態変更) はサポートされていません。また、非宣言 (ACTive から INActive への状態変更) もサポートされていません。

TA_AUTOTRAN: "{Y | N}"

トランザクション・コンテキストの範囲外で実行された呼び出しに対して、自動的にトランザクションを開始するかどうかを示します。

制限事項 : この属性を実行時に更新しても、アクティブと同等の状態にある T_INTERFACE オブジェクトには反映されず、UBBCONFIG ファイル内のこの属性に指定した値が TA_TXPOLICY によってオーバライドされることがあります。TA_TXPOLICY の値が、always、never、または ignore のときは、次のようになります。

always	値 N は実行時には影響しません。Y に設定された場合と同様に動作します。
never	値 Y は実行時には影響しません。インターフェイスがトランザクションに含まれることはありません。
ignore	値 Y は実行時には影響しません。インターフェイスがトランザクションに含まれることはありません。

TA_LOAD: 1 <= *num* <= 32K

この T_INTERFACE オブジェクトは、システムに対する負荷を設定します。インターフェイスの負荷は、ロード・バランシングのために使用します。つまり、すでに負荷が大きいキューは、新規の要求ではあまり選択されません。

制限事項 : ドメイン・レベルのオブジェクトに対してこの属性を実行時に更新しても、同じインターフェイスの対応するグループ・レベルのオブジェクトには反映されません。

TA_PRIO: 1 <= *num* <= 101

この T_INTERFACE オブジェクトは、指定された優先順位でキューから取り出されます。複数のインターフェイス要求がサービス・キューで待機している場合、優先順位の高い要求から処理されます。

制限事項 : ドメイン・レベルのオブジェクトに対してこの属性を実行時に更新しても、同じインターフェイスの対応するグループ・レベルのオブジェクトには反映されません。

TA_TIMEOUT: 0 <= *num*

このインターフェイスの個々のメソッド呼び出しを処理する際の時間制限 (単位は秒)。このインターフェイスのメソッドの呼び出しを処理するサーバは、要求の処理が指定した時間制限値を超えると異常終了します。この属性を 0 に設定すると、サーバは異常終了しません。

制限事項 : ドメイン・レベルのオブジェクトに対してこの属性を実行時に更新しても、同じインターフェイスの対応するグループ・レベルのオブジェクトには反映されません。

TA_TRANTIME: 0 <= *num*

この T_INTERFACE オブジェクト用に自動的に開始されたトランザクションのトランザクション・タイムアウト値 (単位は秒)。インター

フェイスの T_INTERFACE:TA_AUTOTRAN 属性値が Y である場合に、トランザクション・モードでない要求を受信すると、トランザクションが自動的に開始されます。
制限事項 : ドメイン・レベルのオブジェクトに対してこの属性を実行時に更新しても、同じインターフェイスの対応するグループ・レベルのオブジェクトには反映されません。
注意：ドメイン・レベルのオブジェクトに対してこの値を実行時に更新すると、警告メッセージが表示されます。これは、更新後の値が以降のアプリケーション起動時のデフォルト設定にしか使用されないためです。

TA_FBROUTINGNAME: *string*[1..15]
このインターフェイスに関連付けられたファクトリ・ベースのルーティング基準。名前 FBROUTINGNAME を使用して、メッセージ・ベースのルーティングに他のルーティング基準を設定できるようにしておきます。このほうが、ROUTINGNAME に負荷をかけるより簡単です。
制限事項 : この属性は、ドメイン・レベルの T_INTERFACE オブジェクトに対してのみ設定されます (したがって、TA_SRVGRP は "")。

TA_LMID: *LMID*
アクティブ同等のグループ・レベル T_INTERFACE オブジェクトが関連付けられている現在の論理マシン。ドメイン・レベルのオブジェクトでは、ローカルな照会が実行されない限り (つまり、TA_FLAGS の MIB_LOCAL ビットを設定しない限り)、この属性は空白 ("") になります。ローカルの場合、複数のドメイン・レベルのオブジェクトが、各オブジェクトで示されているマシンから取得したローカル値とともに、同じインターフェイス (マシンごとに 1 つ) に対して返されます。

TA_NUMSERVERS: 0 <= *num*
このインターフェイスを提供する対応サーバの数。

TA_TPPOLICY: "{method | transaction | process}"
TP フレームワークの非アクティブ化方針。サーバの起動時にフレームワークに登録される方針を反映します。インターフェイスを登録する最初のサーバが、T_INTERFACE の値を設定します。この値は変更できません。

TA_TXPOLICY: "{optional | always | never | ignore}"
インターフェイスのトランザクション方針。この属性の設定は、TA_AUTOTRAN 属性の効果に影響します。詳細については、

TA_AUTOTRAN を参照してください。この属性は常に読み取り専用です。この属性は、開発者がサーバの構築時に設定し、サーバの起動時に登録されます。

TA_NCOMPLETED: 0 <= *num*

対応する T_IFQUEUE オブジェクトが最初に提供されてから、これまでに完了したインターフェイス・メソッド呼び出しの数。ドメイン・レベルのオブジェクトをローカルで照会 (TA_FLAGS MIB_LOCAL ビットを設定) すると、マシンごとに 1 つのオブジェクトが、そのマシンで指定されたインターフェイスの統計情報とともに返されます。

TA_NQUEUED: 0 <= *num*

このインターフェイスのキューに現時点で登録されている要求の数。ドメイン・レベルのオブジェクトをローカルで照会 (TA_FLAGS MIB_LOCAL ビットを設定) すると、マシンごとに 1 つのオブジェクトが、そのマシンで指定されたインターフェイスの統計情報とともに返されます。

インプリメンテーションのヒント

T_INTERFACE は、インターフェイスから BEA Tuxedo サービスへのマッピングです。MIB サーバは、対応する T_SERVICE オブジェクトの既存のロジックを呼び出すことにより、インターフェイスの get/set 操作の一部をインプリメントできます。

T_MACHINE クラスの定義

概要 T_MACHINE クラスは、特定のマシンに関係のあるアプリケーション属性を表します。これらの属性の値は、マシンの特性、マシンごとのサイズ、統計値、カスタマイズ・オプション、UNIX システムのファイル名などを表します。

属性表

TM_MIB(5): T_MACHINE クラス定義の属性表

属性	タイプ	パーミッション	値	デフォルト値
TA_LMID(r)(*) (注 1)	string	rU-r--r--	*string*[1..30]	N/A
TA_PMID(r)(*) (注 1)	string	rU-r--r--	*string*[1..30]	N/A
TA_TUXCONFIG(r)	string	rw-r--r--	*string*[2..256] (注 5)	N/A
TA_TUXDIR(r)	string	rw-r--r--	*string*[2..256] (注 6)	N/A
TA_APPDIR(r)	string	rw-r--r--	*string*[2..256] (注 6)	N/A
TA_STATE(k)	string	rwyr-yr--	GET: "{ACT | INA | PAR}" SET: "{NEW | INV | ACT | RAC | INA | FIN | CLE}"	N/A N/A
TA_UID	long	rw-r--r--	0 <= *num*	(注 2)
TA_GID	long	rw-r--r--	0 <= *num*	(注 2)
TA_ENVFILE	string	rwyr--r--	*string*[0..256] (注 6)	""
TA_PERM	long	rwyr--r--	0001 <= *num* <= 0777	(注 2)
TA_ULOGPFX	string	rwyr--r--	*string*[0..256] (注 6)	(注 3)
TA_TYPE	string	rw-r--r--	*string*[0..15]	""
TA_MAXACCESSERS	long	rw-r--r--	1 <= *num* < 32,768	(注 2)
TA_MAXCONV	long	rw-r--r--	0 <= *num* < 32,768	(注 2)

TM_MIB(5): T_MACHINE **クラス定義の属性表 (続き)**

属性	タイプ	パーミッション	値	デフォルト値
TA_MAXGTT	long	rw-r--r--	0 <= *num* < 32,768	(注 2)

TM_MIB(5): T_MACHINE **クラス定義の属性表 (続き)**

属性	タイプ	パーミッション	値	デフォルト値
TA_MAXWSCLIENTS	long	rw-r--r--	0 <= *num* < 32,768	0
TA_MAXACLCACHE	long	rw-r--r--	10 <= *num* <= 32,000	100
TA_TLOGDEVICE	string	rw-r--r--	*string*[0..256] (注 5)	""
TA_TLOGNAME	string	rw-r--r--	*string*[0..30]	"TLOG"
TA_TLOGSIZE	long	rw-r--r--	1 <= *num* < 2,049	100
TA_BRIDGE	string	rw-r--r--	*string*[0..78]	N/A
TA_BRTHREADS	string	rw-r--r--	"{Y | N}"	"N"
TA_NADDR	string	rw-r--r--	*string*[0..256] (注 6)	N/A
TA_NLSADDR	string	rw-r--r--	*string*[0..256] (注 6)	N/A
TA_FADDR	string	rw-r--r--	*string*[0..256] (注 6)	""
TA_FRANGE	long	rw-r--r--	1 <= *num* <= 65,535	1
TA_CMPLIMIT	string	rwyr-yr--	"*remote*[,*local*]"	MAXLONG
TA_TMNETLOAD	long	rwyr-yr--	0 <= *num* < 32,768	0
TA_SPINCOUNT	long	rwyr-yr--	0 <= *num*	0
TA_ROLE	string	r--r--r--	"{MASTER | BACKUP | OTHER}"	N/A
TA_MINOR	long	R--R--R--	1 <= *num*	N/A
TA_RELEASE	long	R--R--R--	1 <= *num*	N/A
TA_MINENCRYPTBITS	string	rwxrwx---	"{0 | 40 | 56 | 128}" (注 4)	"0"
TA_MAXENCRYPTBITS	string	rwxrwx---	"{0 | 40 | 56 | 128}" (注 4)	"128"
TA_MAXPENDINGBYTES	long	rw-r--r--	100000 <= *num* <= MAXLONG	2147483647

TM_MIB(5): T_MACHINE **クラス定義の属性表 (続き)**

属性	タイプ	パーミッション	値	デフォルト値
TA_SICACHEENTRIESMAX	string	rw-r--r--	"0"-"32767"	"500"
TA_SEC_PRINCIPAL_NAME	string	rwxr--r--	*string*[0..511]	""
TA_SEC_PRINCIPAL_LOCATION	string	rwxr--r--	*string*[0..511]	""
TA_SEC_PRINCIPAL_PASSVAR	string	rwxr--r--	*string*[0..511]	""
TA_SIGNATURE_REQUIRED	string	rwxr--r--	"{Y|N}"	"N"
TA_ENCRYPTION_REQUIRED	string	rwxr--r--	"{Y|N}"	"N"
T_MACHINE クラス： ローカル属性				
TA_CURACCESSERS	long	R--R--R--	0 <= *num* < 32,768	N/A
TA_CURCLIENTS	long	R--R--R--	0 <= *num* < 32,768	N/A
TA_CURCONV	long	R--R--R--	0 <= *num* < 32,768	N/A
TA_CURGTT	long	R--R--R--	0 <= *num* < 32,768	N/A
TA_CURRLOAD	long	R--R--R--	0 <= *num*	N/A
TA_CURWSCLIENTS	long	R--R--R--	0 <= *num* < 32,768	N/A
TA_HWACCESSERS	long	R--R--R--	0 <= *num* < 32,768	N/A
TA_HWCLIENTS	long	R--R--R--	0 <= *num* < 32,768	N/A
TA_HWCONV	long	R--R--R--	0 <= *num* < 32,768	N/A
TA_HWGTT	long	R--R--R--	0 <= *num* < 32,768	N/A
TA_HWWSCLIENTS	long	R--R--R--	0 <= *num* < 32,768	N/A
TA_NUMCONV	long	R-XR-XR--	0 <= *num*	N/A
TA_NUMDEQUEUE	long	R-XR-XR--	0 <= *num*	N/A
TA_NUMENQUEUE	long	R-XR-XR--	0 <= *num*	N/A
TA_NUMPOST	long	R-XR-XR--	0 <= *num*	N/A
TA_NUMREQ	long	R-XR-XR--	0 <= *num*	N/A

TM_MIB(5): T_MACHINE **クラス定義の属性表 (続き)**

属性	タイプ	パーミッション	値	デフォルト値
TA_NUMSUBSCRIBE	long	R-XR-XR--	0 <= *num*	N/A
TA_NUMTRAN	long	R-XR-XR--	0 <= *num*	N/A
TA_NUMTRANABT	long	R-XR-XR--	0 <= *num*	N/A
TA_NUMTRANCMT	long	R-XR-XR--	0 <= *num*	N/A
TA_PAGESIZE	long	R--R--R--	1 <= *num*	N/A
TA_SWRELEASE	string	R--R--R--	*string*[0..78]	N/A
TA_HWACLCACHE	long	R--R--R--	0 <= *num*	N/A
TA_ACLCACHEHITS	long	R--R--R--	0 <= *num*	N/A
TA_ACLCACHEACCESS	long	R--R--R--	0 <= *num*	N/A
TA_ACLFAIL	long	R--R--R--	0 <= *num*	N/A
TA_WKCOMPLETED	long	R--R--R--	0 <= *num*	N/A
TA_WKINITIATED	long	R--R--R--	0 <= *num*	N/A

(k)—GET キー・フィールド
(r)— オブジェクトの作成に必要なフィールド (SET TA_STATE NEW)
(*)—GET/SET キー、SET 操作では 1 つ以上必要

注 1 TA_LMID および TA_PMID は、このクラス内でそれぞれ一意である必要があります。SET 操作では、これらのいずれか一方のフィールドのみを使用します。両方を指定する場合は、どちらも同じオブジェクトを指している必要があります。
注 2 デフォルト設定は、T_DOMAIN クラスでこの属性に指定したのと同じ値になります。
注 3 デフォルト値は、このマシンの TA_APPDIR の後に /ULOG が続く文字列です。
注 4 リンク・レベルの暗号化値の 40 ビットは、下位互換性を維持するために提供されています。
注 5 BEA Tuxedo 8.0 以前では、この属性の文字列の最大長は 64 バイトです。
注 6 BEA Tuxedo 8.0 以前では、この属性の文字列の最大長は 78 バイトです。

属性の意味

TA_LMID: *string*[1..30]
論理マシン識別子。この識別子は、TM_MIB 定義の残りの部分で、アプリケーション・リソースを T_MACHINE オブジェクトにマップするための唯一の手段として使用します。

TA_PMID: *string*[1..30]
物理マシン識別子。この識別子は、指定したシステムで uname -n コマンドを実行した場合に返される UNIX システムのノード名と一致している必要があります。

TA_TUXCONFIG: *string*[2..256] (BEA Tuxedo 8.0 以前では最大 64 バイト)
バイナリ形式の BEA Tuxedo システムのコンフィギュレーション・ファイルが置かれているマシン上のファイルまたはデバイスの絶対パス名。管理者は、マスタ・マシンの TA_TUXCONFIG 属性値が示すファイルを 1 つのみ保持する必要があります。このファイルに格納される情報は、ほかの T_MACHINE オブジェクトがアクティブな状態になると、それらのオブジェクトに自動的に複製転送されます。環境内でのこの属性の使用方法については、後述の TA_ENVFILE を参照してください。

TA_TUXDIR: *string*[2..256] (BEA Tuxedo 8.0 以前では最大 78 バイト)
このマシン上での BEA Tuxedo システム・ソフトウェアの場所を示すディレクトリの絶対パス名。環境内でのこの属性の使用方法については、後述の TA_ENVFILE を参照してください。

TA_APPDIR: *string*[2..256] (BEA Tuxedo 8.0 以前では最大 78 バイト)
アプリケーション・ディレクトリの絶対パス名のリスト (個々のディレクトリの区切りにはコロンを使用)。1 番目のディレクトリは、このマシン上でブートされるすべてのアプリケーションと管理サーバのカレント・ディレクトリとして使用されます。アプリケーション・サーバを立ち上げる際には、リストに含まれるすべてのディレクトリがサーチされます。環境内でのこの属性の使用方法については、後述の TA_ENVFILE を参照してください。

TA_STATE:

GET: "{ACTive | INActive | PARtitioned}"
GET 操作は、選択した T_MACHINE オブジェクトのコンフィギュレーション情報および実行時情報を取得します。以下に示す状

態は、GET 要求への応答で返される TA_STATE の意味を示します。

ACTive	T_MACHINE オブジェクト (管理サーバ、すなわち DBBL、BBL、BRIDGE) が定義済みで、アクティブな状態にあることを示します。
INActive	T_MACHINE オブジェクトは定義済みで、アクティブでない状態にあることを示します。
PARtitioned	T_MACHINE オブジェクトは定義済みで、アクセス可能な掲示板にアクティブとして登録されていますが、現在はアクセスできないことを示します。この状態は、パーミッションの決定においては ACTive と同等です。

SET: "{NEW | INValid | ACTive | ReACtivate | INActive | ForceINactive | CLEaning}"

SET 操作は、選択した T_MACHINE オブジェクトのコンフィギュレーション情報および実行時情報を更新します。以下に示す状態は、SET 要求で設定される TA_STATE の意味を示します。以下に示されていない状態は設定できません。

NEW	アプリケーションの T_MACHINE オブジェクトを作成します。状態の変更は INValid 状態でのみ可能です。正常に終了すると、オブジェクトの状態は INActive になります。
unset	既存の T_MACHINE オブジェクトを変更します。この組み合わせは、ACTive 状態または INActive 状態でのみ可能です。正常終了した場合、オブジェクトの状態は変わりません。
INValid	アプリケーションの T_MACHINE オブジェクトを削除します。状態の変更は、INActive 状態でのみ可能です。正常終了すると、オブジェクトの状態は INValid になります。

ACTive	T_MACHINE オブジェクトをアクティブにします。必要な管理サーバ (DBBL、BBL、BRIDGE など) は指定したサイトで開始され、そのサイトで実行されるようにコンフィギュレーションされたアプリケーション・サーバも開始されます (TA_FLAGS 設定による制約が適用されます)。この状態遷移に対するパーミッションの決定に際しては、アクティブなオブジェクトのパーミッション (--x--x--x) が考慮されます。状態の変更は、INActive 状態でのみ可能です。正常に終了すると、オブジェクトの状態は ACTive になります。 マシンをアクティブにする際に個々のサーバの状態が必要な場合には、TMIB_NOTIFY TA_FLAG 値を使用します。
ReACtivate	T_MACHINE オブジェクトをアクティブにします。必要な管理サーバ (DBBL、BBL、BRIDGE など) は指定したサイトで開始され、そのサイトで実行されるようにコンフィギュレーションされたアプリケーション・サーバも開始されます (TA_FLAGS 設定による制約が適用されます)。この状態遷移に対するパーミッションの決定に際しては、アクティブなオブジェクトのパーミッション (--x--x--x) が考慮されます。状態の変更は、ACTive 状態または INActive 状態でのみ可能です。正常に終了すると、オブジェクトの状態は ACTive になります。 マシンを再びアクティブにする際に個々のサーバの状態が必要な場合には、TMIB_NOTIFY TA_FLAG 値を使用します。

INActive	T_MACHINE オブジェクトを非アクティブにします。必要な管理サーバ (BBL、BRIDGE など) は指定したサイトで停止し、そのサイトで実行されているアプリケーション・サーバも停止します (TA_FLAGS 設定による制約が適用されます)。状態の変更は、状態が ACTive で、指定したマシンのほかのアプリケーション・リソースがアクティブでない場合にのみ可能です。正常に終了すると、オブジェクトの状態は INActive になります。 マシンを非アクティブにする際に個々のサーバの状態が必要な場合には、TMIB_NOTIFY TA_FLAG 値を使用します。
ForceINactive	T_MACHINE オブジェクトを、アタッチされたクライアントとは無関係に非アクティブにします必要な管理サーバ (BBL、BRIDGE など) は指定したサイトで停止し、そのサイトで実行されているアプリケーション・サーバも停止します (TA_FLAGS 設定による制約が適用されます)。状態の変更は、ACTive 状態でのみ可能です。正常に終了すると、オブジェクトの状態は INActive になります。 マシンを非アクティブにする際に個々のサーバの状態が必要な場合には、TMIB_NOTIFY TA_FLAG 値を使用します。

CLEaning	指定したマシンおよびそのマシンに関係するクリーン・アップ / スキャニング処理を開始します。マシン上に DEAD 状態のクライアントやサーバが存在する場合は、この時点で検出されます。マシンがアプリケーションの MASTER サイトから分断されている場合は、グローバル掲示板のそのマシンのエントリは削除されます。この組み合わせは、アプリケーションが ACTive で、T_MACHINE オブジェクトが ACTive 状態または PARtitioned 状態にある場合にのみ可能です。分断されていないマシンに対する操作が正常に終了した場合、状態は変更されません。分断されているマシンに対する操作が正常に終了すると、オブジェクトの状態は INActive になります。

制限事項 : ForceINactive または INActive への状態の変更は、マスタ・マシン以外のマシンに対してのみ実行できます。マスタ・サイトの管理プロセスは、T_DOMAIN クラスを使用して非アクティブに変更します。

TA_UID: 0 <= *num*

このマシンの BEA Tuxedo システム・アプリケーションの管理者の UNIX システム・ユーザ識別子。tmboot(1)、tmshutdown(1)、tmadmin(1) などの管理コマンドは、このマシンで指定されたユーザとして実行する必要があります。このマシン上のアプリケーションや管理サーバは、このユーザとして起動されます。

制限事項 : UNIX システム固有の属性です。アプリケーションを実行しているプラットフォームが UNIX ベースでない場合、この属性は返されないことがあります。

TA_GID: 0 <= *num*

このマシンの BEA Tuxedo システム・アプリケーションの管理者の UNIX システム・グループ識別子。tmboot(1)、tmshutdown(1)、tmadmin(1) などの管理コマンドは、このマシンで指定されたグループの一員として実行する必要があります。このマシン上のアプリケーションや管理サーバは、このグループの一員として起動されます。

制限事項 : UNIX システム固有の属性です。アプリケーションを実行しているプラットフォームが UNIX ベースでない場合、この属性は返されないことがあります。

TA_ENVFILE: *string*[0..256] (BEA Tuxedo 8.0 以前では最大 78 バイト)
このマシンで実行しているクライアントやサーバの環境ファイル。無効なファイル名を指定すると環境に追加されません。*string* の値は環境内に配置されます。

起動時は、ローカル・サーバが tmboot(1) の環境を継承し、MASTER 上にないリモート・サーバが tlisten(1) の環境を継承します。また、対応する T_MACHINE オブジェクトの情報に基づいてサーバが起動されると、TUXCONFIG、TUXDIR、および APPDIR も環境に配置されます。PATH は環境内で次のように設定されます。

APPDIR:TUXDIR/bin:/bin:/usr/bin:*path*

path は、マシンの環境ファイルの最初の PATH= 行の値です。これ以降の PATH= 行はすべて無視されます。この PATH の値は、サーバの検索パスとして使用され、単純なパス名または相対パス名で指定されます。したがって、先頭はスラッシュではありません。
LD_LIBRARY_PATH は環境内で次のように設定されます。

APPDIR:TUXDIR/lib:/lib:/usr/lib:*lib*

lib は、マシンの環境ファイルの最初の LD_LIBRARY_PATH= 行の値です。これ以降の LD_LIBRARY_PATH= 行はすべて無視されます。

サーバの初期化時 (tpsvrinit() を呼び出す前) には、サーバがマシンとサーバの両方の ENVFILE ファイルの変数を読み取ってエクスポートします。変数がマシンとサーバの両方の ENVFILE ファイルに設定されている場合は、サーバの ENVFILE ファイルの値によってマシンの ENVFILE ファイルの値がオーバライドされます。ただし、PATH はオーバライドではなく追加されます。クライアントはマシンの ENVFILE ファイルのみを処理します。マシンとサーバの ENVFILE ファイルの処理では、*ident*= の形式でない行は無視されます。*ident* はアンダースコアまたは英字で始まり、アンダースコアまたは英数字のみを含めることができます。PATH= 行が見つかると、PATH が次のように設定されます。

APPDIR:TUXDIR/bin:/bin:/usr/bin:*path*

path は、マシンの環境ファイルの最初の PATH= 行の値です。これ以降の PATH= 行はすべて無視されます。マシンとサーバの両方の環境ファイルに PATH が存在する場合、*path* は *path1:path2* となります。*path1* はマシンの ENVFILE から読み取ったパス、*path2* はサーバの ENVFILE から読み取ったパスです。LD_LIBRARY_PATH= 行が見つかると、LD_LIBRARY_PATH が次のように設定されます。

APPDIR:TUXDIR/lib:/lib:/usr/lib:*lib*

lib は、マシンの環境ファイルの最初の LD_LIBRARY_PATH= 行の値です。これ以降の LD_LIBRARY_PATH= 行はすべて無視されます。TUXDIR、APPDIR、または TUXCONFIG をリセットしようとしても、対応する T_MACHINE 属性値と値が一致していない場合には無視されて警告メッセージが表示されます。制限事項 : アクティブなオブジェクトでこの属性を変更しても、実行中のサーバやクライアントには反映されません。

TA_PERM: 0001 <= *num* <= 0777

このマシン上に作成する共用メモリ掲示板に関連付ける UNIX システム・パーミッション。システムおよびアプリケーションのメッセージ・キューに対するデフォルトの UNIX システム・パーミッションです。

制限事項 : アクティブなオブジェクトでこの属性を変更しても、実行中のサーバやクライアントには反映されません。

UNIX システム固有の属性です。アプリケーションを実行しているプラットフォームが UNIX ベースでない場合、この属性は返されないことがあります。

TA_ULOGPFX: *string*[0..256] (BEA Tuxedo 8.0 以前では最大 78 バイト)

このマシン上の userlog() ファイルの絶対パス名の接頭辞。userlog() ファイルの名前は、TA_ULOGPFX 属性値に文字列 *.mmddyy* を付加することにより作成されます。*mmddyy* は、メッセージが生成された月、日、年を表します。このマシン上で実行しているクライアントやサーバが生成するアプリケーションやシステムの userlog() メッセージは、すべてこのファイルに書き込まれます。

制限事項 : アクティブなオブジェクトでこの属性を変更しても、実行中のサーバやクライアントには反映されません。

TA_TYPE: *string*[0..15]

マシン・タイプ。マシンを、類似のデータ表現を持つクラスに分類するために使用します。同じタイプのマシン間における通信では、データのエンコードは行われません。この属性にはどのような文字列値でも使用でき、その値は比較のためにのみ使用されます。アプリケーションが異種マシンのネットワークにまたがる場合や、コンパイラが異なる構造体表現を生成する場合は、別の TA_TYPE 属性を設定する必要があります。この属性のデフォルト値は長さゼロの文字列です。これは、TA_TYPE 属性値に長さゼロの文字列を持つすべてのマシンと一致します。

TA_MAXACCESSERS: 1 <= *num* < 32,768

このマシンの掲示板に同時に接続できるクライアントおよびサーバの最大数。指定しない場合、T_DOMAIN クラスで指定した TA_MAXACCESSERS の値がデフォルト値になります。

この数には、BBL、restartsrv、cleanupsrv、tmshutdown()、tmadmin() などのシステム管理プロセスを含める必要はありません。ただし、DBBL、すべてのブリッジ・プロセス、すべてのシステム提供サーバ・プロセスとアプリケーション・サーバ・プロセス、およびこのサイトで使用する可能性があるクライアント・プロセスはこの数に含めますシステム提供のサーバには、AUTHSVR、TMQUEUE、TMQFORWARD、TMUSREVT、TMSYSEVT、TMS (T_GROUP TA_TMSNAME 属性を参照)、TMS_QM、GWTDOMAIN、WSL などがあります。アプリケーションがこのサイトでワークステーション・リスナ (WSL) を起動する場合は、起動する WSL と使用する可能性があるワークステーション・ハンドラ (WSH) の両方をこの数に含める必要があります。

BEA Tuxedo リリース 7.1 より前 (6.5 以前) では、ユーザ・ライセンス数をチェックする仕組みにおいて、アプリケーションの TA_MAXACCESSERS 属性と TA_MAXSERVERS 属性 (T_DOMAIN:TA_MAXSERVERS 属性を参照) が使用されていました。つまり、アプリケーションで実行中の 1 台以上のマシンの TA_MAXACCESSERS の数と、特定のマシンの TA_MAXACCESSERS の数の合計が、TA_MAXSERVERS の数とユーザ・ライセンス数の合計より大きい場合、マシンを起動することはできませんでした。したがって、アプリケーションの TA_MAXACCESSERS パラメータには、TA_MAXSERVERS の数とユーザ・ライセンス数の合計か、またはそれより小さい値を指定しなければなりませんでした。

BEA Tuxedo のリリース 7.1 以降では、アプリケーションに設定されているユーザ・ライセンスの数と、現在使用されているユーザ・ライセンスの数に基づいて、ライセンスのチェックが行われます。すべてのユーザ・ライセンスが使用中になると、アプリケーションに新しいクライアントが参加することはできなくなります。

TA_MAXCONV: 0 <= *num* < 32,768

このマシン上のクライアントとサーバが同時に関与できる会話の最大数。指定しない場合、T_DOMAIN クラスで指定した TA_MAXCONV の値がデフォルト値になります。1 つのサーバで、最大 64 個の会話を同時に行うことができます。

TA_MAXGTT: 0 <= *num* < 32,768

このマシンが同時に関与できるグローバル・トランザクションの最大数値を指定しない場合、デフォルトで T_DOMAIN クラスの値が指定されます。

TA_MAXWSCLIENTS: 0 <= *num* < 32,768

ワークステーション・クライアント (ネイティブ・クライアントでないクライアント) 用に予約するこのマシン上のアクセサ・エントリの数。TA_MAXWSCLIENTS を指定しない場合は、デフォルトで 0 が設定されます。

ここに指定する値は、TA_MAXACCESSERS 属性で指定したアクセサ・スロットの総数の一部になります。つまり、TA_MAXWSCLIENTS 用に予約したアクセサ・スロットは、このマシン上の別のクライアントおよびサーバでは使用できません。この値を TA_MAXACCESSERS より大きな値に設定した場合はエラーになります。

TA_MAXWSCLIENTS 属性を使用するのは、BEA Tuxedo システムの Workstation 機能を使用する場合のみです。ワークステーション・クライアントからシステムへのアクセスは、BEA Tuxedo システムに組み込まれている代理プロセス、つまりワークステーション・ハンドラによって多重化されます。そのため、この属性を適切に設定すると、プロセス間通信 (IPC) リソースを節約できます。

TA_MAXACLCACHE: 10 <= *num* <= 32,000

TA_SECURITY が ACL または MANDATORY_ACL に設定されている場合のキャッシュ内の ACL 用エントリ数この属性を適切に設定すると、共

用メモリ上のリソースを節約しながら、ACL をチェックするためのディスク・アクセスの回数を減らすことができます。

TA_TLOGDEVICE: *string*[0..256] (BEA Tuxedo 8.0 以前では最大 64 バイト)
このマシンの DTP トランザクション・ログを保持するための BEA Tuxedo のファイル・システムを持つデバイス (raw スライス) または UNIX システム・ファイル。DTP トランザクション・ログは、BEA Tuxedo システムの VTOC テーブルとしてデバイスに格納されています。このデバイスまたはファイルは、このマシンの TA_TUXCONFIG 属性で指定したデバイスまたはファイルと同じでもかまいません。

TA_TLOGNAME: *string*[0..30]
このマシンの DTP トランザクション・ログの名前。1 つの TLOGDEVICE に複数の DTP トランザクション・ログがある場合、それぞれの名前は一意でなければなりません。TA_TLOGNAME は、DTP トランザクション・ログ・テーブルが作成される TA_TLOGDEVICE 上のどのテーブル名とも異なっている必要があります。

TA_TLOGSIZE: 1 <= *num* < 2,049
このマシンの DTP トランザクション・ログのサイズ (ページ単位)。TA_TLOGSIZE 属性の値に対しては、TA_TLOGDEVICE 属性で指定した BEA Tuxedo ファイル・システムの空き容量に基づく制約が適用されます。

TA_BRIDGE: *string*[0..78]
この論理マシンのブリッジ・プロセスがネットワーク・アクセスに使用するデバイスの名前。この名前は、ネットワーク対応のアプリケーションに TLI ベースの BEA Tuxedo システム・バイナリを通じて参加する際に必要になります。この属性は、ソケット・ベースの BEA Tuxedo システム・バイナリには必要ありません。

TA_BRTHREADS: "{Y | N}"
この論理マシンのブリッジ・プロセスを、マルチスレッド実行 ("Y") またはシングルスレッド実行 ("N") にコンフィギュレーションします。デフォルトは "N" です。この属性は、BEA Tuxedo 8.1 以降が動作するアプリケーションにのみ適用されます。

TA_BRTHREADS を "Y" に設定しても、CPU が複数あるマシンにしか影響しません。ただし、複数の CPU がなくても、TA_BRTHREADS を "Y" に設定することは可能です。

ローカル・マシンで TA_BRTHREADS を "Y" に設定し、リモート・マシンで TA_BRTHREADS を "N" (デフォルト) に設定することは可能ですが、マシン間のスループットがシングルスレッドのブリッジ・プロセスより大きくなることはありません。

シングルスレッド実行またはマルチスレッド実行にコンフィギュレーションしたブリッジ・プロセスは、BEA Tuxedo の以前のリリース、WebLogic Enterprise の BEA Tuxedo リリース 8.0 以前、および WebLogic Enterprise リリース 5.1 以前で実行するブリッジ・プロセスとの相互運用が可能です。通常、スレッド化されたブリッジはスレッド化されていないブリッジと相互運用できます。これは、スレッド化によって外部機能や動作が変わることはないためです。

注記　BRTHREADS=Y に設定し、ブリッジ環境に TMNOTHREADS=Y が含まれている場合、ブリッジはスレッド・モードで起動し、ブリッジが TMNOTHREADS の設定を無視したことを示す警告メッセージがログに記録されます。TMNOTHREADS 環境変数は、リリース 8.0 の BEA Tuxedo 製品で新たに追加されました。

TA_NADDR: *string*[0..256] (BEA Tuxedo 8.0 以前では最大 78 バイト)

論理マシン上のブリッジ・プロセスが接続指示受け付けアドレスとして使用する完全なネットワーク・アドレスを指定します。ブリッジに対する接続指示受け付けアドレスは、アプリケーションに参加しているほかのブリッジ・プロセスの通信手段となります。論理マシンがネットワーク・アプリケーションに参加する場合、つまり T_DOMAIN:TA_OPTIONS 属性で LAN オプションを指定した場合は、この属性を設定する必要があります。

string の形式が "0x*hex-digits*" または "\\x*hex-digits*" の場合、文字数が偶数の有効な 16 進数値を含める必要があります。このような形式の文字列は、指定された文字列の 16 進数表現を含む文字配列に内部変換されます。TCP/IP アドレスの場合は、

"*//hostname:port*"

または

"*//#.#.#.#:port*"

のいずれかの形式を使用します。

TA_NLSADDR: *string*[0..256] (BEA Tuxedo 8.0 以前では最大 78 バイト)

この論理マシンが示すノードでネットワークにサービスを提供する tlisten(1) プロセスが使用するネットワーク・アドレス。このネットワーク・アドレスは、前述の TA_NADDR で指定した形式と同じ形式になります。

論理マシンがネットワーク・アプリケーションに参加する場合、つまり T_DOMAIN:TA_OPTIONS 属性で LAN オプションを指定した場合は、この属性を設定する必要があります。

TA_FADDR: *string*[0..256] (BEA Tuxedo 8.0 以前では最大 78 バイト)

tmboot、tmloadcf、BRIDGE などのローカル・プロセスが、アウトバウンド接続を確立する前にバインドできる完全なネットワーク・アドレスを指定します。このアドレスには、TCP/IP アドレスを指定する必要があります。この属性および TA_FRANGE 属性は、TCP/IP ポート (プロセスがアウトバウンド接続を行う前にバインドするときのバインド先) の範囲を指定します。この属性が NULL または空文字列に設定されている場合、オペレーティング・システムはバインド先のローカル・ポートをランダムに選択します。

string の形式が "0x*hex-digits*" の場合、文字数が偶数の有効な 16 進数値を含める必要があります。このような形式の文字列は、指定された文字列の 16 進数表現を含む文字配列に内部変換されます。

TCP/IP アドレスの場合は、以下のいずれかの形式になります。

- "//*hostname*:*port*"
- "//#.#.#.#:*port*"

TA_FRANGE: 1<= *num* <= 65,535

ローカル・プロセスがアウトバウンド接続を確立する前にバインドする TCP/IP ポートの範囲を指定します。範囲のベースとなるアドレスは、TA_FADDR 属性で指定します。

TA_CMPLIMIT: "*remote*[,*local*]"

リモート・トラフィックおよびオプションのローカル・トラフィックの圧縮が発生するメッセージ・サイズのしきい値。*remote* と *local* には、負の数でない数値、またはマシンに設定された最大の long 値に動的に変換される MAXLONG という文字列をセットできます。*remote* のみを設定した場合、*local* はデフォルトで MAXLONG になります。

制限事項 : BEA Tuxedo リリース 4.2.2 以前を実行しているアクティブなサイトでは、この属性値は T_MACHINE オブジェクトの一部ではなくなります。ただし、サイトのリリースは実行時まで不明であるため、アクティブでないオブジェクトに対してはこの属性を設定してアクセスできます。BEA Tuxedo リリース 4.2.2 以前を使用しているサイトがアクティブになると、設定された値は使用されなくなります。

TA_TMNETLOAD: 0 <= *num* < 32,768

このマシンのロード・バランシングの間に評価されるリモート・サービスに追加するサービス負荷。

制限事項 : BEA Tuxedo リリース 4.2.2 以前を実行しているアクティブなサイトでは、この属性値は T_MACHINE オブジェクトの一部ではなくなります。ただし、サイトのリリースは実行時まで不明であるため、アクティブでないオブジェクトに対してはこの属性を設定してアクセスできます。BEA Tuxedo リリース 4.2.2 以前を使用しているサイトがアクティブになると、設定された値は使用されなくなります。

TA_SPINCOUNT: 0 <= *num*

このマシンで、チケット入手前のユーザ・レベル・セマフォ・アクセスに対して使用するスピン・カウント。デフォルト設定は、各マシンの BEA Tuxedo システムのバイナリに組み込まれています。この属性を使用すると、これらのデフォルト設定を実行時に調節できます。この属性値をゼロに設定すると、スピン・カウントを各サイトの組み込みのデフォルト設定にリセットできます。また、この属性または UBBCONFIG ファイルで値が設定されていない場合、システムは TMSPINCOUNT 環境変数を使用します。

制限事項 : BEA Tuxedo リリース 4.2.2 以前を実行しているアクティブなサイトでは、この属性値は T_MACHINE オブジェクトの一部ではなくなります。ただし、サイトのリリースは実行時まで不明であるため、アクティブでないオブジェクトに対してはこの属性を設定してアクセスできます。BEA Tuxedo リリース 4.2.2 以前を使用しているサイトがアクティブになると、設定された値は使用されなくなります。

TA_ROLE: "{MASTER | BACKUP | OTHER}"

アプリケーションにおけるこのマシンのロール。"MASTER" はこのマシンをマスタ・マシンとして使用することを示し、"BACKUP" はこのマシンをバックアップ用のマスタ・マシンとして使用することを示しま

す。"OTHER" は、このマシンがマスタ・マシンでもバックアップ用のマスタ・マシンでもないことを示します。

TA_MINOR: 1 <= *num*
このマシンの BEA Tuxedo システム・プロトコルのマイナー・リリース番号。

TA_RELEASE: 1 <= *num*
このマシンの BEA Tuxedo システム・プロトコルのメジャー・リリース番号。この番号は、同じマシンの TA_SWRELEASE とは異なる場合があります。

TA_MINENCRYPTBITS: "{0 | 40 | 56 | 128}"
このマシンへのネットワーク・リンクを確立する際に必要な暗号化の最低レベルを指定します。0 は暗号化が行われないことを示し、40、56、および 128 は暗号化キーの長さをビット単位で指定します。この最小レベルの暗号化が一致しない時は、リンクの確立は失敗します。デフォルトは 0 です。

注記　リンク・レベルの暗号化の値 40 は、下位互換性を維持するために提供されています。

制限事項 : この属性を変更しても、確立済みのネットワーク・リンクには反映されません。

TA_MAXENCRYPTBITS: "{0 | 40 | 56 | 128}"
ネットワーク・リンクを確立する際に調整できる暗号化の最高レベルを指定します。0 は暗号化が行われないことを示し、40、56、および 128 は暗号化キーの長さをビット単位で指定します。デフォルトは 128 です。

注記　リンク・レベルの暗号化の値 40 は、下位互換性を維持するために提供されています。

制限事項 : この属性を変更しても、確立済みのネットワーク・リンクには反映されません。

TA_MAXPENDINGBYTES: 100000 <= *num* <= MAXLONG
ブリッジ・プロセスで送信されるのを待つメッセージに対して割り当てられる領域の上限を指定します。

TA_SICACHEENTRIESMAX: "0"-"32767"
このマシンが保持するサービスおよびインターフェイスのキャッシュ・エントリの数。指定しない場合、値は "500" に設定されます。値を "0" にすると、このマシンではサービス・キャッシュが使用されなくなります。

TA_SEC_PRINCIPAL_NAME: *string*[0..511]
BEA Tuxedo 7.1 以降が動作するアプリケーションで認証用に使用されるセキュリティ・プリンシパル名。この属性の最大文字数は、文字列の最後を表す NULL 文字列を除いて 511 文字です。この属性に指定するプリンシパル名は、このマシンで実行される 1 つ以上のシステム・プロセスの識別子として使用されます。

TA_SEC_PRINCIPAL_NAME は、コンフィギュレーションの階層のうち、T_DOMAIN クラス、T_MACHINE クラス、T_GROUP クラス、および T_SERVER クラスの 4 つのレベルのどこでも指定できます。特定のコンフィギュレーション・レベルでのプリンシパル名は、下位レベルでオーバライドできます。TA_SEC_PRINCIPAL_NAME がどのレベルでも指定されていない場合、アプリケーションのプリンシパル名にはこのドメインの TA_DOMAINID 文字列がデフォルトで設定されます。

TA_SEC_PRINCIPAL_NAME のほかにも、TA_SEC_PRINCIPAL_LOCATION と TA_SEC_PRINCIPAL_PASSVAR という属性があります。後の 2 つの属性は、アプリケーション起動時に、BEA Tuxedo 7.1 以降で動作するシステム・プロセスに対して復号化キーのオープンする処理に関係する属性です。特定のレベルで TA_SEC_PRINCIPAL_NAME のみが指定されている場合には、それ以外の 2 つの属性に長さゼロの NULL 文字列が設定されます。

TA_SEC_PRINCIPAL_LOCATION: *string*[0..511]
TA_SEC_PRINCIPAL_NAME に指定したプリンシパルの復号化 (秘密) キーを格納するファイルまたはデバイスの位置。この属性の最大文字数は、文字列の最後を表す NULL 文字列を除いて 511 文字です。

TA_SEC_PRINCIPAL_LOCATION は、コンフィギュレーションの階層のうち、T_DOMAIN クラス、T_MACHINE クラス、T_GROUP クラス、および T_SERVER クラスの 4 つのレベルのどこでも指定できます。この属性は、どのレベルで指定する場合でも TA_SEC_PRINCIPAL_NAME 属性と対になっている必要があり、それ以外の場合には無視されます

(TA_SEC_PRINCIPAL_PASSVAR はオプションです。この属性が指定されていない場合、システムによって長さゼロの NULL 文字列が設定されます)。

TA_SEC_PRINCIPAL_PASSVAR: *string*[0..511]

TA_SEC_PRINCIPAL_NAME に指定したプリンシパルのパスワードを格納する変数。この属性の最大文字数は、文字列の最後を表す NULL 文字列を除いて 511 文字です。

TA_SEC_PRINCIPAL_PASSVAR は、コンフィギュレーションの階層のうち、T_DOMAIN クラス、T_MACHINE クラス、T_GROUP クラス、および T_SERVER クラスの 4 つのレベルのどこでも指定できます。この属性は、どのレベルで指定する場合でも TA_SEC_PRINCIPAL_NAME 属性と対になっている必要があり、それ以外の場合には無視されます (TA_SEC_PRINCIPAL_LOCATION はオプションです。この属性が指定されていない場合、システムによって長さゼロの NULL 文字列が設定されます)。

初期化時は、TA_SEC_PRINCIPAL_PASSVAR に設定した復号化キーの各パスワードを管理者が入力する必要があります。管理者が入力したパスワードはシステム側で自動的に暗号化され、それぞれが対応するパスワード変数に割り当てられます。

TA_SIGNATURE_REQUIRED: "{Y | N}"

"Y" に設定すると、このマシンで実行するすべてのプロセスで、その入力メッセージ・バッファのデジタル署名が必要となります。指定しない場合、デフォルト値の "N" が設定されます。この属性は、BEA Tuxedo 7.1 以降が動作するアプリケーションにのみ適用されます。

TA_SIGNATURE_REQUIRED は、コンフィギュレーションの階層のうち、T_DOMAIN クラス、T_MACHINE クラス、T_GROUP クラス、および T_SERVICE クラスの 4 つのレベルのどこでも指定できます。特定のレベルで SIGNATURE_REQUIRED に Y を設定すると、下位レベルで動作するすべてのプロセスに署名が必要となります。

TA_ENCRYPTION_REQUIRED: "{Y | N}"

"Y" に設定すると、このマシンで実行するすべてのプロセスで暗号化された入力メッセージ・バッファが必要となります。指定しない場

合、デフォルト値の "N" が設定されます。この属性は、BEA Tuxedo 7.1 以降が動作するアプリケーションにのみ適用されます。

TA_ENCRYPTION_REQUIRED は、コンフィギュレーションの階層のうち、T_DOMAIN クラス、T_MACHINE クラス、T_GROUP クラス、および T_SERVICE クラスの 4 つのレベルのどこでも指定できます。特定のレベルで TA_ENCRYPTION_REQUIRED に "Y" を設定すると、下位レベルで動作するすべてのプロセスに暗号化が必要となります。

TA_CURACCESSERS: 0 <= *num* < 32,768
: 現在このマシンに直接、またはワークステーション・ハンドラ経由でアクセスしているクライアントとサーバの数。

TA_CURCLIENTS: 0 <= *num* < 32,768
: 現在このマシンにログインしているネイティブ・クライアントおよびワークステーション・クライアントの数。

TA_CURCONV: 0 <= *num* < 32,768
: このマシンに参加者が存在するアクティブな会話の数。

TA_CURGTT: 0 <= *num* < 32,768
: このマシンで使用中のトランザクション・テーブル・エントリの数。

TA_CURRLOAD: 0 <= *num*
: このマシンでキューに登録されている現在のサービス負荷。制限事項 : T_DOMAIN:TA_LDBAL 属性が "N" に設定されているか、T_DOMAIN:TA_MODEL 属性が "MP" に設定されている場合、FML32 NULL 値が返されます (0)。

TA_CURWSCLIENTS: 0 <= *num* < 32,768
: 現在このマシンにログインしているワークステーション・クライアントの数。

TA_HWACCESSERS: 0 <= *num* < 32,768
: このマシンに直接、またはワークステーション・ハンドラ経由でアクセスするクライアントとサーバの最大数。

TA_HWCLIENTS: 0 <= *num* < 32,768
: このマシンにログインするネイティブ・クライアントおよびワークステーション・クライアントの最大数。

TA_HWCONV: 0 <= *num* < 32,768
　　このマシンに参加者が存在するアクティブな会話の最大数。

TA_HWGTT: 0 <= *num* < 32,768
　　このマシンで使用中のトランザクション・テーブル・エントリの最大数。

TA_HWWSCLIENTS: 0 <= *num* < 32,768
　　現在このマシンにログインするワークステーション・クライアントの最大数。

TA_NUMCONV: 0 <= *num*
　　このマシンから実行された tpconnect() 操作の数。

TA_NUMDEQUEUE: 0 <= *num*
　　このマシンから実行された tpdequeue() 操作の数。

TA_NUMENQUEUE: 0 <= *num*
　　このマシンから実行された tpenqueue() 操作の数。

TA_NUMPOST: 0 <= *num*
　　このマシンから実行された tppost() 操作の数。

TA_NUMREQ: 0 <= *num*
　　このマシンから実行された tpacall() 操作または tpcall() 操作の数。

TA_NUMSUBSCRIBE: 0 <= *num*
　　このマシンから実行された tpsubscribe() 操作の数。

TA_NUMTRAN: 0 <= *num*
　　このマシンから開始 (tpbegin()) されたトランザクションの数。

TA_NUMTRANABT: 0 <= *num*
　　このマシンからアボート (tpabort()) されたトランザクションの数。

TA_NUMTRANCMT: 0 <= *num*
　　このマシンからコミット (tpcommit()) されたトランザクションの数。

TA_PAGESIZE: 1 <= *num*
　　このマシンで使用するディスクのページ・サイズ。

TA_SWRELEASE: *string*[0..78]
: このマシンのバイナリのソフトウェア・リリース。バイナリが BEA Tuxedo システム・マスタ・バイナリでない場合は長さゼロの文字列になります。

TA_HWACLCACHE: 0 <= *num*
: ACL キャッシュで使用するエントリの最大数。

TA_ACLCACHEHITS: 0 <= *num*
: 「ヒット」した (エントリがキャッシュ内に存在していた) ACL キャッシュへのアクセスの数。

TA_ACLCACHEACCESS: 0 <= *num*
: ACL キャッシュへのアクセスの数。

TA_ACLFAIL: 0 <= *num*
: アクセス制御違反になった ACL キャッシュへのアクセスの数。

TA_WKCOMPLETED: 0 <= *num*
: このマシンで実行しているサーバがキューから取り出して正常に処理したサービス負荷の合計。この属性は long の最大値を超えるとゼロに戻って再スタートします。長時間実行しているアプリケーションでは、この値が一巡していることがありますので注意してください。

TA_WKINITIATED: 0 <= *num*
: このマシンで実行しているクライアントまたはサーバがキューに登録したサービス負荷の合計。この属性は long の最大値を超えるとゼロに戻って再スタートします。長時間実行しているアプリケーションでは、この値が一巡していることがありますので注意してください。

制限事項

SHM モード (T_DOMAIN:TA_MODEL 属性を参照) のアプリケーションは、T_MACHINE オブジェクトを 1 つしか持つことができません。LAN オプション (T_DOMAIN:TA_MODEL 属性を参照) を設定した MP モード (T_DOMAIN:TA_OPTIONS 属性を参照) のアプリケーションは、T_DOMAIN:TA_MAXMACHINES 属性で定義された T_MACHINE オブジェクトをコンフィギュレーション可能な最大数まで持つことができます。このクラスの属性の多くは、アプリケーションがサイト上で非アクティブなときにしか調節できません。最低限アクティブなアプリケーションにおいても、少なくともマスタ・マシンはアクティブでなければならないため、マスタ・マシン・オブジェクトについては、ATMI インターフェイス・ルーチンをアプリケーションの管理に使用することはできません。そのため、起動されていないア

プリケーションをコンフィギュレーションするための手段として `tpadmcall()` という関数が用意されています。この関数を使用すると、マスタ・マシンのこれらの属性を設定できます。

T_MSG クラスの定義

概要 T_MSG クラスは、BEA Tuxedo システムが管理する UNIX システム・メッセージ・キューの実行時属性を表します。

属性表

表 51TM_MIB(5): T_MSG クラス定義の属性表

属性 (注 1)	タイプ	パーミッション	値	デフォルト値
TA_LMID(k)	string	R--R--R--	*LMID*	N/A
TA_MSGID(k)	long	R--R--R--	1 <= *num*	N/A
TA_STATE(k)	string	R--R--R--	GET: "ACT" SET:N/A	N/A N/A
TA_CURTIME	long	R--R--R--	1 <= *num*	N/A
TA_MSG_CBYTES	long	R--R--R--	1 <= *num*	N/A
TA_MSG_CTIME	long	R--R--R--	1 <= *num*	N/A
TA_MSG_LRPID	long	R--R--R--	1 <= *num*	N/A
TA_MSG_LSPID	long	R--R--R--	1 <= *num*	N/A
TA_MSG_QBYTES	long	R--R--R--	1 <= *num*	N/A
TA_MSG_QNUM	long	R--R--R--	1 <= *num*	N/A
TA_MSG_RTIME	long	R--R--R--	1 <= *num*	N/A
TA_MSG_STIME	long	R--R--R--	1 <= *num*	N/A
(k)—GET key field				

注 1 T_MSG クラスのすべての属性はローカル属性です。

属性の意味 TA_LMID:*LMID*
論理マシン識別子。

TA_MSGID: 1 <= *num*

UNIX システムのメッセージ・キューの識別子。制限事項 : UNIX システム固有の属性です。アプリケーションを実行しているプラットフォームが UNIX ベースでない場合、この属性は返されないことがあります。

TA_STATE:

GET: "{ACTive}"

GET 操作は、選択した T_MSG オブジェクトの実行時情報を検索します。以下に示す状態は、GET 要求への応答で返される TA_STATE の意味を示します。

ACTive	T_MSG オブジェクトがアクティブであることを示します。これは、関連のある T_MACHINE オブジェクトがアクティブであることを意味します。

SET:

SET 操作は、このクラスでは使用できません。

TA_CURTIME: 1 <= *num*

T_BRIDGE:T_MSG で time(2) システム・コールから返される 1970 年 1 月 1 日の 00:00:00 UTC から現在までの時間 (単位は秒)。この属性は、T_MSG:TA_?TIME 属性値からの経過時間を算出するために使用できます。

TA_MSG_CBYTES: 1 <= *num*

キュー上の現在のバイト数。

TA_MSG_CTIME: 1 <= *num*

キューに関連付けられた *msqid_ds* 構造体のメンバを最後に変更した msgctl(2) 操作の時間。

TA_MSG_LRPID: 1 <= *num*

キューからの読み取りを最後に実行したプロセスの識別子。

TA_MSG_LSPID: 1 <= *num*

キューへの書き込みを最後に実行したプロセスの識別子。

TA_MSG_QBYTES: 1 <= *num*
キュー上の最大バイト数。

TA_MSG_QNUM: 1 <= *num*
キュー上の現在のメッセージ数。

TA_MSG_RTIME: 1 <= *num*
キューからの読み取りを最後に実行してから経過した時間。

TA_MSG_STIME: 1 <= *num*
キューへの書き込みを最後に実行してから経過した時間。

制限事項 このクラスは UNIX システムに固有のクラスであり、UNIX をインプリメントしていない BEA Tuxedo システムではサポートされません。

T_NETGROUP クラスの定義

概要 T_NETGROUP クラスは、ネットワーク・グループのアプリケーション属性を表します。ネットワーク・グループは LMID のグループで、T_NETMAP クラスに定義された TA_NADDR ネットワーク・アドレスで通信できます。

属性表

表 52TM_MIB(5): T_NETGROUP **クラス定義の属性表**

属性	タイプ	パーミッション	値	デフォルト値
TA_NETGROUP(r)(*)	string	rU-------	*string*[1..30]	"DEFAULTNET"
TA_NETGRPNO(r)(*)	long	rU-------	1 <= *num* < 8192	N/A
TA_STATE(k)	string	rw-r--r--	GET: "VAL" SET: "{NEW | INV}"	N/A N/A
TA_NETPRIO(*)	long	rwyrw----	1 <= *num* < 8,192	100

(k)—GET キー・フィールド
(r)— オブジェクトの作成に必要なフィールド (SET TA_STATE NEW)
(*)—GET/SET キー、SET 操作では 1 つ以上必要

属性の意味

TA_NETGROUP: *string*[1..30]
ネットワーク・グループの論理名。グループ名は表示可能な文字列で、シャープ、カンマ、コロン、および改行文字は使用できません。

TA_NETGRPNO: 1 <= *num* <= 8192
ネットワーク・グループに関連付けるグループ識別子。

TA_STATE:

GET: "{VALid}"
GET 操作は、選択した T_NETGROUP オブジェクトのコンフィギュレーション情報を検索します。以下に示す状態は、GET 要求への応答で返される TA_STATE の意味を示します。

VALid	T_NETGROUP オブジェクトは定義済みで、非アクティブな状態です。これがこのクラスの唯一の有効な状態です。NETGROUP が ACTive 状態になることはありません。

SET: "{NEW | INValid}"

SET 操作は、選択した T_NETGROUP オブジェクトのコンフィギュレーション情報を更新します。以下に示す状態は、SET 要求で設定される TA_STATE の意味を示します。以下に示されていない状態は設定できません。

NEW	アプリケーションの T_NETGROUP オブジェクトを作成します。状態の変更は INValid 状態でのみ可能です。正常終了すると、オブジェクトの状態は VALid になります。
unset	既存の T_NETGROUP オブジェクトを変更します。VALid 状態の場合のみ変更できます。正常終了した場合、オブジェクトの状態は変わりません。
INValid	アプリケーションから T_NETGROUP オブジェクトを削除します。状態変更ができるのは、状態が VALid で、このネットワーク・グループ・オブジェクトをキーとする T_NETMAP クラスにオブジェクトが存在しない場合に限ります。正常終了すると、オブジェクトの状態は INValid になります。

TA_NETPRIO: 1 <= *num* < 8,192

このネットワーク・グループの優先順位バンド。優先順位バンドが同じネットワーク・グループはすべて同時に使用されます。特定の優先順位を持つすべてのネットワーク回線が、管理者またはネットワークによって切断された場合は、1 つ下の優先順位の回線が使用されます。優先順位の高い回線に対しては、接続の再試行が行われます。

注意：BEA Tuxedo リリース 6.4 では、並列データ回線は優先グループ番号のネットワーク・グループ番号 (NETGRPNO) で優先付けされます。今後のリリースでは、並列のデータ回線の優先順位を別のアルゴリズムを使用して決めることができます。

制限事項　なし

T_NETMAP クラスの定義

概要 T_NETMAP クラスは、TM_MIB の T_MACHINE クラスからの TA_LMID を、T_NETGROUP クラスからの TA_NETGROUP オブジェクトに関連付けます。つまり、このクラスには論理マシンのネットワーク・グループへの割り当てが格納されます。TA_LMID は、複数の TA_NETGROUP グループに含めることができます。ある LMID が別の LMID に接続している場合、ブリッジ・プロセスは 2 つの LMID が属すネットワーク・グループのサブセットを決定します。一対の LMID がいくつかの共通したグループに存在する場合、それらの LMID は TA_NETPRIO の降順にソートされます (TA_NETGRPNO は二次キー)。TA_NETPRIO が同じネットワーク・グループは、ネットワーク・データを並列で送信します。ネットワークのエラーにより、最も優先順位の高いグループを使用してデータが送信できない場合は、次に優先順位の高いネットワーク・グループを使用します。これをフェイルオーバーと呼びます。データを送信しているネットワーク・グループよりも優先順位の高いすべてのネットワーク・グループが定期的に試行されます。TA_NETPRIO 値のより高いネットワーク接続が確立すると、優先順位の低い接続へのデータ送信はスケジューリングされなくなります。優先順位の低い接続は、必要がなくなると順番に切断されます。これをフェイルバックと呼びます。

属性表

表 53TM_MIB(5): T_NETMAP **クラス定義の属性表**

属性	タイプ	パーミッション	値	デフォルト値
TA_NETGROUP(r)(*)	string	ru-------	*string*[1..30]	N/A
TA_LMID(r)(*)	string	ru-------	*LMID*	N/A
TA_STATE	string	RW-------	GET: "VAL" SET: "{NEW | INV}"	N/A N/A
TA_NADDR	string	rw-r--r--	*string*[1..256] (注 1)	""
TA_FADDR	string	rw-r--r--	*string*[0..256] (注 1)	""
TA_FRANGE	long	rw-r--r--	1 <= *num* <= 65,535	1
TA_MINENCRYPTBITS	string	rwxrwx---	"{0 | 40 | 56 | 128}" (注 2)	"0"

表 53TM_MIB(5): T_NETMAP **クラス定義の属性表**

属性	タイプ	パーミッション	値	デフォルト値
TA_MAXENCRYPTBITS	string	rwxrwx---	"{0 | 40 | 56 | 128}"(注 2)	"128"

(r)— オブジェクトの作成に必要なフィールド (SET TA_STATE NEW)
(*)—GET/SET キー。SET 操作では 1 つ以上必要

注 1 BEA Tuxedo 8.0 以前では、この属性の文字列の最大長は 78 バイトです。
注 2 リンク・レベルの暗号化値の 40 ビットは、下位互換性を維持するために提供されています。

属性の意味

TA_NETGROUP: *string*[1..30]

この属性は、T_NETGROUP のクラス内にある関連付けられたネットワーク・グループの名前です。

TA_LMID:*LMID*

このネットワークのマッピングに使用する T_MACHINE クラス (TM_MIB 内) の論理マシン名です。

TA_STATE:

GET: "{VALid}"

GET 操作は、選択した T_NETMAP オブジェクトの実行時情報を検索します。以下に示す状態は、GET 要求への応答で返される TA_STATE の意味を示します。

VALid	T_NETMAP オブジェクトが定義されています。これがこのクラスの唯一の有効な状態です。ネットワークのマッピングが ACTive 状態になることはありません。

SET: "{NEW | INValid}"

SET 操作は、選択した T_NETMAP オブジェクトのコンフィギュレーション情報を更新します。以下に示す状態は、SET 要求で設定される TA_STATE の意味を示します。これ以外の状態を設定することはできません。

NEW	アプリケーションの T_NETMAP オブジェクトを作成します。状態の変更は INValid 状態でのみ可能です。正常終了すると、オブジェクトの状態は VALid になります。
unset	既存の T_NETMAP オブジェクトを変更します。正常終了した場合、オブジェクトの状態は変わりません。
INValid	指定したネットワークのマッピングを削除します。マッピングによりアクティブになっていたネットワーク・リンクは切断されます。この切断により、ネットワーク・リンクに関連付けられた T_BRIDGE オブジェクト (TM_MIB 内) で状態変更が発生する場合があります。

TA_NADDR: *string*[1..256] (BEA Tuxedo 8.0 以前では最大 78 バイト)

論理マシン上のブリッジ・プロセスが接続指示受け付けアドレスとして使用する完全なネットワーク・アドレスを指定します。BRIDGE の接続指示受け付けアドレスは、アプリケーションに参加している別のブリッジ・プロセスがコンタクトするときに使用するものです。つまり、T_DOMAIN:TA_OPTIONS 属性値で LAN オプションが設定されているときには必ず設定します。

string の形式が "0x*hex-digits*" の場合、文字数が偶数の有効な 16 進数値を含める必要があります。このような形式の文字列は、指定された文字列の 16 進数表現を含む文字配列に内部変換されます。

TCP/IP アドレスの場合は、以下のいずれかの形式になります。

- "*//hostname*:*port*"
- "*//#.#.#.#*:*port*"

TA_FADDR: *string*[0..256] (BEA Tuxedo 8.0 以前では最大 78 バイト)

tmboot、tmloadcf、BRIDGE などのローカル・プロセスが、アウトバウンド接続を確立する前にバインドできる完全なネットワーク・アドレスを指定します。このアドレスには、TCP/IP アドレスを指定する必要があります。この属性および TA_FRANGE 属性は、TCP/IP ポート (プロセスがアウトバウンド接続を行う前にバインドするときのバインド先) の範囲を指定します。この属性が NULL または空文字列に設定されている場合、オペレーティング・システムはバインド先のローカル・ポートをランダムに選択します。

string の形式が "0x*hex-digits*" の場合、文字数が偶数の有効な 16 進数値を含める必要があります。このような形式の文字列は、指定された文字列の 16 進数表現を含む文字配列に内部変換されます。

TCP/IP アドレスの場合は、以下のいずれかの形式になります。

- "*//hostname*:*port*"
- "*//#.#.#.#*:*port*"

TA_FRANGE: 1<= *num* <= 65,535
ローカル・プロセスがアウトバウンド接続を確立する前にバインドする TCP/IP ポートの範囲を指定します。範囲のベースとなるアドレスは、TA_FADDR 属性で指定します。

TA_MINENCRYPTBITS: "{0 | 40 | 56 | 128}"
ネットワーク・リンクを確立するときに必要な最高の暗号化レベルを指定します。0 は暗号化が行われないことを示し、40、56、および 128 は暗号化キーの長さをビット単位で指定します。ここで指定する最小レベルの暗号化が満たされない場合、リンクの確立は失敗します。デフォルトは 0 です。

注記　リンク・レベルの暗号化の値 40 は、下位互換性を維持するために提供されています。

制限事項 : この属性を変更しても、確立済みのネットワーク・リンクには反映されません。

TA_MAXENCRYPTBITS: "{0 | 40 | 56 | 128}"
リンクを確立するときに実行できる最高の暗号化レベルを指定します。0 は暗号化が行われないことを示し、40、56、および 128 は暗号化キーの長さをビット単位で指定します。デフォルトは 128 です。

注記　リンク・レベルの暗号化の値 40 は、下位互換性を維持するために提供されています。

制限事項 : この属性を変更しても、確立済みのネットワーク・リンクには反映されません。

128 ビット暗号化の使用が許可されている場合には、TA_MAXENCRYPTBITS のデフォルト値は 128 になります。56 ビット暗号化の使用が許可されている場合には、デフォルト値は 56 になります。暗号化の使用が許可されていない場合には、デフォルト値は 0 ビット

になります。ブリッジ・プロセスは接続の際、共通の最も高い TA_MAXENCRYPTBITS に調整されます。

制限事項　なし

T_QUEUE クラスの定義

概要 T_QUEUE クラスは、アプリケーション内のキューの実行時属性を表します。これらの属性の値によって、実行中のアプリケーションでサーバに割り当てられた BEA Tuxedo システムの要求キューを識別できます。また、それぞれのキュー・オブジェクトに関連付けられたアプリケーションの作業負荷に関する統計値も記録できます。

複数のマシンが存在するアプリケーションで MIB_LOCAL フラグを使用して GET 操作を実行する場合は、アクティブな各キューに対して複数のオブジェクト (ローカル属性を収集する論理マシンごとに 1 つずつ) が返されます 。

属性表

表 54TM_MIB(5): T_QUEUE **クラス定義の属性表**

属性	タイプ	パーミッション	値	デフォルト値
TA_RQADDR(*)	string	R--R--R--	*string*[1..30]	N/A
TA_SERVERNAME(k)	string	R--R--R--	*string*[1..78]	N/A
TA_STATE(k)	string	R--R--R--	GET: "{ACT | MIG | SUS | PAR}" SET:N/A	N/A N/A
TA_GRACE	long	R--R--R--	0 <= *num*	N/A
TA_MAXGEN	long	R--R--R--	1 <= *num* < 256	N/A
TA_RCMD	string	R--R--R--	*string*[0..256] (注 1)	N/A
TA_RESTART	string	R--R--R--	"{Y | N}"	N/A
TA_CONV	string	R--R--R--	"{Y | N}"	N/A
TA_LMID(k)	string	R--R--R--	*LMID*	N/A
TA_RQID	long	R--R--R--	1 <= *num*	N/A
TA_SERVERCNT	long	R--R--R--	1 <= *num* < 8,192	N/A

T_QUEUE クラス：ローカル属性

表 54TM_MIB(5): T_QUEUE **クラス定義の属性表 (続き)**

属性	タイプ	パーミッション	値	デフォルト値
TA_TOTNQUEUED	long	R-XR-XR--	0 <= *num*	N/A
TA_TOTWKQUEUED	long	R-XR-XR--	0 <= *num*	N/A
TA_SOURCE(k)	string	R--R--R--	*LMID*	N/A
TA_NQUEUED	long	R--R--R--	0 <= *num*	N/A
TA_WKQUEUED	long	R--R--R--	0 <= *num*	N/A

(k)—GET キー・フィールド
(*)—GET/SET キー。SET 操作では 1 つ以上必要

注 1 BEA Tuxedo 8.0 以前では、この属性の文字列の最大長は 78 バイトです。

属性の意味

TA_RQADDR: *string*[1..30]
要求キューのシンボリック・アドレス。同じ T_SERVER:TA_RQADDR 属性の値を持つサーバは、複数サーバ単一キュー (MSSQ) セットにまとめられます。T_QUEUE オブジェクトで返される属性値は、このシンボリックなキュー・アドレスに関連付けられたすべてのアクティブなサーバに適用されます。

TA_SERVERNAME: *string*[1..78]
サーバの実行可能ファイルの絶対パス名。T_QUEUE:TA_LMID 属性が示すマシンで、TA_SERVERNAME が示すサーバを実行していることを示します。この属性を GET 操作でキー・フィールドとして指定した場合は、関連パス名を指定することができ、すべての適切な絶対パスが一致します。

TA_STATE:

GET: "{ACTive | MIGrating | SUSpended | PARtitioned}"
GET 操作は、選択した T_QUEUE オブジェクトの実行時情報を検索します。T_QUEUE クラスは、コンフィギュレーション情報を直接示すわけではありません。ここで説明したコンフィギュレーション関連の属性は、関連のある T_SERVER オブジェクト

の一部として設定する必要があります。以下に示す状態は、GET 要求への応答で返される TA_STATE の意味を示します。

ACTive	この T_QUEUE オブジェクトに関連付けられたサーバが少なくとも 1 つアクティブな状態にあることを示します。
MIGrating	この T_QUEUE オブジェクトに関連付けられたサーバが現在 MIGrating 状態にあることを示します。この状態の詳細については、T_SERVER クラスを参照してください。この状態は、パーミッションの決定においては ACTive と同等です。
SUSpended	この T_QUEUE オブジェクトに関連付けられたサーバが現在 SUSpended 状態にあることを示します。この状態の詳細については、T_SERVER クラスを参照してください。この状態は、パーミッションの決定においては ACTive と同等です。
PARtitioned	この T_QUEUE オブジェクトに関連付けられたサーバが現在 PARtitioned 状態にあることを示します。この状態の詳細については、T_SERVER クラスを参照してください。この状態は、パーミッションの決定においては ACTive と同等です。

SET:

SET 操作は、選択した T_QUEUE オブジェクトの実行時情報を更新します。状態の変更は、T_QUEUE オブジェクトの情報の更新時にのみ可能です。既存の T_QUEUE オブジェクトの変更は、オブジェクトが ACTive 状態にある場合にのみ可能です。

TA_GRACE: 0 <= *num*

T_QUEUE:TA_MAXGEN の制限が適用される期間を示します (単位は秒)。この属性は、再起動が可能なサーバに対してのみ (T_QUEUE:TA_RESTART 属性が "Y" にセットされている場合にのみ) 有効です。この属性を 0 に設定すると、サーバはいつでも再起動できます。

TA_MAXGEN: 1 <= *num* < 256
: 指定された猶予期間 (T_QUEUE:TA_GRACE) において、このキューに関連付けられた再起動可能なサーバ (T_QUEUE:TA_RESTART == "Y") に許可された最大の世代数。各サーバを最初にアクティブにする動作を 1 つの世代としてカウントし、その後の再起動もそれぞれ 1 つの世代としてカウントします。

TA_RCMD: *string*[0..256] (BEA Tuxedo 8.0 以前では最大 78 バイト)
: このキューに関連付けられたアプリケーション・サーバのシステムの再起動と並行して実行するアプリケーション指定のコマンド。

TA_RESTART: "{Y | N}"
: このキューに関連付けられたサーバを再起動できる ("Y") かできない ("N") かを示します。

TA_CONV: "{Y | N}"
: このキューに関連付けられたサーバが、会話方式 ("Y") であるか要求 / 応答方式 ("N") であるかを示します。

TA_LMID:*LMID*
: このキューに関連付けられたサーバがアクティブになっている論理マシン。

TA_RQID: 1 <= *num*
: UNIX システムのメッセージ・キューの識別子。
制限事項 : UNIX システム固有の属性です。アプリケーションを実行しているプラットフォームが UNIX ベースでない場合、この属性は返されないことがあります。

TA_SERVERCNT: 1 <= *num* < 8,192
: このキューに関連付けられたアクティブなサーバの数。

TA_TOTNQUEUED: 0 <= *num*
: このキューがアクティブな間のキューの長さの合計。この合計値には、キュー上ですでにアクティブでなくなっているサーバによって登録および処理された要求も含まれます。キューに新たな要求が割り当てられると、要求がキューに登録される直前にそのキューの長さが合計値に加算されます。

  制限事項 : T_DOMAIN:TA_LDBAL 属性が "N" に設定されている場合、および T_DOMAIN:TA_MODEL 属性が "MP" の場合、TA_TOTNQUEUED は返されま

せん。また、同様のコンフィギュレーションでは、この属性に対する更新は無視されます。したがって、この属性が返される場合には、TA_LMID と TA_SOURCE が同じ値になります。

TA_TOTWKQUEUED: 0 <= *num*

このキューがアクティブな間にキューに登録された負荷の合計。この合計値には、キュー上ですでにアクティブでなくなっているサーバによって登録および処理された要求も含まれます。キューに新たな要求が割り当てられると、要求がキューに登録される直前にそのキュー上の負荷が合計値に加算されます。

制限事項 : T_DOMAIN:TA_LDBAL 属性が "N" に設定されている場合、および T_DOMAIN:TA_MODEL 属性が "MP" の場合、TA_TOTWKQUEUED は返されません。また、同様のコンフィギュレーションでは、この属性に対する更新は無視されます。したがって、この属性が返される場合には、TA_LMID と TA_SOURCE が同じ値になります。

TA_SOURCE:*LMID*

ローカル属性値の検索元となる論理マシン。

TA_NQUEUED: 0 <= *num*

TA_SOURCE の論理マシンから現在このキューに登録されている要求の数。この値は、要求がキューに登録されると増加し、サーバがキューから要求を取り出すと減少します。

制限事項 : "T_DOMAIN":"TA_LDBAL" 属性が "N" に設定されている場合、および T_DOMAIN:TA_MODEL 属性が "MP" の場合、TA_NQUEUED は返されません。したがって、この属性が返される場合には、TA_LMID と TA_SOURCE が同じ値になります。

TA_WKQUEUED: 0 <= *num*

TA_SOURCE の論理マシンから現在このキューに登録されている負荷。T_DOMAIN:TA_MODEL 属性が SHM に設定されており、T_DOMAIN:TA_LDBAL 属性が "Y" に設定されている場合、TA_WKQUEUED 属性はアプリケーション全体を通じてこのキューに登録されている負荷を示します。ただし、TA_MODEL が MP に設定されており、TA_LDBAL が "Y" にセットされている場合は、最近のタイムスパンにおいて TA_SOURCE の論理マシンからこのキューに登録された負荷を示します。この属性は、ロード・バランシングのために使用します。そのため、新たに起動した

サーバが排除されないよう、BBL が各マシンのこの属性値を定期的にゼロにリセットします。

制限事項　なし

T_ROUTING クラスの定義

概要 T_ROUTING クラスは、アプリケーションに対するルーティング指定のコンフィギュレーション属性を表します。これらの属性値によって、フィールド名、バッファ・タイプ、およびルーティングの定義に関して、アプリケーション・データに依存するルーティング基準を識別できます。

属性表

表 55TM_MIB(5): T_ROUTING クラス定義の属性表

属性	タイプ	パーミッション	値	デフォルト値
TA_ROUTINGNAME(r)(*)	string	ru-r--r--	*string*[1..15]	N/A
TA_ROUTINGTYPE(r)	string	ru-r--r--	SERVICE or FACTORY	"SERVICE"
TA_BUFTYPE(r)(*)	string	ru-r--r--	*string*[1..256]	N/A(注 1)
TA_FIELD(r)(k)(*)	string	ru-r--r--	*string*[1..30]	N/A(注 1)
TA_FIELDTYPE	string	ru-r--r--	[char | short | long | float | double | string]	"string"
TA_FIELDTYPE(r) (ファクトリ・ベース・ルーティングのみ)	string	rw-r--r--	string[1..30]	N/A
TA_RANGES(r)	carray	rw-r--r--	*carray*[1..2048]	N/A
TA_TYPE	string	ru-r--r--	string[1..15]	"SERVICE"
TA_STATE(k)	string	rw-r--r--	GET: "VAL" SET: "{NEW | INV}"	N/A N/A

(k)—GET キー・フィールド
(r)— オブジェクトの作成に必要なフィールド (SET TA_STATE NEW)
(*)—GET/SET キー、SET 操作では 1 つ以上必要

注 1 TA_BUFTYPE は、ATMI のデータ依存型ルーティング基準にのみ適用されます。TA_FIELDTYPE は、CORBA のファクトリ・ベース・ルーティング基

準にのみ適用されます。u (一意性) パーミッションは、該当する場合にのみ適用されます。つまり、TA_ROUTINGNAME、TA_TYPE、および TA_BUFTYPE の組み合わせは、TA_TYPE=SERVICE で一意である必要があります。また、TA_ROUTINGNAME、TA_TYPE、および TA_FIELD の組み合わせは、TA_TYPE=FACTORY で一意である必要があります。
TA_TYPE 属性は、TA_ROUTING オブジェクトに許可する属性を決定します。TYPE=SERVICE は、ATMI のデータ依存型ルーティング基準に対応します。TYPE=FACTORY は、CORBA のファクトリ・ベース・ルーティングに対応します。デフォルトは "SERVICE" です。TA_TYPE が指定されていない場合、SET 操作はデータ依存型ルーティングに対する操作とみなされます。TA_FIELDTYPE を指定しても、データ依存型ルーティングに対しては無効です。TA_BUFTYPE を指定しても、ファクトリ・ベース・ルーティングに対しては無効です。

属性の意味

TA_ROUTINGNAME: *string*[1..15]
ルーティング基準の名前。

TA_ROUTINGTYPE:*type*
ルーティングのタイプを指定します。デフォルトは TYPE=SERVICE で、ATMI 環境で使用される既存の UBBCONFIG ファイルが引き続き正常に動作することを保証します。CORBA インターフェイス用にファクトリ・ベース・ルーティングを実装する場合は、TYPE=FACTORY を使用します。

TA_BUFTYPE: "*type1*[:*subtype1*[,*subtype2*...]][;*type2*[:*subtype3*[,...]]] ..."
このルーティング・エントリで有効なデータ・バッファのタイプとサブタイプのリスト。最大で 32 のタイプとサブ・タイプの組み合わせを使用できます。タイプは、FML、FML32、XML、VIEW、VIEW32、X_C_TYPE、および X_COMMON に制限されています。FML、FML32、または XML に対してはサブタイプを指定できず、VIEW、VIEW32、X_C_TYPE、および X_COMMON ではサブタイプを指定する必要があります ("*" は使用できません)。サブ・タイプの名前には、セミコロン (;)、カンマ (,)、コロン (:)、アスタリスク (*) は使用できません。タイプとサブタイプのペアのうち、重複するものは同じルーティング基準名として指定できません。タイプとサブタイプのペアが一意の場合、複数のルーティング・エントリは同じ基準名を持つことができます。単一のルーティング・エントリに複数のバッファ・タイプが指定される場合、各バッファ・タイプに対するルーティング・フィールドのデータ型は同じでなければなりません。

TA_FIELD: *string*[1..30]

ルーティング・フィールドの名前。TA_TYPE=FACTORY である場合、このファクトリ・ルーティング基準に関連付けられているインターフェイスの PortableServer::POA::create _reference_with_criteria に対する NVList パラメータで指定されたフィールドとみなされます。詳細については、ファクトリ・ベース・ルーティングに関するセクションを参照してください。

TA_TYPE=SERVICE の場合、TA_FIELD フィールドは、FML バッファまたは FML32 バッファ、XML バッファ、FML フィールド・テーブル (環境変数の FLDTBLDIR と FIELDTBLS、または FLDTBLDIR32 と FIELDTBLS32 を使用) で識別されるビュー・フィールド名、または FML ビュー・テーブル (環境変数の VIEWDIR と VIEWFILES、または VIEWDIR32 と VIEWFILES32 を使用) とみなされます。この情報は、メッセージの送信時に、データ依存型ルーティングに関連するフィールド値を取得するために使用されます。

XML バッファ・タイプの場合、TA_FIELD には、ルーティング要素のタイプ (または名前) か、ルーティング要素の属性名のいずれかが含まれます。

XML バッファ・タイプの場合、TA_FIELD パラメータの構文は次のとおりです。

"root_element[*/child_element*][*/child_element*][*/. . .*][*/@attribute_name*]"

要素は、XML ドキュメントまたはデータグラム要素のタイプとして処理されます。インデックスはサポートされません。したがって、BEA Tuxedo システムは、データ依存型ルーティングで XML バッファを処理する際に、与えられた要素タイプの最初のオカレンスだけを認識します。この情報は、メッセージの送信時に、データ依存型ルーティングに関連する要素の内容を取得するために使用されます。内容は UTF-8 で符号化された文字列である必要があります。

属性は、定義されている要素の XML ドキュメントまたはデータグラム属性として処理されます。この情報は、メッセージの送信時に、データ依存型ルーティングに関連する属性値を取得するために使用されます。値は UTF-8 で符号化された文字列である必要があります。

要素名と属性名の組み合わせの長さは最大で 30 文字です。

ルーティング・フィールドのタイプは、TA_FIELDTYPE 属性で指定できます。

TA_FIELDTYPE: "{char | short | long | float | double | string}"
: TA_FIELDTYPE 属性で指定したルーティング・フィールドのタイプ。フィールドのタイプには、char、short、long、float、double、または string を指定できますが、指定できるタイプは 1 つのみです。この属性は、XML バッファをルーティングする場合にのみ使用されます。ルーティング・フィールドのデフォルトのデータ型は string です。

TA_FIELDTYPE (ファクトリ・ベース・ルーティングのみ)
: ルーティング・フィールドのタイプ。このフィールドは、TA_TYPE=FACTORY の場合にのみ有効です。指定可能なタイプは、SHORT、LONG、FLOAT、DOUBLE、CHAR、STRING です。この属性の指定は、ファクトリ・ベース・ルーティング基準に対してのみ有効です。

TA_RANGES: *carray*[1..2048]
: ルーティング・フィールドの範囲および関連付けられたサーバ・グループを指定します。*string* の形式は、カンマで区切って並べられた範囲とグループ名の組み合わせです。範囲とグループ名の組み合わせの形式は次のとおりです。

  lower[-upper]:group

  lower と *upper* は、符号を持つ数値、または一重引用符で囲んだ文字列です。*lower* には、*upper* 以下の値を設定する必要があります。文字列値で一重引用符を使用する場合は、引用符の前にバックスラッシュを 2 つを入力します (例 : 'O\\'Brien')。マシン上の関連するフィールドのデータ型の最小値を示すには、MIN を使用します。マシン上の関連するフィールドのデータ型の最大値を示すには、MAX を使用します。したがって、"MIN - -5 " は -5 以下のすべての数を表し、"6-MAX" は 6 以上のすべての数を表します。

  範囲内のメタキャラクタ "*" (ワイルドカード) は、すでにエントリとして指定した範囲では使用されなかった任意の値を示します。各エントリでは、1 つのワイルドカードによる範囲指定のみ可能です。1 つのエントリで使用できるワイルドカード範囲は 1 つのみで、最後になければなりません (その後の範囲は無視されます)。

ルーティング・フィールドには、FML でサポートされている任意のデータ型を指定できます。数値ルーティング・フィールドには数値で範囲を指定し、文字列ルーティング・フィールドには、文字列で範囲を指定する必要があります。

文字列で範囲を設定する場合は、文字列、carray、および文字フィールド型の値を一重引用符で囲みます。先頭に符号を付けることはできません。short 型および long 型の整数値は数字の文字列であり、オプションで先頭に正の符号または負の符号を付けることができます。浮動小数点数は、C コンパイラまたは atof(3) で受け入れられる形式で指定します。つまり、符号 (オプション)、数字の文字列 (オプションで小数点を追加)、e または E (オプション)、符号またはスペース (オプション)、整数という形式で指定します。

グループ名は、フィールドが範囲と一致する場合に要求のルーティング先となるグループを示します。グループ名 "*" は、サーバが必要なサービスを提供するグループであればどのグループにでも要求をルーティングできることを示します。

制限事項 : 256 バイトを超える属性値を使用すると、BEA Tuxedo リリース 4.2.2 以前との互換性が失われます。

TA_STATE:

GET: "{VALid}"

GET 操作は、選択した T_ROUTING オブジェクトのコンフィギュレーション情報を検索します。以下に示す状態は、GET 要求への応答で返される TA_STATE の意味を示します。これら以外の状態は返されません。

VALid	T_ROUTING オブジェクトが定義されていることを示します。これがこのクラスの唯一の有効な状態です。ルーティング条件は、ACTive になることはありません。コンフィギュレーションによってサービス名と関連付けられ、実行時にデータ依存ルーティングを提供するために使用されます。この状態は INActive と同等で、パーミッションのチェックに使用します。

SET: "{NEW | INValid}"

SET 操作は、選択した T_ROUTING オブジェクトのコンフィギュレーション情報を更新します。以下に示す状態は、SET 要求で設定される TA_STATE の意味を示します。以下に示されていない状態は設定できません。

NEW	アプリケーションに対する T_ROUTING オブジェクトを作成します。状態の変更は INValid 状態でのみ可能です。正常終了すると、オブジェクトの状態は VALid になります。
unset	既存の T_ROUTING オブジェクトを変更します。この組み合わせは INValid 状態では使用できません。正常終了した場合、オブジェクトの状態は変わりません。
INValid	アプリケーションに対する T_ROUTING オブジェクトを削除します。状態の変更は VALid 状態でのみ可能です。正常終了すると、オブジェクトの状態は INValid になります。

TA_TYPE

ルーティング基準のタイプ。有効な値は "FACTORY" または "SERVICE" です。"FACTORY" を指定すると、ルーティング基準は CORBA インターフェイス用のファクトリ・ベース・ルーティングに適用されます。ファクトリ・ベース・ルーティング基準では、必ず TYPE=FACTORY を指定する必要があります。"SERVICE" を指定すると、ルーティング基準は ATMI サービス用のデータ依存型ルーティングに適用されます。デフォルトは "SERVICE" です。したがって、データ依存型ルーティングを使用する場合はこの属性の指定を省略できます。ここで指定するルーティング基準のタイプは、この MIB クラスに対して定義されるほかのフィールドの値に影響します。これらのフィールドで指定できる値については個別に説明します。ファクトリ・ベース・ルーティング基準に対する SET 操作では、TA_TYPE の指定が必須です。

制限事項　なし

T_SERVER クラスの定義

概要 T_SERVER クラスは、アプリケーション内のサーバのコンフィギュレーション属性と実行時属性を表します。これらの属性値によって、コンフィギュレーション済みのサーバを識別したり、各サーバ・オブジェクトに関連する統計値やリソースを実行時にトラッキングしたりできます。返される情報には、サーバのすべてのコンテキストで共通のフィールドが常に含まれています。また、システムにマルチコンテキストとして定義されていないサーバ (TA_MAXDISPATCHTHREADS の値が 1) の場合は、このクラスにサーバのコンテキストに関する情報が含まれます。システムにマルチコンテキストとして定義されているサーバの場合は、プレースホルダの値がコンテキストごとの属性に対して通知されます。コンテキストごとの属性は、常に T_SERVERCTXT クラスにあります。T_SERVERCTXT クラスは、シングル・コンテキストのサーバに対しても定義されます。

TA_CLTLMID、TA_CLTPID、TA_CLTREPLY、TA_CMTRET、TA_CURCONV、TA_CURREQ、TA_CURRSERVICE、TA_LASTGRP、TA_SVCTIMEOUT、TA_TIMELEFT、および TA_TRANLEV の各属性は、サーバ・ディスパッチ・コンテキストごとに固有な属性です。これら以外の属性は、すべてのサーバ・ディスパッチ・コンテキストで共通です。

属性表

表 56TM_MIB(5): T_SERVER **クラス定義の属性表**

属性	タイプ	パーミッション	値	デフォルト値
TA_SRVGRP(r)(*)	string	ru-r--r--	*string*[1..30]	N/A
TA_SRVID(r)(*)	long	ru-r--r--	1 <= *num* < 30,001	N/A
TA_SERVERNAME(k)(r)	string	rw-r--r--	*string*[1..78]	N/A
TA_GRPNO(k)	long	r--r--r--	1 <= *num* < 30,000	N/A
TA_STATE(k)	string	rwxr-xr--	GET: "{ACT | INA | MIG | CLE | RES | SUS | PAR | DEA}"	N/A
			SET: "{NEW | INV | ACT | INA | DEA}"	N/A
TA_BASESRVID	long	r--r--r--	1 <= *num* < 30,001	N/A

表 56TM_MIB(5): T_SERVER **クラス定義の属性表 (続き)**

属性	タイプ	パーミッション	値	デフォルト値
TA_CLOPT	string	rwyr--r--	*string*[0..256]	"-A"
TA_ENVFILE	string	rwyr--r--	*string*[0..256] (注 2)	""
TA_GRACE	long	rwyr--r--	0 <= *num*	86,400
TA_MAXGEN	long	rwyr--r--	1 <= *num* < 256	1
TA_MAX	long	rwxr--r--	1 <= *num* < 1,001	1
TA_MIN	long	rwyr--r--	1 <= *num* < 1,001	1
TA_MINDISPATCHTHREADS	long	rwyr--r--	1 <= *num* < 1,000	1
TA_MAXDISPATCHTHREADS	long	rwyr--r--	0 <= *num* < 1,000	0
TA_THREADSTACKSIZE	long	rwyr--r--	0 <= *num* <= 2147483647	0
TA_CURDISPATCHTHREADS	long	R-XR-XR--	0 <= *num*	N/A
TA_HWDISPATCHTHREADS	long	R-XR-XR--	0 <= *num*	N/A
TA_NUMDISPATCHTHREADS	long	R-XR-XR--	0 <= *num*	N/A
TA_RCMD	string	rwyr--r--	*string*[0..256] (注 2)	""
TA_RESTART	string	rwyr--r--	"{Y | N}"	N
TA_SEQUENCE(k)	long	rwxr--r--	1 <= *num* < 10,000	>= 10,000
TA_SYSTEM_ACCESS	string	rwyr--r--	"{FASTPATH | PROTECTED}"	(注 1)
TA_CONV(k)	string	rw-r--r--	"{Y|N}"	"N"
TA_REPLYQ	string	rw-r--r--	"{Y|N}"	"N"
TA_RPPERM	long	rw-r--r--	0001 <= *num* <= 0777	(注 1)
TA_RQADDR(k)	string	rw-r--r--	*string*[0..30]	"*GRPNO.SRVID*"
TA_RQPERM	long	rw-r--r--	0001 <= *num* <= 0777	(注 1)

表 56TM_MIB(5): T_SERVER **クラス定義の属性表 (続き)**

属性	タイプ	パーミッション	値	デフォルト値
TA_LMID(k)	string	R--R--R--	*LMID*	N/A
TA_GENERATION	long	R--R--R--	1 <= *num* < 32,768	N/A

表 56TM_MIB(5): T_SERVER **クラス定義の属性表 (続き)**

属性	タイプ	パーミッション	値	デフォルト値
TA_PID(k)	long	R--R--R--	1 <= *num*	N/A
TA_RPID	long	R--R--R--	1 <= *num*	N/A
TA_RQID	long	R--R--R--	1 <= *num*	N/A
TA_TIMERESTART	long	R--R--R--	1 <= *num*	N/A
TA_TIMESTART	long	R--R--R--	1 <= *num*	N/A
TA_SEC_PRINCIPAL_NAME	string	rwxr--r--	*string*[0..511]	""
TA_SEC_PRINCIPAL_LOCATION	string	rwxr--r--	*string*[0..511]	""
TA_SEC_PRINCIPAL_PASSVAR	string	rwxr--r--	*string*[0..511]	""
TA_SICACHEENTRIESMAX	string	rw-r--r--	{"0"-"32767" | "DEFAULT"}	"DEFAULT"
T_SERVER クラス：ローカル属性				
TA_NUMCONV	long	R-XR-XR--	0 <= *num*	N/A
TA_NUMDEQUEUE	long	R-XR-XR--	0 <= *num*	N/A
TA_NUMENQUEUE	long	R-XR-XR--	0 <= *num*	N/A
TA_NUMPOST	long	R-XR-XR--	0 <= *num*	N/A
TA_NUMREQ	long	R-XR-XR--	0 <= *num*	N/A
TA_NUMSUBSCRIBE	long	R-XR-XR--	0 <= *num*	N/A
TA_NUMTRAN	long	R-XR-XR--	0 <= *num*	N/A
TA_NUMTRANABT	long	R-XR-XR--	0 <= *num*	N/A
TA_NUMTRANCMT	long	R-XR-XR--	0 <= *num*	N/A
TA_TOTREQC	long	R-XR-XR--	0 <= *num*	N/A
TA_TOTWORKL	long	R-XR-XR--	0 <= *num*	N/A

表 56TM_MIB(5): T_SERVER **クラス定義の属性表 (続き)**

属性	タイプ	パーミッション	値	デフォルト値
TA_CLTLMID	string	R--R--R--	*LMID*	N/A

表 56TM_MIB(5): T_SERVER **クラス定義の属性表 (続き)**

属性	タイプ	パーミッション	値	デフォルト値
TA_CLTPID	long	R--R--R--	1 <= *num*	N/A
TA_CLTREPLY	string	R--R--R--	"{Y | N}"	N/A
TA_CMTRET	string	R--R--R--	"{COMPLETE | LOGGED}"	N/A
TA_CURCONV	long	R--R--R--	0 <= *num*	N/A
TA_CUROBJECTS	long	R--R--R--	0 <= num	N/A
TA_CURINTERFACE	string	R--R--R--	*string*[0..128]	N/A
TA_CURREQ	long	R--R--R--	0 <= *num*	N/A
TA_CURRSERVICE	string	R--R--R--	*string*[0..15]	N/A
TA_CURTIME	long	R--R--R--	1 <= *num*	N/A
TA_LASTGRP	long	R--R--R--	1 <= *num* < 30,000	N/A
TA_SVCTIMEOUT	long	R--R--R--	0 <= *num*	N/A
TA_TIMELEFT	long	R--R--R--	0 <= *num*	N/A
TA_TRANLEV	long	R--R--R--	0 <= *num*	N/A

(k)—GET キー・フィールド
(r)— オブジェクトの作成に必要なフィールド (SET TA_STATE NEW)
(*)—GET/SET キー、SET 操作では 1 つ以上必要

注 1 デフォルト設定は、T_DOMAIN クラスでこの属性に指定したのと同じ値になります。
注 2 BEA Tuxedo 8.0 以前では、この属性の文字列の最大長は 78 バイトです。

属性の意味

TA_SRVGRP: *string*[1..30]
サーバ・グループの論理名。また、サーバ・グループ名にはアスタリスク (*)、カンマ (,)、コロン (:) は使用できません。

TA_SRVID: 1 <= *num* < 30,001
サーバ・グループ内で一意なサーバ識別番号。

TA_SERVERNAME: *string*[1..78]

サーバの実行可能ファイルの名前。TA_SERVERNAME によって識別されるサーバは、このサーバのサーバ・グループの T_GROUP:TA_LMID によって識別されるマシン上で実行されます。相対パス名を指定すると、実行ファイルの検索が、最初 TA_APPDIR で実行されてから、TA_TUXDIR/bin、/bin、/usr/bin、*path* の順番で実行されます。なお、*path* はマシン環境ファイルで最初に現れる PATH= 行の値です。アクティブなサーバに返される属性値は、常に絶対パス名になることに注意してください。TA_APPDIR と TA_TUXDIR の値は、適切な T_MACHINE オブジェクトから取得します。環境変数の処理方法については、T_MACHINE:TA_ENVFILE 属性の項を参照してください。

TA_GRPNO: 1 <= *num* < 30,000

このサーバのグループに関連付けられたグループ番号。

TA_STATE:

GET: "{ACTive | INActive | MIGrating | CLEaning | REStarting | SUSpended | PARtitioned | DEAd}"

GET 操作は、選択した T_SERVER オブジェクトのコンフィギュレーション情報および実行時情報を検索します。以下に示す状態は、GET 要求への応答で返される TA_STATE の意味を示します。

ACTive	T_SERVER オブジェクトは、定義済みでアクティブ状態です。これは、サーバがビジーかアイドルかを示すものではありません。長さがゼロでない TA_CURRSERVICE 属性を持ったアクティブ・サーバは、ビジー・サーバつまりサーバ要求を処理中のサーバとして解釈されます。
INActive	T_SERVER オブジェクトは、定義済みで非アクティブ状態です。

MIGrating	T_SERVER オブジェクトは定義済みで、現時点でサーバ・グループのセカンダリ論理マシンへの移行状態にあります。セカンダリ論理マシンとは、T_GROUP:TA_LMID 属性に登録された論理マシンのうち、T_GROUP:TA_CURLMID 属性に該当しない論理マシンです。この状態は、パーミッションの決定においては ACTive と同等です。
CLEaning	T_SERVER は定義済みで、異常終了が発生したためシステムによってクリーンアップされている状態です。実行可能サーバは、TA_GRACE の時間内に TA_MAXGEN の起動 / 再起動時間を超えるとこの状態に移行します。この状態は、パーミッションの決定においては ACTive と同等です。
REStarting	T_SERVER は定義済みで、異常終了が発生したためシステムによって再起動されている状態です。この状態は、パーミッションの決定においては ACTive と同等です。
SUSpended	T_SERVER オブジェクトは定義済みで、現在シャットダウンを保留して停止している状態です。この状態は、パーミッションの決定においては ACTive と同等です。
PARtitioned	T_SERVER オブジェクトは定義済みでアクティブな状態ですが、サーバが動作しているマシンは現在 T_DOMAIN:TA_MASTER サイトから分断されています。この状態は、パーミッションの決定においては ACTive と同等です。
DEAd	T_SERVER オブジェクトは定義済みで、掲示板ではアクティブと識別されているにもかかわらず、異常終了が原因で現在は実行されていないことを示します。この状態が維持されるのは、サーバのローカル BBL が終了を検知し、動作 (REStarting|CLEaning) を行うまでです。この状態が返されるのは、MIB_LOCAL TA_FLAGS 値が指定され、サーバが動作しているマシンが接続可能な場合のみです。この状態は、パーミッションの決定においては ACTive と同等です。

SET: "{NEW | INValid | ACTive | INActive | DEAd}"

SET 操作は、選択した T_SERVER オブジェクトのコンフィギュレーション情報および実行時情報を更新します。以下に示す状態は、SET 要求で設定される TA_STATE の意味を示します。以下に示されていない状態は設定できません。

NEW	アプリケーションに対する T_SERVER オブジェクトを作成します。状態の変更は INValid 状態でのみ可能です。正常に終了すると、オブジェクトの状態は INActive になります。
unset	既存の T_SERVER オブジェクトを変更します。この組み合わせは、ACTive 状態または INActive 状態でのみ可能です。正常終了した場合、オブジェクトの状態は変わりません。
INValid	アプリケーションに対する T_SERVER オブジェクトを削除します。状態の変更は、INActive 状態でのみ可能です。正常終了すると、オブジェクトの状態は INValid になります。
ACTive	T_SERVER オブジェクトをアクティブにします。状態の変更は、INActive 状態でのみ可能です (MIGrating 状態のサーバは、T_GROUP:TA_STATE を ACTive に設定して再起動する必要があります)。この状態遷移に対するパーミッションの決定に際しては、アクティブなオブジェクトのパーミッション (--x--x--x) が考慮されます。正常に終了すると、オブジェクトの状態は ACTive になります。サーバをアクティブにする際に個々のサーバの状態が必要な場合には、TMIB_NOTIFY TA_FLAG 値を使用します。
INActive	T_SERVER オブジェクトを非アクティブにします。状態の変更は、ACTive 状態でのみ可能です。正常に終了すると、オブジェクトの状態は INActive になります。サーバを非アクティブにする際に個々のサーバの状態が必要な場合には、TMIB_NOTIFY TA_FLAG 値を使用します。

DEAd	適切なタイムアウト間隔 (MIB(5) の TA_MIBTIMEOUT を参照) 後もまだサーバが動作している場合に、サーバに SIGTERM シグナルと SIGKILL シグナルを送信して T_SERVER オブジェクトを非アクティブ化します。デフォルトでは、SIGTERM シグナルによってサーバの順序立てたシャットダウンが開始され、サーバは再起動可能であっても非アクティブになります。サーバが 1 つのサービスを長時間にわたって処理している場合、または SIGTERM シグナルが無効になっている場合は、SIGKILL が使用され、システムはそれを異常終了として処理します。状態の変更は、ACTive 状態または SUSpended 状態でのみ可能です。正常に終了すると、オブジェクトの状態は INActive、CLEaning、または REStarting になります。

TA_BASESRVID: 1 <= *num* < 30,001
: ベース・サーバ識別子。TA_MAX 属性値が 1 のサーバの場合、この属性は常に TA_SRVID と同じになります。一方、TA_MAX 値が 1 より大きいサーバの場合、この属性は同一の環境設定を持つサーバの集まりが同一のベース・サーバ識別子であることを示します。

TA_CLOPT: *string*[0..256]
: サーバをアクティブにする際に渡すコマンド行オプション。詳細については、servopts(5) のリファレンス・ページを参照してください。制限事項 : この属性を実行時に変更しても、実行中のサーバには反映されません。

TA_ENVFILE: *string*[0..256] (BEA Tuxedo 8.0 以前では最大 78 バイト)
: サーバ固有の環境ファイルこのファイルを使用して環境を変更する方法の詳細については、T_MACHINE:TA_ENVFILE を参照してください。制限事項 : この属性を実行時に変更しても、実行中のサーバには反映されません。

TA_GRACE: 0 <= *num*
: T_QUEUE:T_SERVER の制限が適用される期間を示します (単位は秒)。この属性は、再起動が可能なサーバに対してのみ (T_SERVER:TA_RESTART 属性が "Y" にセットされている場合にのみ) 有効です。再起動しているサーバが TA_MAXGEN 制限値を超えても、

TA_GRACE の期限が切れている場合は、システムは現在の世代 (T_SERVER:TA_GENERATION) を 1 にリセットし、初期起動時間 (T_SERVER:TA_TIMESTART) を現在の時刻にリセットします。この属性を 0 に設定すると、サーバはいつでも再起動できます。

要求キューを共有するサーバ (つまり、T_SERVER:TA_RQADDR の値が同一のサーバ) は、この属性も同じ値に設定する必要があります。この属性値が異なる場合、キュー上のすべてのサーバに関連付けられる実行時値が、最初にアクティブにしたサーバによって確立されます。

制限事項 : この属性を実行時に変更すると、実行中のサーバおよび要求キューを共有しているほかのアクティブ・サーバに反映されます。ただし、変更されるのは選択したサーバのコンフィギュレーション・パラメータのみです。したがって、キューを共有するすべてのサーバについてこの属性を同じ値に設定しないと、その後サーバをアクティブにする際の起動順序によってアプリケーションの動作が変わってしまいます。

TA_MAXGEN: 1 <= *num* < 256

指定された猶予期間 (T_SERVER:TA_GRACE) において、再起動可能なサーバ (T_SERVER:TA_RESTART == "Y") に許可された最大の世代数。サーバを最初にアクティブにする動作を 1 つの世代としてカウントし、その後の再起動もそれぞれ 1 つの世代としてカウントします。最大世代数を超えた後の処理については前述の TA_GRACE を参照してください。

要求キューを共有するサーバ (つまり、T_SERVER:TA_RQADDR の値が同一のサーバ) は、この属性も同じ値に設定する必要があります。この属性値が異なる場合、キュー上のすべてのサーバに関連付けられる実行時値が、最初にアクティブにしたサーバによって確立されます。

制限事項 : この属性を実行時に変更すると、実行中のサーバおよび要求キューを共有しているほかのアクティブ・サーバに反映されます。ただし、変更されるのは選択したサーバのコンフィギュレーション・パラメータのみです。したがって、キューを共有するすべてのサーバについてこの属性を同じ値に設定しないと、その後サーバをアクティブする際の起動順序によってアプリケーションの動作が変わってしまいます。

TA_MAX: 1 <= *num* < 1,001
: 起動するサーバのオカレンスの最大数。最初に、tmboot() がサーバの T_SERVER:TA_MIN オブジェクトを起動します。その他のオブジェクトは、特定サーバ ID を起動して個別に起動するか、自動生成によって起動します (会話型サーバの場合のみ)。この属性を実行時に変更すると、同一のコンフィギュレーションを持つサーバのうち実行中のサーバ (前述の TA_BASESRVID を参照) と、サーバのコンフィギュレーション定義に反映されます。

TA_MIN: 1 <= *num* < 1,001
: 起動するサーバのオカレンスの最小数。T_SERVER:TA_RQADDR が指定されており、TA_MIN が 1 より大きい場合、そのサーバは MSSQ セットを形成します。サーバのサーバ識別子は、T_SERVER:TA_SRVID から TA_SRVID + T_SERVER:TA_MAX - 1 までとなります。各サーバには、すべて同一のシーケンス番号とその他のサーバ・パラメータが付けられます。

  制限事項 : この属性を実行時に変更しても、実行中のサーバには反映されません。

TA_MINDISPATCHTHREADS: 1 <= *num* < 1,000
: 最初のサーバの起動時に開始されるサーバ・ディスパッチ・スレッドの数。この属性は、サーバが buildserver -t コマンドを使用して構築された場合にのみ有効です。

  TA_MAXDISPATCHTHREADS > 1 のときに使用される個別のディスパッチャ・スレッドは、TA_MINDISPATCHTHREADS の値の一部としてはカウントされません。この場合、TA_MINDISPATCHTHREADS <= TA_MAXDISPATCHTHREADS である必要があります。
  TA_MINDISPATCHTHREADS が指定されていない場合のデフォルト値は 0 です。

  制限事項 : この属性を実行時に変更しても、実行中のサーバには反映されません。

TA_MAXDISPATCHTHREADS: 0 <= *num* < 1,000
: 個々のサーバ・プロセスで生成可能な、同時にディスパッチされるスレッドの最大数を指定します。この属性は、サーバが buildserver -t コマンドを使用して構築された場合にのみ有効です。

TA_MAXDISPATCHTHREADS > 1 の場合、別のディスパッチ・スレッドが使用されます。このディスパッチ・スレッドは、パラメータで指定した数には含まれません。この場合、TA_MINDISPATCHTHREADS <= TA_MAXDISPATCHTHREADS である必要があります。
TA_MAXDISPATCHTHREADS が指定されていない場合のデフォルト値は 1 です。

制限事項 : この属性を実行時に変更しても、実行中のサーバには反映されません。

TA_THREADSTACKSIZE: 0 <= *num* <= 2147483647

マルチスレッド・サーバの各ディスパッチ・スレッドに対して作成されるスタックのサイズ。このオプションは、TA_MAXDISPATCHTHREADS に 1 より大きい値が指定された場合のみサーバに影響を与えます。

この属性が指定されていない場合、または 0 が指定されている場合、デフォルトのスレッド・サイズが使用されます。デフォルト・サイズは、オペレーティング・システムのデフォルト・サイズです。ただし、この値がマルチスレッド BEA Tuxedo アプリケーション用に十分であることが分かっている場合は、BEA Tuxedo のデフォルト・サイズが使用されます。現在、BEA Tuxedo のデフォルト・スレッド・スタック・サイズは 1,024,000 です。

スレッド・スタック・サイズを上回った場合、サーバはコア・ダンプを行います。

制限事項 : この属性を実行時に変更しても、実行中のサーバには反映されません。

TA_CURDISPATCHTHREADS: 0 <= *num*

このサーバに対するアクティブなサービス・ディスパッチ・スレッドの現在の数。

TA_HWDISPATCHTHREADS: 0 <= *num*

このサーバを最後に再起動して以降に、このサーバに対して作成できるアクティブなサービス・ディスパッチ・スレッドの最大数。アイドル状態のサービス・スレッドのキャッシュを制御するパラメータを指定している場合、ここで指定する数とサービス呼び出しの数が異なる場合があります。

TA_NUMDISPATCHTHREADS: 0 <= *num*

このサーバを最後に再起動して以降に、このサーバに対して作成されたアクティブなサービス・ディスパッチ・スレッドの総数。

TA_RCMD: *string*[0..256] (BEA Tuxedo 8.0 以前では最大 78 バイト)

アプリケーション・サーバのシステムの再起動と並行して実行するアプリケーション指定のコマンド。

要求キューを共有するサーバ (つまり、T_SERVER:TA_RQADDR の値が同一のサーバ) は、この属性も同じ値に設定する必要があります。この属性値が異なる場合、キュー上のすべてのサーバに関連付けられる実行時値が、最初にアクティブにしたサーバによって確立されます。

制限事項 : この属性を実行時に変更すると、実行中のサーバおよび要求キューを共有しているほかのアクティブ・サーバに反映されます。ただし、変更されるのは選択したサーバのコンフィギュレーション・パラメータのみです。したがって、キューを共有するすべてのサーバについてこの属性を同じ値に設定しないと、その後サーバをアクティブする際の起動順序によってアプリケーションの動作が変わってしまいます。

注記　Windows 2000 システムでリダイレクトまたはパイプを選択する場合は、以下のいずれかの方法を使用してください。

- リダイレクトまたはパイプを、コマンド・ファイルまたはスクリプト内から行います。
- キュー・マネージャ管理プログラムからリダイレクトするには、コマンドの前に cmd を付けます。次に例を示します。
  ```
  cmd /c ipconfig > out.txt
  ```
- バイナリ実行可能ファイルの作成を選択する場合は、Windows API 関数の AllocConsole() を使用して、バイナリ実行可能ファイル内にコンソールを割り当てます。

TA_RESTART: "{Y | N}"

サーバを再起動可能 ("Y") または再起動不可能 ("N") に設定します。このサーバ・グループに対してサーバの移行 (T_DOMAIN:TA_OPTIONS/MIGRATE 属性および代替サイトによる T_GROUP:TA_LMID) を指定した場合は、TA_RESTART を "Y" に設定する必要があります。

要求キューを共有するサーバ (つまり、T_SERVER:TA_RQADDR の値が同一のサーバ) は、この属性も同じ値に設定する必要があります。この属性値が異なる場合、キュー上のすべてのサーバに関連付けられる実行時値が、最初にアクティブにしたサーバによって確立されます。

制限事項 : この属性を実行時に変更すると、実行中のサーバおよび要求キューを共有しているほかのアクティブ・サーバに反映されます。ただし、変更されるのは選択したサーバのコンフィギュレーション・パラメータのみです。したがって、キューを共有するすべてのサーバについてこの属性を同じ値に設定しないと、その後サーバをアクティブにする際の起動順序によってアプリケーションの動作が変わってしまいます。

TA_SEQUENCE: 1 <= *num* < 10,000

このサーバを、他のサーバに関連していつ起動 (tmboot(1)) またはシャットダウン (tmshutdown(1)) するかを指定します。TA_SEQUENCE 属性を指定しないか、無効な値を指定して T_SERVER オブジェクトを追加すると、10,000 以上で、かつその他の自動的に選択されたデフォルト値よりも大きい値が作成されます。サーバは、tmboot() によってシーケンス番号の昇順で起動され、tmshutdown() によって降順でシャットダウンされます。この属性を実行時に変更すると、tmboot() と tmshutdown() にのみ反映され、実行中のサーバが以降の tmshutdown() の呼び出しによってシャットダウンされる順序に影響します。

TA_SYSTEM_ACCESS: "{FASTPATH | PROTECTED}"

BEA Tuxedo システム・ライブラリが、このサーバ・プロセス内で BEA Tuxedo システムの内部テーブルにアクセスするために使用するモード。この属性の詳細については、T_DOMAIN:TA_SYSTEM_ACCESS を参照してください。

制限事項 : (1) この属性を実行時に変更しても、実行中のサーバには反映されません。(2) TA_SYSTEM_ACCESS を PROTECTED に設定しても、マルチスレッド・サーバには効果がない場合があります。これは、あるスレッドが BEA Tuxedo コードを実行しているときに (つまりスレッドが掲示板にアタッチされているとき) 別のスレッドがユーザ・コードを実行できるからです。BEA Tuxedo では、このような状況を防止することはできません。

TA_CONV: "{Y|N}"
会話型サーバ ("Y") または要求 / 応答型サーバ ("N") を指定します。

TA_REPLYQ: "{Y | N}"
サーバに対して別の応答キューを割り当てます (TA_REPLYQ == "Y")。応答を受信する必要のある MSSQ サーバでは、この属性を "Y" に設定する必要があります。

注記 Windows 2000 システムでリダイレクトまたはパイプする場合は、TA_RCMD 属性の説明の際に示した、いずれかの方法を使用する必要があります。

TA_RPPERM: 0001 <= *num* <= 0777
サーバの応答キューに対する UNIX システムのパーミッション。個別の応答キューが割り当てられない場合 (T_SERVER:TA_REPLYQ == "N")、TA_RPPERM は無視されます。

注記 Windows 2000 システムでリダイレクトまたはパイプする場合は、TA_RCMD 属性の説明の際に示した、いずれかの方法を使用する必要があります。

TA_RQADDR: *string*[0..30]
サーバの要求キューのシンボリック・アドレス。複数のサーバに対して同じ TA_RQADDR 属性値を指定すると、複数サーバ / 単一キュー (MSSQ) セットを定義できます。TA_RQADDR 属性値が同じサーバは、同じサーバ・グループに属している必要があります。

TA_RQPERM: 0001 <= *num* <= 0777
要求キューに対する UNIX システムのパーミッション。

制限事項 : UNIX システム固有の属性です。アプリケーションを実行しているプラットフォームが UNIX ベースでない場合、この属性は返されないことがあります。

TA_LMID:*LMID*
サーバを実行している現在の論理マシン。

TA_GENERATION: 1 <= *num* < 32,768
サーバの世代。サーバを最初に tmboot(1) で起動した場合や、TM_MIB(5) でアクティブにした場合、その世代は 1 に設定されます。サーバが異常終了して再起動されるごとに世代が増分されます。な

お、世代が T_SERVER:TA_MAXGEN を超え、T_SERVER:TA_GRACE の期限が切れると、サーバが再起動されて世代が 1 にリセットされます。

TA_PID: 1 <= *num*

サーバの UNIX システム・プロセス識別子。サーバが別々のマシンに存在する場合はプロセス識別子が重複しても構わないため、この属性は一意とはなりません。

制限事項 : UNIX システム固有の属性です。アプリケーションを実行しているプラットフォームが UNIX ベースでない場合、この属性は返されないことがあります。

TA_RPID: 1 <= *num*

サーバの応答キューに対する UNIX システムのメッセージ・キューの識別子。個別の応答キューが割り当てられない場合 (

制限事項 : UNIX システム固有の属性です。アプリケーションを実行しているプラットフォームが UNIX ベースでない場合、この属性は返されないことがあります。

TA_RQID: 1 <= *num*

サーバの要求キューに対する UNIX システムのメッセージ・キューの識別子。個別の応答キューが割り当てられない場合 (

制限事項 : UNIX システム固有の属性です。アプリケーションを実行しているプラットフォームが UNIX ベースでない場合、この属性は返されないことがあります。

TA_TIMERESTART: 1 <= *num*

T_SERVER:TA_LMID で time(2) システム・コールから返される 1970 年 1 月 1 日の 00:00:00 UTC から、このサーバが最後に起動または再起動された時点までの時間 (単位は秒)。

TA_TIMESTART: 1 <= *num*

T_SERVER:TA_LMID で time(2) システム・コールから返される 1970 年 1 月 1 日の 00:00:00 UTC から、このサーバが初めて起動された時点までの時間 (単位は秒)。サーバを再起動してもこの値はリセットされ

ませんが、T_SERVER:TA_MAXGEN を超え、T_SERVER:TA_GRACE の期限が切れると、この属性は再起動された時間にリセットされます。

TA_SICACHEENTRIESMAX: {"0"-"32767" | "DEFAULT"}
このマシンが保持するサービスおよびインターフェイスのキャッシュ・エントリの数。値を "0" にすると、このマシンではサービス・キャッシュが使用されなくなります。値を "DEFAULT" にした場合、このサーバに対する値は対応する T_MACHINE クラスのエントリによって決まります。

TA_SEC_PRINCIPAL_NAME: *string*[0..511]
BEA Tuxedo 7.1 以降が動作するアプリケーションで認証用に使用されるセキュリティ・プリンシパル名。この属性の最大文字数は、文字列の終端を表す NULL 文字を除いて 511 文字です。この属性に指定するプリンシパル名は、このサーバで実行されるシステム・プロセスの識別子として使用されます。

TA_SEC_PRINCIPAL_NAME は、コンフィギュレーションの階層のうち、T_DOMAIN クラス、T_MACHINE クラス、T_GROUP クラス、および T_SERVER クラスの 4 つのレベルのどこでも指定できます。特定のコンフィギュレーション・レベルでのプリンシパル名は、それより下位レベルで変更可能です。TA_SEC_PRINCIPAL_NAME がどのレベルでも指定されていない場合、アプリケーションのプリンシパル名にはこのドメインの TA_DOMAINID 文字列がデフォルトで設定されます。

TA_SEC_PRINCIPAL_NAME のほかにも、TA_SEC_PRINCIPAL_LOCATION と TA_SEC_PRINCIPAL_PASSVAR という属性があります。後の 2 つの属性は、アプリケーション起動時に、BEA Tuxedo 7.1 以降で動作するシステム・プロセスに対して復号化キーのオープンする処理に関係する属性です。特定のレベルで TA_SEC_PRINCIPAL_NAME のみが指定されている場合には、それ以外の 2 つの属性に長さゼロの NULL 文字列が設定されます。

TA_SEC_PRINCIPAL_LOCATION: *string*[0..511]
TA_SEC_PRINCIPAL_NAME に指定したプリンシパルの復号化 (秘密) キーを格納するファイルまたはデバイスの位置。この属性の最大文字数は、文字列の終端を表す NULL 文字を除いて 511 文字です。

TA_SEC_PRINCIPAL_LOCATION は、コンフィギュレーションの階層のうち、T_DOMAIN クラス、T_MACHINE クラス、T_GROUP クラス、および

T_SERVER クラスの 4 つのレベルのどこでも指定できます。この属性は、どのレベルで指定する場合でも TA_SEC_PRINCIPAL_NAME 属性と対になっている必要があり、それ以外の場合には無視されます。(TA_SEC_PRINCIPAL_PASSVAR はオプションです。これが指定されていない場合、システムによって長さゼロの NULL 文字列が設定されます。

TA_SEC_PRINCIPAL_PASSVAR: *string*[0..511]

TA_SEC_PRINCIPAL_NAME に指定したプリンシパルのパスワードを格納する変数。この属性の最大文字数は、文字列の終端を表す NULL 文字を除いて 511 文字です。

TA_SEC_PRINCIPAL_PASSVAR は、コンフィギュレーションの階層のうち、T_DOMAIN クラス、T_MACHINE クラス、T_GROUP クラス、および T_SERVER クラスの 4 つのレベルのどこでも指定できます。この属性は、どのレベルで指定する場合でも TA_SEC_PRINCIPAL_NAME 属性と対になっている必要があり、それ以外の場合には無視されます。(TA_SEC_PRINCIPAL_LOCATION はオプションです。これが指定されていない場合、システムによって長さゼロの NULL 文字列が設定されます。

初期化時は、TA_SEC_PRINCIPAL_PASSVAR に設定した復号化キーの各パスワードを管理者が入力する必要があります。管理者が入力したパスワードはシステム側で自動的に暗号化され、暗号化されたそれぞれのパスワードは対応するパスワード変数に割り当てられます。

TA_NUMCONV: 0 <= *num*

このサーバが tpconnect() を使用して開始した会話の数。

TA_NUMDEQUEUE: 0 <= *num*

このサーバが tpdequeue() を使用してキューからの取り出し操作を開始した回数。

TA_NUMENQUEUE: 0 <= *num*

このサーバが tpenqueue() を使用してキューへの登録操作を開始した回数。

TA_NUMPOST: 0 <= *num*

このサーバが tppost() を使用して開始したポストの数。

TA_NUMREQ: 0 <= *num*
: このサーバが tpcall() または tpacall() を使用して開始した要求の数。

TA_NUMSUBSCRIBE: 0 <= *num*
: このサーバが tpsubscribe() を使用して行ったサブスクリプションの数。

TA_NUMTRAN: 0 <= *num*
: このサーバが最後に起動または再起動されて以降に開始されたトランザクションの数。

TA_NUMTRANABT: 0 <= *num*
: このサーバが最後に起動または再起動されて以降にアボートされたトランザクションの数。

TA_NUMTRANCMT: 0 <= *num*
: このサーバが最後に起動または再起動されて以降にコミットされたトランザクションの数。

TA_TOTREQC: 0 <= *num*
: このサーバが完了した総要求数。会話型サーバの場合 (T_SERVER:TA_CONV == "Y")、この属性値は完了した着信会話数を示します。この実行時属性は、サーバの再起動時には保持されますが、サーバをシャットダウンすると失われます。

TA_TOTWORKL: 0 <= *num*
: このサーバが完了した負荷。会話型サーバの場合 (T_SERVER:TA_CONV == "Y")、この属性値は完了した着信会話の負荷を示します。この実行時属性は、サーバの再起動時には保持されますが、サーバをシャットダウンすると失われます。

TA_CLTLMID:*LMID*
: 要求元のクライアントまたはサーバの論理マシン。

  また、このフィールド要素は、シングルコンテキスト・サーバとマルチコンテキスト・サーバの両方で T_SERVERCTXT クラスに含まれます。

  要求元のクライアントまたはサーバとは、サーバが現在実行しているサービスを要求したプロセスです。このフィールドの値は、シングルコンテキスト・サーバに対してのみ有効です。マルチコンテキスト・サーバの場合は、プレースホルダとして NULL 文字列が返されます。

TA_CLTPID: 1 <= *num*

要求元のクライアントまたはサーバの UNIX システム・プロセス識別子。

また、このフィールド要素は、シングルコンテキスト・サーバとマルチコンテキスト・サーバの両方で T_SERVERCTXT クラスに含まれます。

このフィールドの値は、シングルコンテキスト・サーバに対してのみ有効です。マルチコンテキスト・サーバの場合は、プレースホルダとして 0 が返されます。

制限事項 : UNIX システム固有の属性です。アプリケーションを実行しているプラットフォームが UNIX ベースでない場合、この属性は返されないことがあります。

TA_CLTREPLY: "{Y | N}"

要求元のクライアントまたはサーバが応答を要求しているか ("Y")、いないか ("N") を指定します。

また、このフィールド要素は、シングルコンテキスト・サーバとマルチコンテキスト・サーバの両方で T_SERVERCTXT クラスに含まれます。

このフィールドの値は、シングルコンテキスト・サーバに対してのみ有効です。マルチコンテキスト・サーバの場合は、プレースホルダとして NULL 文字列が返されます。

TA_CMTRET: "{COMPLETE | LOGGED}"

このサーバの TP_COMMIT_CONTROL 特性の設定。

また、このフィールド要素は、シングルコンテキスト・サーバとマルチコンテキスト・サーバの両方で T_SERVERCTXT クラスに含まれます。

この特性の詳細については、ATMI 関数呼び出し tpscmt() の説明を参照してください。このフィールドの値は、シングルコンテキスト・サーバに対してのみ有効です。マルチコンテキスト・サーバの場合は、プレースホルダとして NULL 文字列が返されます。

TA_CURCONV: 0 <= *num*

このサーバが tpconnect() を使用して開始した会話のうち、現在もアクティブな会話の数。マルチコンテキスト・サーバの場合、このフィールドはすべてのサーバ・コンテキストの合計を表します。個々のサーバ・コンテキストの値は、T_SERVERCTXT クラスにあります。

TA_CUROBJECTS: 0 <= *num*

このサーバの掲示板のオブジェクト・テーブルで使用しているエントリの数。スコープはローカルです。

TA_CURINTERFACE: *string*[0..128]

このサーバで現在アクティブなインターフェイスの名前。スコープはローカルです。

TA_CURREQ: 0 <= *num*

このサーバが tpcall() または tpacall() を使用して開始した要求のうち、現在もアクティブな要求の数。マルチコンテキスト・サーバの場合、このフィールドはすべてのサーバ・コンテキストの合計を表します。個々のサーバ・コンテキストの値は、T_SERVERCTXT クラスにあります。

TA_CURRSERVICE: *string*[0. .15]

サーバが現在処理中のサービスがある場合、そのサービスの名前。

また、このフィールド要素は、シングルコンテキスト・サーバとマルチコンテキスト・サーバの両方で T_SERVERCTXT クラスに含まれます。

このフィールドの値は、シングルコンテキスト・サーバに対してのみ有効です。マルチコンテキスト・サーバの場合は、プレースホルダとして 0 が返されます。

TA_CURTIME: 1 <= *num*

T_SERVER:TA_LMID で time(2) システム・コールから返される 1970 年 1 月 1 日の 00:00:00 UTC から現在までの時間 (単位は秒)。この属性は、T_SERVER:TA_TIMESTART 属性値および T_SERVER:TA_TIMERESTART 属性値からの経過時間を算出するために使用できます。

TA_LASTGRP: 1 <= *num* < 30,000

最後に開始されたサービス要求またはこのサーバから開始された会話のサーバ・グループ番号 (T_GROUP:TA_GRPNO)。

また、このフィールド要素は、シングルコンテキスト・サーバとマルチコンテキスト・サーバの両方で T_SERVERCTXT クラスに含まれます。

このフィールドの値は、シングルコンテキスト・サーバに対してのみ有効です。マルチコンテキスト・サーバの場合は、プレースホルダとして 0 が返されます。

TA_SVCTIMEOUT: 0 <= *num*

このサーバが現在のサービス要求を処理するのに残された時間 (単位は秒)。

また、このフィールド要素は、シングルコンテキスト・サーバとマルチコンテキスト・サーバの両方で T_SERVERCTXT クラスに含まれます。

アクティブなサービスに対して 0 を指定すると、タイムアウト処理は行われません。詳細については、T_SERVICE:TA_SVCTIMEOUT を参照してください。このフィールドの値は、シングルコンテキスト・サーバに対してのみ有効です。マルチコンテキスト・サーバの場合は、プレースホルダとして 0 が返されます。

TA_TIMELEFT: 0 <= *num*

このサーバが現時点で待っている応答がタイムアウトするまでの残り時間 (単位は秒)。

また、このフィールド要素は、シングルコンテキスト・サーバとマルチコンテキスト・サーバの両方で T_SERVERCTXT クラスに含まれます。

タイムアウトは、トランザクション・タイムアウトまたはブロック・タイムアウトです。

このフィールドの値は、シングルコンテキスト・サーバに対してのみ有効です。マルチコンテキスト・サーバの場合は、プレースホルダとして 0 が返されます。

TA_TRANLEV: 0 <= *num*

このサーバの現在のトランザクション・レベル。

また、このフィールド要素は、シングルコンテキスト・サーバとマルチコンテキスト・サーバの両方で T_SERVERCTXT クラスに含まれます。

値がゼロの場合は、サーバが現在トランザクションに関与していないことを示します。このフィールドの値は、シングルコンテキスト・サーバに対してのみ有効です。マルチコンテキスト・サーバの場合は、プレースホルダとして 0 が返されます。

制限事項　なし

T_SERVERCTXT クラスの定義

概要 T_SERVERCTXT クラスは、アプリケーション内の各サーバ・ディスパッチ・コンテキストのコンフィギュレーション属性と実行時属性を表します。このクラスは、シングルコンテキスト・サーバおよびマルチコンテキスト・サーバの両方に対して定義されます。シングルコンテキスト・サーバでは、このクラスの値を T_SERVER クラスの一部として繰り返し使用します。T_SERVERCTXT クラスの属性は読み取り専用です。

これらの属性値によって、各サーバ・ディスパッチ・コンテキストに関連する統計値やリソースを実行時にトラッキングすることができます。

属性表

表 57TM_MIB(5): T_SERVERCTXT クラス定義の属性表

属性 (注 1)	タイプ	パーミッション	値	デフォルト値
TA_SRVGRP(k)	string	r--r--r--	*string*[1..30]	N/A
TA_SRVID(k)	long	r--r--r--	1 <= *num* < 30,001	N/A
TA_CONTEXTID(k)	long	r--r--r--	-2 <= *num* < 30,000	N/A
TA_CLTLMID	string	r--r--r--	*LMID*	N/A
TA_CLTPID	long	r--r--r--	1 <= *num*	N/A
TA_CLTREPLY	string	r--r--r--	"{Y | N}"	N/A
TA_CMTRET	string	R--R--R--	"{COMPLETE | LOGGED}"	N/A
TA_CURCONV	long	r--r--r--	0 <= *num*	N/A
TA_CURREQ	long	r--r--r--	0 <= *num*	N/A
TA_CURRSERVICE	string	r--r--r--	*string*[0..15]	N/A
TA_LASTGRP	long	r--r--r--	1 <= *num* <30,000	N/A
TA_SVCTIMEOUT	long	r--r--r--	0 <= *num*	N/A
TA_TIMELEFT	long	r--r--r--	0 <= *num*	N/A
TA_TRANLEV	long	r--r--r--	0 <= *num*	N/A

表 57TM_MIB(5): T_SERVERCTXT クラス定義の属性表

属性 (注 1)	タイプ	パーミッション	値	デフォルト値
(k)—GET キー・フィールド				

注 1 T_SERVERCTXT クラスのすべての属性はローカル属性です。

属性の意味

TA_SRVGRP: *string*[1..30]
サーバ・グループの論理名。また、サーバ・グループ名にはアスタリスク (*)、カンマ (,)、コロン (:) は使用できません。

TA_SRVID: 1 <= *num* < 30,001
サーバ・グループ内で一意なサーバ識別番号。

TA_CONTEXTID: 0 <= *num* < 30000
この特定のサーバ・コンテキストの識別子。

TA_CLTLMID:*LMID*
要求元のクライアントまたはサーバの論理マシン。要求元のクライアントまたはサーバとは、サーバが現在実行しているサービスを要求したプロセスです。

TA_CLTPID: 1 <= *num*
要求元のクライアントまたはサーバの UNIX システム・プロセス識別子。

制限事項 : UNIX システム固有の属性です。アプリケーションを実行しているプラットフォームが UNIX ベースでない場合、この属性は返されないことがあります。

TA_CLTREPLY: "{Y | N}"
要求元のクライアントまたはサーバが応答を要求しているか ("Y")、いないか ("N") を指定します。

TA_CMTRET: "{COMPLETE | LOGGED}"
このサーバの TP_COMMIT_CONTROL 特性の設定。この特性の詳細については、BEA Tuxedo ATMI 関数 tpscmt(3c) の説明を参照してください。

TA_CURCONV: 0 <= *num*
: このサーバが tpconnect() を使用して開始した会話のうち、現在もアクティブな会話の数。

TA_CURREQ: 0 <= *num*
: このサーバが tpcall() または tpacall() を使用して開始した要求のうち、現在もアクティブな要求の数。

TA_CURRSERVICE: *string*[0..15]
: サーバが現在処理中のサービスがある場合、そのサービスの名前。

TA_LASTGRP: 1 <= *num* < 30,000
: 最後に開始されたサービス要求またはこのサーバから開始された会話のサーバ・グループ番号 (T_GROUP:TA_GRPNO)。

TA_SVCTIMEOUT: 0 <= *num*
: このサーバが現在のサービス要求を処理するのに残された時間 (単位は秒)。
アクティブなサービスに対して 0 を指定すると、タイムアウト処理は行われません。詳細については、T_SERVICE:TA_SVCTIMEOUT を参照してください。

TA_TIMELEFT: 0 <= *num*
: このサーバが現時点で待っている応答がタイムアウトするまでの残り時間 (単位は秒)。タイムアウトは、トランザクション・タイムアウトまたはブロック・タイムアウトです。

TA_TRANLEV: 0 <= *num*
: このサーバの現在のトランザクション・レベル。値がゼロの場合は、サーバが現在トランザクションに関与していないことを示します。

制限事項　なし

T_SERVICE クラスの定義

概要 T_SERVICE クラスは、アプリケーション内のサービスのコンフィギュレーション属性を表します。これらの属性値によって、コンフィギュレーション済みのサービスを識別できます。T_SERVICE オブジェクトは、T_SVCGRP クラスの一部としてコンフィギュレーションされていないサービスに対するアクティブ化時のコンフィギュレーション属性を提供します。アプリケーション内でアクティブなサービスの実行時情報は、T_SVCGRP クラスでのみ取得可能です。T_SERVICE クラスを実行時に更新しても、通常はアクティブな T_SVCGRP オブジェクトには反映されません (TA_ROUTINGNAME は例外です)。

T_SERVICE クラスおよび T_SVCGRP クラスは、アプリケーション内のサービス名に対するアクティブ化時の属性設定を定義します。サーバを初めてアクティブにしたため、または tpadvertise() を呼び出したために新たなサービスがアクティブ化 (宣言) されると、サービス開始時に使用する属性値は以下の順序で決定されます。

1. サービス名とサービス・グループが一致するコンフィギュレーション済みの T_SVCGRP オブジェクトが存在する場合、宣言されたサービスの初期コンフィギュレーションには、このオブジェクトで定義された属性が使用されます。
2. 1 に該当せず、かつサービス名が一致するコンフィギュレーション済みの T_SERVICE オブジェクトが存在する場合、宣言されたサービスの初期コンフィギュレーションには、このオブジェクトで定義された属性が使用されます。
3. 1 および 2 に該当せず、かつ TA_SERVICENAME 属性値が一致するコンフィギュレーション済みの T_SVCGRP オブジェクトが見つかった場合、宣言されたサービスの初期コンフィギュレーションには、最初に見つかったオブジェクトが使用されます。
4. 上記のいずれにも該当しない場合、宣言されたサービスの初期コンフィギュレーションには、サービス属性のシステム・デフォルト値が使用されます。

アプリケーション・サービスに対するコンフィギュレーション属性の指定はすべて省略可能です。つまり、コンフィギュレーション値を指定しない場合、サービスのアクティブ化時にサーバが宣言したサービスには確立済みのデフォルト・サービス値が使用されます (サービスのアクティブ化時に属性

値を識別する方法については上記を参照のこと)。サーバが提供するサービス名は実行時に作成されます (buildserver(1) を参照)。ただし、サーバ・オブジェクトに指定されたコマンド行オプションによってオーバライドされる場合もあります (T_SERVER:TA_CLOPT および servopts(5) を参照)。

属性表

表 58TM_MIB(5): T_SERVICE クラス定義の属性表

属性	タイプ	パーミッション	値	デフォルト値
TA_SERVICENAME(r)(*)	string	ru-r--r--	*string*[1..15]	N/A
TA_STATE(k)	string	rw-r--r--	GET: "{ACT | INA}" SET: "{NEW | INV}"	N/A N/A
TA_AUTOTRAN	string	rwyr--r--	"{Y | N}"	"N"
TA_LOAD	long	rwyr--r--	1 <= *num* < 32,768	50
TA_PRIO	long	rwyr--r--	1 <= *num* < 101	50
TA_SVCTIMEOUT	long	rwyr--r--	0 <= *num*	0
TA_TRANTIME	long	rwyr--r--	0 <= *num*	30
TA_BUFTYPE	string	rw-r--r--	*string*[1..256]	"ALL"
TA_ROUTING 名前	string	rwxr--r--	*string*[0..15]	""
TA_SIGNATURE_REQUIRED	string	rwxr--r--	"{Y|N}"	"N"
TA_ENCRYPTION_REQUIRED	string	rwxr--r--	"{Y|N}"	"N"

(k)—GET キー・フィールド
(r)— オブジェクトの作成に必要なフィールド (SET TA_STATE NEW)
(*)—GET/SET キー、SET 操作では 1 つ以上必要

属性の意味　TA_SERVICENAME: *string*[1..15]
サービス名。

TA_STATE:

GET: "{ACTive | INActive}"

GET 操作は、選択した T_SERVICE オブジェクトのコンフィギュレーション情報を検索します。以下に示す状態は、GET 要求への応答で返される TA_STATE の意味を示します。

ACTive	T_SERVICE オブジェクトが定義済みで、TA_SERVICENAME の値が一致する T_SVCGRP オブジェクトの少なくとも 1 つはアクティブです。
INActive	T_SERVICE オブジェクトが定義済みで、TA_SERVICENAME の値が一致する T_SVCGRP オブジェクトはアクティブではありません。

SET: "{NEW | INValid}"

SET 操作は、選択した T_SERVICE オブジェクトのコンフィギュレーション情報を更新します。以下に示す状態は、SET 要求で設定される TA_STATE の意味を示します。以下に示されていない状態は設定できません。

NEW	アプリケーションに対する T_SERVICE オブジェクトを作成します。状態の変更は INValid 状態でのみ可能です。正常に終了すると、オブジェクトの状態は INActive になります。制限事項 : コンフィギュレーションされていないサービスが、それらを宣言しているサーバによってアクティブなままになっている場合があります。この場合、新規に T_SERVICE オブジェクトを作成することはできません。
unset	既存の T_SERVICE オブジェクトを変更します。この組み合わせは INValid 状態では使用できません。正常終了した場合、オブジェクトの状態は変わりません。
INValid	アプリケーションに対する T_SERVICE オブジェクトを削除します。状態の変更は、INActive 状態でのみ可能です。正常終了すると、オブジェクトの状態は INValid になります。

TA_AUTOTRAN: "{Y | N}"
要求がまだトランザクション・モードでない場合に、このサービスに対するサービス要求メッセージを受信すると、トランザクションが自動的に開始されます ("Y" を指定した場合)。制限事項 : この属性を実行時に変更しても、アクティブな T_SVCGRP オブジェクトには反映されません。

TA_LOAD: 1 <= *num* < 32,768
この T_SERVICE オブジェクトは、システムに対する負荷を設定します。サービスの負荷は、ロード・バランシングのために使用します。つまり、すでに負荷が大きいキューは、新規の要求ではあまり選択されません。サービスの負荷は、T_DOMAIN:TA_LDBAL 属性値が "Y" に設定されている場合にのみ有効です。

制限事項 : この属性を実行時に変更しても、アクティブな T_SVCGRP オブジェクトには反映されません。

TA_PRIO: 1 <= *num* < 101
この T_SERVICE オブジェクトは、指定された優先順位でキューから取り出されます。複数のサービス要求がサービス・キューで待機している場合、優先順位の高い要求から処理されます。

制限事項 : この属性を実行時に変更しても、アクティブな T_SVCGRP オブジェクトには反映されません。

TA_SVCTIMEOUT: 0 <= *num*
このサービス名に対する要求を処理する際の時間制限 (単位は秒)。このサービスのサービス要求を処理するサーバは、要求の処理が指定した時間制限値を超えると異常終了します。この属性を 0 に設定すると、サービスは異常終了しません。

制限事項 : この属性を実行時に変更しても、アクティブな T_SVCGRP オブジェクトには反映されません。この属性値は、BEA Tuxedo リリース 4.2.2 以前のサイトでは適用されません。

TA_TRANTIME: 0 <= *num*
この T_SERVICE オブジェクト用に自動的に開始されたトランザクションのトランザクション・タイムアウト値 (単位は秒)。サービスの T_SERVICE:TA_AUTOTRAN 属性値が "Y" である場合に、トランザクション・モードでない要求を受信すると、トランザクションが自動的に開始されます。

制限事項 : この属性を実行時に変更しても、アクティブな T_SVCGRP オブジェクトには反映されません。

TA_BUFTYPE: "*type1*[*:subtype1*[,*subtype2 . . .*]][*;type2*[*:subtype3*[,...]]]..."
このサービスで受け付けるデータ・バッファのタイプおよびサブタイプのリスト。最大で 32 のタイプとサブ・タイプの組み合わせを使用できます。BEA Tuxedo システムに用意されているデータ・バッファのタイプには、FML と FML32 (FML バッファ用)、XML (XML バッファ用)、VIEW、VIEW32、X_C_TYPE、または X_COMMON (FML VIEW 用)、STRING (NULL で終了する文字配列用)、および CARRAY または X_OCTET (送信時に符号化も復号化もされない文字配列用) があります。これらのタイプのうち、VIEW、VIEW32、X_C_TYPE、および X_COMMON にはサブタイプがあります。VIEW サブタイプは、サービスが期待する特定の VIEW の名前を指定します。アプリケーションのタイプとサブタイプも追加できます（tuxtypes(5) を参照)。サブタイプを持つバッファ・タイプでは、サブタイプに "*" を指定して、該当サービスが関連するバッファ・タイプのすべてのサブタイプを受け付けるようにできます。

1 つのサービスが解釈できるバッファ・タイプは一定のものに限られています。すなわち、そのバッファ・タイプ・スイッチにあるものしか解釈できません (tuxtypes(5) を参照)。TA_BUFTYPE 属性値が ALL に設定されている場合、そのサービスはそのバッファ・タイプ・スイッチにあるバッファ・タイプをすべて受け付けます。

タイプ名は 8 文字以下、サブタイプ名は 16 文字以下で指定することができます。タイプおよびサブタイプの名前には、セミコロン (;)、カンマ (,)、コロン (:)、アスタリスク (*) は使用できません。

制限事項 : この属性値は、このサービス名を持つアプリケーション・サービスの各インスタンスでサポートする必要のあるバッファ・タイプを表します。この属性値は、サービスのアクティブ化の際に処理されるため、一致するサービス名を持つアクティブな T_SVCGRP オブジェクトが存在しない場合にのみ更新できます。

TA_ROUTINGNAME: *string*[0..15]
この T_SERVICE オブジェクトは、指定されたルーティング基準名を保持します。この属性をアクティブに変更すると、関連するすべての T_SVCGRP オブジェクトに反映されます。

TA_SIGNATURE_REQUIRED: "{Y | N}"

"Y" に設定すると、このサービスのすべてのインスタンスで、その入力メッセージ・バッファのデジタル署名が必要となります。指定しない場合、デフォルト値の "N" が設定されます。この属性は、BEA Tuxedo 7.1 以降が動作するアプリケーションにのみ適用されます。

TA_SIGNATURE_REQUIRED は、コンフィギュレーションの階層のうち、T_DOMAIN クラス、T_MACHINE クラス、T_GROUP クラス、および T_SERVICE クラスの 4 つのレベルのどこでも指定できます。特定のレベルで SIGNATURE_REQUIRED に "Y" を設定すると、下位レベルで動作するすべてのプロセスに署名が必要となります。

TA_ENCRYPTION_REQUIRED: "{Y | N}

"Y" に設定すると、このサービスのすべてのインスタンスで暗号化された入力メッセージ・バッファが必要となります。指定しない場合、デフォルト値の "N" が設定されます。この属性は、BEA Tuxedo 7.1 以降が動作するアプリケーションにのみ適用されます。

TA_ENCRYPTION_REQUIRED は、コンフィギュレーションの階層のうち、T_DOMAIN クラス、T_MACHINE クラス、T_GROUP クラス、および T_SERVICE クラスの 4 つのレベルのどこでも指定できます。特定のレベルで TA_ENCRYPTION_REQUIRED に Y を設定すると、下位レベルで動作するすべてのプロセスに暗号化が必要となります。

制限事項　なし

T_SVCGRP クラスの定義

概要 T_SVCGRP クラスは、アプリケーション内のサービスまたはグループのコンフィギュレーション属性と実行時属性を表します。これらの属性値によって、コンフィギュレーション済みのサービスやグループを識別したり、各オブジェクトに関連する統計値やリソースを実行時にトラッキングしたりできます。

T_SERVICE クラスおよび T_SVCGRP クラスは、アプリケーション内のサービス名に対するアクティブ化時の属性設定を定義します。サーバを初めてアクティブにしたため、または tpadvertise() を呼び出したために新たなサービスがアクティブ化 (宣言) されると、サービス開始時に使用する属性値は以下の順序で決定されます。

1. サービス名とサービス・グループが一致するコンフィギュレーション済みの T_SVCGRP オブジェクトが存在する場合、宣言されたサービスの初期コンフィギュレーションには、このオブジェクトで定義された属性が使用されます。
2. 1 に該当せず、かつサービス名が一致するコンフィギュレーション済みの T_SERVICE オブジェクトが存在する場合、宣言されたサービスの初期コンフィギュレーションには、このオブジェクトで定義された属性が使用されます。
3. 1 および 2 に該当せず、かつ TA_SERVICENAME 属性値が一致するコンフィギュレーション済みの T_SVCGRP オブジェクトが見つかった場合、宣言されたサービスの初期コンフィギュレーションには、最初に見つかったオブジェクトが使用されます。
4. 上記のいずれにも該当しない場合、宣言されたサービスの初期コンフィギュレーションには、サービス属性のシステム・デフォルト値が使用されます。

アプリケーション・サービス対するコンフィギュレーション属性の指定はすべて省略可能です。つまり、コンフィギュレーション値を指定しない場合、サービスのアクティブ化時にサーバが宣言したサービスには確立済みのデフォルト・サービス値が使用されます (サービスのアクティブ化時に属性値を識別する方法については上記を参照のこと)。サーバが提供するサービス名は実行時に作成されます (buildserver(1) を参照)。ただし、サーバ・オブジェクトに指定されたコマンド行オプションによってオーバライドされる場合もあります (T_SERVER:TA_CLOPT および servopts(5) を参照)。

いったん T_SVCGRP オブジェクトがアクティブになると、T_SVCGRP クラスでのみ表現されるようになります。サービスを提供するグループ内に複数のサーバがある場合、特定のサービス名 / グループ名の組み合わせは、実行時に複数の T_SVCGRP クラスに関連付けられます。

属性表

表 59TM_MIB(5): T_SVCGRP クラス定義の属性表

属性	タイプ	パーミッション	値	デフォルト値
TA_SERVICENAME(r)(*)	string	ru-r--r--	*string*[1..15]	N/A
TA_SRVGRP(r)(*)	string	ru-r--r--	*string*[1..30]	N/A
TA_GRPNO(k)	long	r--r--r--	1 <= *num* < 30,000	N/A
TA_STATE(k)	string	rwxr-xr--	GET: "{ACT | INA | SUS | PAR}" SET: "{NEW | INV | ACT | INA | SUS}"	N/A N/A
TA_AUTOTRAN	string	rwxr-xr--	"{Y | N}"	"N"
TA_LOAD	long	rwxr-xr--	1 <= *num* < 32,768	50
TA_PRIO	long	rwxr-xr--	1 <= *num* < 101	50
TA_SVCTIMEOUT	long	rwyr-yr--	0 <= *num*	0
TA_TRANTIME	long	rwxr-xr--	0 <= *num*	30
TA_LMID(k)	string	R--R--R--	*LMID*	N/A
TA_RQADDR(*)	string	R--R--R--	*string*[1..30]	N/A
TA_SRVID(*)	long	R--R--R--	1 <= *num* < 30,001	N/A
TA_SVCRNAM	string	R-XR-XR--	*string*[1..15]	(注 2)
TA_BUFTYPE	string	r--r--r--	*string*[1..256]	N/A
TA_ROUTINGNAME	string	r--r--r--	*string*[0..15]	N/A
TA_SVCTYPE(k)	string	r--r--r--	"{APP | CALLABLE | SYSTEM}"	"APP"
T_SVCGRP クラス：ローカル属性				
TA_NCOMPLETED	long	R-XR-XR--	0 <= *num*	N/A
TA_NQUEUED	long	R--R--R--	0 <= *num* < 32,768	N/A

表 59TM_MIB(5): T_SVCGRP **クラス定義の属性表**

属性	タイプ	パーミッション	値	デフォルト値
(k)—GET キー・フィールド (r)— オブジェクトの作成に必要なフィールド (SET TA_STATE NEW) (*)—GET/SET キー、SET 操作では 1 つ以上必要 (注 1)				

注 1 アドレス指定するオブジェクトを一意に識別するには、このクラスでの SET 操作で十分なキー・フィールドを指定する必要があります。オブジェクトがアクティブな場合は、TA_RQADDR または TA_SRVID を指定した TA_SERVICENAME または TA_SRVGRP キー・フィールドを追加する必要があります。アクティブなオブジェクトを変更すると、そのオブジェクト、および関連するコンフィギュレーション・レコードに反映されますが、同じコンフィギュレーション・レコードから実行時属性を生成した他のアクティブ・オブジェクトには反映されません。

注 2 この属性値を指定しない場合は、デフォルトで TA_SERVICENAME となります。

属性の意味

TA_SERVICENAME: *string*[1..15]

サービス名。

TA_SRVGRP: *string*[1..30]

サーバ・グループ名。また、サーバ・グループ名にはアスタリスク (*)、カンマ (,)、コロン (:) は使用できません。サービスのアクティブ化時に使用するサービス属性の検索順序については、前述の T_SVCGRP の概要を参照してください。

TA_GRPNO: 1 <= *num* < 30,000

サーバ・グループ番号。

TA_STATE:

GET: "{ACTive | INActive | SUSpended | PARtitioned}"

GET 操作は、選択した T_SVCGRP オブジェクトのコンフィギュレーション情報および実行時情報を取得します。以下に示す状態は、GET 要求への応答で返される TA_STATE の意味を示します。

ACTive	T_SVCGRP 属性と TA_SRVID 属性の戻り値によって示されたサーバで、T_SVCGRP オブジェクトがアクティブになっています。返された属性値は、サービスの現在の実行時インスタンスを示し、一時的に更新されてもコンフィギュレーション・インスタンスには反映されません。
INActive	T_SVCGRP オブジェクトが定義済みで、非アクティブな状態です。
SUSpended	T_SVCGRP オブジェクトが定義済みで、現在は中断されています。このサービスは、この状態のアプリケーションからはアクセスできません。この状態は、パーミッションの決定においては ACTive と同等です。
PARtitioned	T_SVCGRP オブジェクトが定義済みでアクティブですが、現在はアプリケーションのマスタ・サイトから分断されています。このサービスは、この状態のアプリケーションからはアクセスできません。この状態は、パーミッションの決定においては ACTive と同等です。

SET: "{NEW | INValid | ACTive | INActive | SUSpended}"

SET 操作は、選択した T_SVCGRP オブジェクトのコンフィギュレーション情報および実行時情報を更新します。サービス・オブジェクトを実行時に変更すると、複数のアクティブなサーバに反映される場合がありますので注意してください。以下に示す状態は、SET 要求で設定される TA_STATE の意味を示します。以下に示されていない状態は設定できません。

NEW	アプリケーションに対する T_SVCGRP オブジェクトを作成します。状態の変更は INValid 状態でのみ可能です。正常に終了すると、オブジェクトの状態は INActive になります。 制限事項 : コンフィギュレーションされていないサービスが、それらを宣言しているサーバによってアクティブなままになっている場合があります。この場合、このサービス・クラスの状態は ACTive で、これを更新することはできません。
unset	既存の T_SVCGRP オブジェクトを変更します。この組み合わせは INValid 状態では使用できません。正常終了した場合、オブジェクトの状態は変わりません。
INValid	アプリケーションに対する T_SVCGRP オブジェクトを削除します。状態の変更は、INActive 状態でのみ可能です。正常終了すると、オブジェクトの状態は INValid になります。
ACTive	T_SVCGRP オブジェクトをアクティブ化 (宣言) します。状態の変更は、INActive 状態、SUSpended 状態、または INValid 状態でのみ可能です。この状態変更では、TA_SRVID または TA_RQADDR のいずれかを指定しなければなりません。この状態遷移に対するパーミッションの決定に際しては、アクティブなオブジェクトのパーミッション (--x--x--x) が考慮されます。正常に終了すると、オブジェクトの状態は ACTive になります。 制限事項 : サービス名 (TA_SERVICENAME) が予約文字列 "." で始まる場合、状態を変更することはできません。

INActive	T_SVCGRP オブジェクトを非アクティブにします。状態の変更は、SUSpended 状態でのみ可能です。正常に終了すると、オブジェクトの状態は INActive (コンフィギュレーション済みのエントリの場合) または INValid (コンフィギュレーションされていないエントリの場合) になります。 制限事項 : サービス名 (TA_SERVICENAME) が予約文字列 "drq; で始まる場合、状態を変更することはできません。
SUSpended	T_SVCGRP オブジェクトを中断します。状態の変更は、ACTive 状態でのみ可能です。正常に終了すると、オブジェクトの状態は SUSpended になります。 制限事項 : サービス名 (TA_SERVICENAME) が予約文字列 "_" で始まる場合、状態を変更することはできません。

TA_AUTOTRAN: "{Y | N}"

要求がまだトランザクション・モードでない場合に、このサービスに対するサービス要求メッセージを受信すると、トランザクションが自動的に開始されます ("Y" を指定した場合)。

TA_LOAD: 1 <= *num* < 32,768

この T_SVCGRP オブジェクトは、システムに対する負荷を設定します。サービスの負荷は、ロード・バランシングのために使用します。つまり、すでに負荷が大きいキューは、新規の要求ではあまり選択されません。

TA_PRIO: 1 <= *num* < 101

この T_SVCGRP オブジェクトは、指定された優先順位でキューから取り出されます。複数のサービス要求がサービス・キューで待機している場合、優先順位の高い要求から処理されます。

TA_SVCTIMEOUT: 0 <= *num*

このサービス名に対する要求を処理する際の時間制限 (単位は秒)。このサービスのサービス要求を処理するサーバは、要求の処理が指定した時間制限値を超えると異常終了します。この属性を 0 に設定すると、サービスは異常終了しません。

制限事項 : この属性値は、BEA Tuxedo リリース 4.2.2 以前のサイトでは適用されません。

TA_TRANTIME: 0 <= *num*

このT_SVCGRP オブジェクト用に自動的に開始されたトランザクションのトランザクション・タイムアウト値 (単位は秒)。サービスのT_SVCGRP:TA_AUTOTRAN 属性値が "Y" である場合に、トランザクション・モードでない要求を受信すると、トランザクションが自動的に開始されます。

TA_LMID:*LMID*

このサービスを提供するアクティブなサーバを実行している現在の論理マシン。

TA_RQADDR: *string*[1..30]

このサービスを提供するアクティブなサーバの要求キューのシンボリック・アドレス。この属性の詳細については、T_SERVER:TA_RQADDR を参照してください。

TA_SRVID: 1 <= *num* < 30,001

このサービスを提供するアクティブなサーバのサーバ・グループ内での一意なサーバ識別番号。この属性の詳細については、T_SERVER:TA_SRVID を参照してください。

TA_SVCRNAM: *string*[1..15]

このサービスに対する要求を処理するために割り当てられた対応するサーバ内の関数名。SET 要求時、サーバがそのシンボル・テーブルを使用して関数名を関数にマップし、正常にサービスを宣言できなければなりません。状況によっては (たとえば、サーバで直接 tpadvertise() を呼び出す場合)、ACTive なサービス・オブジェクトの関数名は不明となり、属性値として文字列 "?" が返されます。

制限事項 : この属性は、状態を INActive から ACTive に変更する場合にのみ設定できます。

TA_BUFTYPE: *string*[1..256]

このサービスで使用できるコンフィギュレーション済みのバッファ・タイプ。

制限事項 : この属性は、対応する T_SERVICE クラス・オブジェクト経由でのみ設定可能です。

TA_ROUTINGNAME: *string*[0..15]
ルーティング基準の名前。

制限事項 : この属性は、対応する T_SERVICE クラス・オブジェクト経由でのみ設定可能です。

TA_NCOMPLETED: 0 <= *num*
検索された ACTive 状態または SUSpended 状態のオブジェクトがアクティブ化 (宣言) されて以降に完了したサービス要求の数。

制限事項 : この属性は、T_DOMAIN:TA_LDBAL 属性値が "Y" に設定されている場合にのみ返されます。

TA_SVCTYPE: "{APP | CALLABLE | SYSTEM}"
サービスのタイプ。APP は、アプリケーション定義のサービス名を示します。CALLABLE は、システム提供の呼び出し可能サービスを示します。SYSTEM は、システム提供でシステム呼び出し可能なサービスを示します。SYSTEM サービスは、アプリケーション・クライアントおよびサーバから直接アクセスすることはできません。GET キー・フィールドとして使用する場合は、'|' で区切られたリストを使用して、1 つの要求に対するサービス・グループ・エントリの複数のタイプを検索します。デフォルトでは、APP サービスのみが検索されます。

このサービスのキューに現時点で登録されている要求の数。この属性の値は、要求がキューに登録されると増加し、サーバがキューから要求を取り出すと減少します。制限事項 : この属性は、T_DOMAIN:TA_LDBAL 属性値が "Y" に設定されている場合にのみ返されます。

TA_NQUEUED: 0 <= *num* < 32,768
このサービスのキューに現時点で登録されている要求の数。この属性の値は、要求がキューに登録されると増加し、サーバがキューから要求を取り出すたびに減少します。

制限事項 : この属性は、T_DOMAIN:TA_LDBAL 属性値が "Y" に設定されている場合にのみ返されます。

制限事項 なし

T_TLISTEN クラスの定義

概要 T_TLISTEN クラスは、分散アプリケーションで使用する BEA Tuxedo システムのリスナ・プロセスの実行時属性を表します。

属性表

表 60TM_MIB(5): T_TLISTEN クラス定義の属性表

属性	タイプ	パーミッション	値	デフォルト値
TA_LMID(k)	string	R--R--R--	*LMID*	N/A
TA_STATE(k)	string	R--R--R--	GET: "{ACT|INA}" SET:N/A	N/A N/A
(k)—GET キー・フィールド				

属性の意味

TA_LMID:*LMID*

論理マシン識別子。

TA_STATE:

GET: "{INActive | ACTive}"

T_TLISTEN 操作は、選択した T_CONN オブジェクトの実行時情報を検索します。以下に示す状態は、GET 要求への応答で返される TA_STATE の意味を示します。

INActive	T_CLIENT オブジェクトは非アクティブです。
ACTive	T_TLISTEN オブジェクトはアクティブです。

SET:

SET 操作は、このクラスでは使用できません。この属性は、対応する T_SERVICE クラス・オブジェクト経由でのみ設定可能です。

制限事項 このクラスは、tpadmcall() インターフェイスでは利用できません。

T_TLOG クラスの定義

概要 T_TLOG クラスは、トランザクション・ログのコンフィギュレーション属性と実行時属性を表します。このクラスを使用すると、アプリケーション内のログの作成、削除、移行などを実行できます。

属性表

表 61TM_MIB(5): T_TLOG **クラス定義の属性表**

属性 (注 1)	タイプ	パーミッション	値	デフォルト値
TA_LMID(*)	string	r--r--r--	*LMID*	N/A
TA_STATE(k)	string	r-xr-xr--	GET: "{ACT | INA | WAR}" SET: "WAR"	N/A N/A
TA_TLOGCOUNT	long	r-xr-xr--	1 <= *num*	N/A
TA_TLOGINDEX	long	r-xr-xr--	0 <= *num*	N/A
TA_GRPNO(k)	long	r--r--r--	1 <= *num* < 30,000	(注 2)
TA_TLOGDATA	string	r-xr-xr--	*string*[1..256]	(注 2)

(k)—GET キー・フィールド
(*)—GET/SET キー。SET 操作では 1 つ以上必要

注 1 T_TLOG クラスのすべての属性はローカル属性です。
注 2 T_TLOG クラスの各オブジェクトで、1 つ以上の TA_GRPNO 属性および TA_TLOGDATA 属性が返されます。特定のオブジェクトに属する各属性の値は、TA_TLOGINDEX で始まる TA_TLOGCOUNT のオカレンス数です。

属性の意味

TA_LMID:*LMID*

トランザクション・ログの論理マシン識別子。

TA_STATE:

GET: "{ACTive | INActive | WARmstart}"

GET 操作は、選択した T_TLOG オブジェクトのコンフィギュレーション情報および実行時情報を取得します。以下に示す状態は、GET 要求への応答で返される TA_STATE の意味を示します。

ACTive	トランザクション・ログが存在し、サイトで調整されたトランザクションに対するコミット・レコードをアクティブに記録しています。これは、関連のある T_MACHINE オブジェクトがアクティブであることを意味します。
INACtive	トランザクション・ログが存在しますが、現在は非アクティブです。この状態は、関連のある T_MACHINE オブジェクトが非アクティブであることを意味し、サイトで tlisten(1) プロセスを実行中の場合にのみ返されます。それ以外の場合、サイトは接続不可能で、オブジェクトは返されません。
WARmstart	トランザクション・ログが存在し、現在アクティブでウォームスタート処理としてマークされています。ウォームスタート処理は、次のサーバ・グループをサイトで開始するときに発生します。この状態は、パーミッションの決定においては ACTive と同等です。

SET: "{WARmstart}"

SET 操作は、選択した T_TLOG オブジェクトのコンフィギュレーション情報および実行時情報を更新します。以下に示す状態は、SET 要求で設定される TA_STATE の意味を示します。以下に示されていない状態は設定できません。

unset	T_TLOG オブジェクトを変更します。変更は、ACTive 状態でのみ可能です。正常終了した場合、オブジェクトの状態は変わりません。このクラスで可能なオブジェクト変更は、トランザクション・ログへの追加のみです。この場合、TA_TLOGINDEX および TA_TLOGCOUNT は追加される TA_TLOGDATA を示します。
WARmstart	T_TLOG オブジェクトのウォームスタートを開始します。状態の変更は、ACTive 状態でのみ可能です。正常終了すると、オブジェクトの状態は WARmstart になります。

TA_TLOGCOUNT: 1 <= *num*

カウント、検索、または追加されたトランザクション・ログ・データ・レコード (TA_TLOGDATA) の数。この属性は、状態変更を指定した SET 操作では無視されます。状態変更を指定しない有効な SET 操作の場合、この属性はアクティブなトランザクション・ログに追加されるログ・レコードの数を示します。TA_GRPNO または TA_TLOGDATA を指定しない GET 操作は、使用中のログ・レコードの数を返します。TA_GRPNO のみを指定した GET 操作は、使用中のログ・レコードの数と、指定したグループに対応するコーディネータ・グループを返します。("") に設定された TA_TLOGDATA のみを指定した GET 操作は、使用中のログ・レコードの数を返し、これらのログ・レコードに対応する TA_TLOGDATA 属性と TA_GRPNO 属性の配列を設定します。TA_GRPNO と ("") に設定された TA_TLOGDATA の両方を指定した GET 操作は、使用中のログ・レコードの数と、指定されたグループに一致するコーディネータ・グループを返し、これらのログ・レコードに対応する TA_TLOGDATA 属性と TA_GRPNO 属性の配列を設定します。

TA_TLOGINDEX: 0 <= *num*

このオブジェクトに対応する最初のオブジェクトに固有な属性値 (TA_GRPNO および TA_TLOGDATA) のインデックス。

TA_GRPNO: 1 <= *num* < 30,000

トランザクション・コーディネータのグループ番号。

TA_TLOGDATA: *string*[1..256]

フォーマット済みのトランザクション・ログ・エントリ。この属性値は、直接解釈するのではなく、サーバ・グループの移行の一部であるログ・レコードの移行手段としてのみ使用してください。

制限事項 なし

T_TRANSACTION クラスの定義

概要 T_TRANSACTION クラスは、アプリケーション内のアクティブなトランザクションの実行時属性を表します。

属性表

表 62TM_MIB(5): T_TRANSACTION **クラス定義の属性表**

属性 (注 1)	タイプ	パーミッション	値	デフォルト値
TA_COORDLMID(k)	string	R--R--R--	*LMID*	N/A
TA_LMID(k)	string	R--R--R--	*LMID*	N/A
TA_TPTRANID(*)	string	R--R--R--	*string*[1..78]	N/A
TA_XID(*)	string	R--R--R--	*string*[1..78]	N/A
TA_STATE(k)	string	R-XR-XR--	GET: "{ACT | ABY | ABD | COM | REA | DEC | SUS}" SET: "ABD"	N/A N/A
TA_TIMEOUT	long	R--R--R--	1 <= *num*	N/A
TA_GRPCOUNT	long	R--R--R--	1 <= *num*	N/A
TA_GRPINDEX	long	R--R--R--	0 <= *num*	N/A
TA_GRPNO	long	R--R--R--	1 <= *num* < 30,000	(注 2)
TA_GSTATE	long	R-XR-XR--	GET: "PREP|PABT|PCOM" SET:"{HCO|HAB}"	N/A N/A

(k)—GET キー・フィールド
(*)—GET/SET キー。SET 操作では 1 つ以上必要

注 1 T_TRANSACTION クラスのすべての属性はローカル属性です。
注 2 T_TRANSACTION クラスの各オブジェクトで、1 つ以上の TA_GRPNO 属性および TA_GSTATE 属性が返されます。特定のオブジェクトに属する各属性の値は、TA_GRPINDEX で始まる TA_GRPCOUNT のオカレンス数です。

属性の意味

TA_COORDLMID:*LMID*

トランザクションを調整するサーバ・グループの論理マシン識別子。

TA_LMID:*LMID*

検索マシンの論理マシン識別子。トランザクション属性は基本的にサイトにローカルで、トランザクション管理サーバ (TMS) により共通トランザクション識別子で調整されます。

TA_TPTRANID: *string*[1..78]

tpsuspend() から返され、文字列表現にマップされるトランザクション識別子。等号比較の場合を除き、ユーザはこのフィールドのデータを直接解釈することはできません。

TA_XID: *string*[1..78]

tx_info() から返され、文字列表現にマップされるトランザクション識別子。等号比較の場合を除き、ユーザはこのフィールドのデータを直接解釈することはできません。

TA_STATE:

GET: "{ACTive | ABortonlY | ABorteD | COMcalled | REAdy | DECided | SUSpended}"

GET 操作は、選択した T_TRANSACTION オブジェクトの実行時情報を検索します。以下に示す状態は、GET 要求への応答で返される TA_STATE の意味を示します。同一のグローバル・トランザクションに付属する別個のオブジェクトは、トランザクション識別子は同じですが状態は異なることがあります。一般的に、コーディネータ・サイトで示される状態 (TA_COORDLMID) がトランザクションの本当の状態です。例外は、コーディネータ・サイト以外のサイトが、トランザクションの状態を ABortonlY に遷移させる条件を通知した場合です。この遷移は、最終的にはコーディネータ・サイトに伝達され、トランザクションがロールバックされます。ただし、この変更がすぐにはコーディネータ・サイトに反映されないこともあります。すべての状態は、パーミッションの決定においては ACTive と同等です。

PREPrepare	トランザクションの処理中に xa_end (TMSUSPEND) を呼び出したサーバがトランザクション・グループに含まれており、コミット処理が開始されていることを示します。この状態は、xa_end (TMSUSPEND) を呼び出したすべてのサーバが xa_end (TMSUCESS) を呼び出してグループの状態が READy になるまで、またはターゲット・サーバのいずれかがトランザクションをロールバックしてグループの状態が PostABorT または ABorteD のいずれかになるまで続きます。
PostABorT	サーバが xa_end (TPFAIL) を呼び出しているのに、TMS がまだ xa_rollback() を呼び出していないことを示します。つまり、xa_end (TMSUSPEND) を呼び出した別のサーバが、関連の CORBA オブジェクトをクリーン・アップするための通知を TMS から受けていることを示します。
PostCOMmit	実装されていません。

SET: "{ABorteD}"

SET 操作は、選択した T_TRANSACTION オブジェクトの実行時情報を更新します。以下に示す状態は、SET 要求で設定される TA_STATE の意味を示します。以下に示されていない状態は設定できません。

unset	既存の T_TRANSACTION オブジェクトを変更します。この組み合わせは READy 状態でのみ可能で、個別のグループの状態を更新するために使用します (後述の TA_GSTATE を参照)。正常終了した場合、オブジェクトの状態は変わりません。
ABorteD	アプリケーションに対する T_TRANSACTION をアボートします。状態の変更は、ACTive 状態、ABortonlY 状態、または COMcalled 状態でのみ可能です。正常終了すると、オブジェクトの状態は ABorteD になります。

TA_TIMEOUT: 1 <= *num*

検索サイトでトランザクションがタイムアウトになるまでの残り時間 (単位は秒)。この属性値はトランザクション状態 (TA_STATE) が ACTive である場合にのみ返されます。

TA_GRPCOUNT: 1 <= *num*

検索サイトから返された情報により、トランザクション内の参加者として識別されたグループの数。

TA_GRPINDEX: 1 <= *num*

このグループに対応する最初のオブジェクトに固有な属性値 (TA_GRPNO および TA_GSTATE) のインデックス。

TA_GRPNO: 1 <= *num* < 30,000

参加しているグループのグループ番号。

TA_GSTATE:

GET: "{ACTive | ABorteD | ReaDOnly | REAdy | HCOmmit | HABort | DONe}"

GET 操作は、指定グループに付属する T_TRANSACTION オブジェクトのうち、選択したオブジェクトの実行時情報を検索します。以下に示す状態は、GET 要求への応答で返される TA_GSTATE の意味を示します。これら以外の状態は返されません。同一のグローバル・トランザクションに付属する別個のオブジェクトは、トランザクション識別子は同じですが、個別のグループの状態は異なることがあります。一般的に、グループのサイトで示される状態が、トランザクションに参加しているグループの本当の状態です。例外は、コーディネータ・サイトがトランザクションをアボートすることを決定し、各参加グループの状態を ABorteD に設定した場合です。この遷移はグループのサイトに伝達され、トランザクション内のグループの作業がロールバックされます。ただし、すぐには反映されないことがあります。

ACTive	トランザクションは、指定したグループ内でアクティブです。
ABorteD	トランザクションはロールバックされるものと識別され、指定されたグループのロールバックが開始されました。

ReaDOnly	このグループでは、2 フェーズ・コミットの第 1 フェーズが正常に完了し、リソース・マネージャ上で読み取り操作のみが実行されています。したがって、コミットの第 2 フェーズを実行する必要はありません。
REAdy	グループは 2 フェーズ・コミットの第 1 フェーズを正常に完了し、コミット可能な状態です。
HCOmmit	このグループはヒューリスティックにコミットされました。これは、トランザクションの最終結果と一致する場合と一致しない場合があります。
HABort	このグループはヒューリスティックにロールバックされました。これは、トランザクションの最終結果と一致する場合と一致しない場合があります。
DONe	このグループでは、2 フェーズ・コミットの第 2 フェーズが完了しています。

SET: "{HCOmmit | HABort}"

SET 操作は、選択した T_TRANSACTION オブジェクト内で送信された要求の最初のグループの実行時情報を更新します。以下に示す状態は、SET 要求で設定される TA_GSTATE の意味を示します。以下に示されていない状態は設定できません。状態遷移は、グループのサイトを表すオブジェクト (TA_LMID) 内で実行される場合にのみ可能です。

HCOmmit	グループの作業を、指定したトランザクションの一部としてヒューリスティックにコミットします。状態の変更は、TA_GSTATE が REAdy かつ TA_STATE が REAdy で、指定したグループがコーディネータ・サイト上にない場合にのみ可能です。正常終了すると、オブジェクトの状態は HCOmmit になります。

HABort	グループの作業を、指定したトランザクションの一部としてヒューリスティックにロールバックします。状態の変更は、TA_GSTATE が ACTive または REAdy で、かつ TA_STATE が REAdy であり、指定したグループがコーディネータ・サイト上にない場合にのみ可能です。正常終了すると、オブジェクトの状態は HABort になります。

制限事項　なし

T_ULOG クラスの定義

概要 T_ULOG クラスは、アプリケーション内の userlog() ファイルの実行時属性を表します。

属性表

表 63TM_MIB(5): T_ULOG クラス定義の属性表

属性 (注 1)	タイプ	パーミッション	値	デフォルト値
TA_LMID(k)	string	R--R--R--	*LMID*	(注 2)
TA_PMID(x)	string	R--R--R--	*string*[1..30]	(注 2)
TA_MMDDYY(k)	long	R--R--R--	*mmddyy*	現在の日付
TA_STATE	string	R--R--R--	GET: "ACT" SET:N/A	N/A N/A
TA_ULOGTIME(k)	long	R--R--R--	*hhmmss*	000000
TA_ENDTIME(k)	long	K--K--K--	*hhmmss*	235959
TA_ULOGLINE(k)	long	R--R--R--	1 <= *num*	1
TA_ULOGMSG(x)	string	R--R--R--	*string*[1..256]	N/A
TA_TPTRANID(k)	string	R--R--R--	*string*[1..78]	N/A
TA_XID(k)	string	R--R--R--	*string*[1..78]	N/A
TA_PID(k)	long	R--R--R--	1 <= *num*	N/A
TA_THREADID	integer	r--r--r--	0 <= *num*	NA
TA_CONTEXTID(k)	long	r--r--r--	-2 <= *num* < 30,000	N/A
TA_SEVERITY(x)	string	R--R--R--	*string*[1..30]	N/A
TA_ULOGCAT(x)	string	R--R--R--	*string*[1..30]	N/A
TA_ULOGMSGNUM(k)	long	R--R--R--	1 <= *num*	N/A

表 63TM_MIB(5): T_ULOG クラス定義の属性表 (続き)

属性 (注 1)	タイプ	パーミッション	値	デフォルト値
TA_ULOGPROCNM(x)	string	R--R--R--	*string*[1..30]	N/A

(k)—GET キー・フィールド
(x)— 正規表現の GET キー・フィールド

注 1 T_ULOG クラスのすべての属性はローカル属性です。
注 2 TA_LMID は、システムがどのアプリケーション・ログ・ファイルにアクセスするかを決定する際に必要なフィールドです。返されたレコードを、指定したマシン上で動作するプロセスから生成されたレコードに限定するために使用するのではありません。ネットワーク接続されたファイル・システムを使用して複数のマシンがログ・ファイルを共有する場合、キー・フィールドとして特定の値を指定しても、複数の TA_LMID 値が返されます。同様の理由で、TA_PMID は要求を特定のマシンに指定する際に考慮されるではなく、どのレコードを返すか決定するために使用されます。この条件下では、TA_PMID を正規表現のキー・フィールドとして使用すると便利です。

属性の意味

TA_LMID:*LMID*
検索マシンの論理マシン識別子。

TA_PMID: *string*[1..30]
物理マシン識別子。

TA_MMDDYY: *mmddyy*
見つかった、またはアクセスしたユーザ・ログ・ファイルの日付。

TA_STATE:

GET: "{ACTive}"
GET 操作は、選択した T_ULOG オブジェクトの実行時情報を検索します。以下に示す状態は、GET 要求への応答で返される TA_STATE の意味を示します。

ACTive	返されたオブジェクトには、指定論理マシン上の既存のユーザ・ログ・ファイルが反映されています。

SET:
SET 操作は、このクラスでは使用できません。

TA_ULOGTIME: *hhmmss*
このオブジェクトで表わされるユーザ・ログ・メッセージの時刻。この属性値は、時間を 10,000 倍した値に分を 100 倍した値を加算し、最後に秒を加算して計算されます。キー・フィールドとして使用する場合、この属性はメッセージに対するアクセスの時間範囲の開始時間を表します。

TA_ENDTIME: *hhmmss*
この userlog ファイルへのアクセス時に GET 操作で考慮する最新時刻。

TA_ULOGLINE: 1 <= *num*
ユーザ・ログ・ファイル内で返されたまたは要求されたユーザ・ログ・メッセージの行番号。検索時にキー・フィールドとして使用した場合、この値はログ・ファイル内の開始行を示します。

TA_ULOGMSG: *string*[1..256]
ユーザ・ログ・ファイルに記録されているユーザ・ログ・メッセージのテキスト全体。

TA_TPTRANID: *string*[1..78]
tpsuspend() から返されたトランザクション識別子。等号比較の場合を除き、ユーザはこのフィールドのデータを直接解釈することはできません。トランザクションに関連付けられていないメッセージでは、この属性値として長さが 0 の文字列が検索されます。

TA_XID: *string*[1..78]
tx_info() から返されたトランザクション識別子。等号比較の場合を除き、ユーザはこのフィールドのデータを直接解釈することはできません。トランザクションに関連付けられていないメッセージでは、この属性値として長さが 0 の文字列が検索されます。

TA_PID: 1 <= *num*
ユーザ・ログ・メッセージを生成したクライアントまたはサーバのプロセス識別子。

TA_THREADID: 0 <= *num*
このユーザ・ログ・メッセージを記述したスレッドの識別子。

TA_CONTEXTID: -2 <= *num* < 30,000
この特定のアプリケーション関連の識別子。

TA_SEVERITY: *string*[1..30]
メッセージの重要度レベルが指定されている場合は、その重要度レベル。

TA_ULOGCAT: *string*[1..30]
メッセージがメッセージ・カタログから生成された場合は、そのカタログの名前。

TA_ULOGMSGNUM: 1 <= *num*
メッセージがカタログから生成された場合は、そのカタログのメッセージ番号。

TA_ULOGPROCNM: *string*[1..30]
ユーザ・ログ・メッセージを生成したクライアントまたはサーバのプロセス名。

制限事項 検索は、対応する T_MACHINE オブジェクトも ACTive である場合にのみ行われます。

このクラスの検索は、TA_LMID を指定して指示する必要があります。ワークステーション・クライアントが書き込んだログ・レコードの検索は、クライアントが使用したログ・ファイルが、当該アプリケーションの T_MACHINE クラスで定義されたマシンの 1 つで共有されている場合のみ利用可能です。それ以外の場合、このクラスではログ・レコードを使用できません。

このクラスに対する検索が正常に完了しなかった場合は、常に値が 1 の TA_MORE が返されます。この値は、送信された要求に関する情報が他にも存在することのみを示します。

TM_MIB(5) に関する追加情報

診断

TM_MIB(5) への接続時には、2 つの一般的なタイプのエラーがユーザに返される場合があります。1 つは、管理要求に対する応答を検索する 3 つの ATMI 関数 (tpcall()、tpgetrply()、および tpdequeue()) が返すエラーです。これらのエラーは、該当するリファレンス・ページの記述に従って解釈されます。

ただし、要求がその内容に対応できるシステム・サービスに正常にルーティングされても、システム・サービス側でその要求を処理できないと判断されると、アプリケーション・レベルのサービス障害としてエラーが返されます。このような場合、tpcall() と tpcall() は、tpgetrply() を TPESVCFAIL に設定してエラーを返し、以下のようにエラーの詳細を示す TA_ERROR、TA_STATUS、および TA_BADFLD フィールドと一緒に、元の要求を含む応答メッセージを返します。TMQFORWARD(5) サーバ経由でシステムに転送された要求に対してサービス障害が発生すると、元の要求で識別される異常終了キューに障害応答メッセージが追加されます (TMQFORWARD に対して -d オプションが指定されたと見なされる)。

管理要求の処理中にサービス・エラーが発生すると、TA_STATUS という FM32 フィールドにエラーの内容を説明したテキストが設定され、TA_ERROR という FM32 フィールドにはエラーの原因 (下記参照) を示す値が設定されます。エラー・コードは、いずれもマイナスであることが保証されています。

[other]
: すべてのコンポーネント MIB に共通のその他のエラー・リターン・コードは、MIB(5) リファレンス・ページに指定されています。これらのエラーは、ここに定義する TM_MIB(5) 固有のエラー・コードと相互に排他関係にあることが保証されています。

以下の診断コードは TA_ERROR で戻されるもので、管理要求が正常に完了したことを示します。これらのコードはマイナスでないことが保証されています。

[other]
: すべてのコンポーネント MIB に共通のその他のリターン・コードは、MIB(5) リファレンス・ページに指定されています。これらのコードは、ここに定義する TM_MIB(5) 固有のリターン・コードと相互に排他関係にあることが保証されています。

相互運用性 このリファレンス・ページで定義されているヘッダ・ファイルおよびフィールド・テーブルは、BEA Tuxedo リリース 6.1 以降で利用できます。これらのヘッダやテーブルで定義するフィールドはリリースが変わっても変更されません。ただし、以前のリリースで定義されていない新しいフィールドが追加される場合があります。AdminAPI には、要求を作成するために必要なヘッダ・ファイルとフィールド・テーブルがあれば、どのサイトからでもアクセスできます。

リリースが異なるサイト (共に BEA Tuxedo リリース 6.1 以降) を相互運用する場合、当該リリースの MIB マニュアル・ページに定義されるように、旧サイト上の情報はアクセスおよび更新可能で、以降のリリースで利用可能な情報のサブセットとなります。

移植性 BEA Tuxedo システムの MIB を使用した管理作業をサポートするために必要な既存の FML32 および ATMI 関数、さらにこのリファレンス・ページに定義するヘッダ・ファイルとフィールド・テーブルは、すべてのサポート対象ネイティブ・プラットフォームとワークステーション・プラットフォームで使用可能です。

使用例 このセクションでは、`tpadmcall()` と `tpcall()` を使用して 2 つのノードを持つアプリケーションをコンフィギュレーション、アクティブ化、問い合わせ、および非アクティブ化するためのコードを抜粋しています。ローカル環境に応じて値が変わる部分には変数名を使用します。たとえば、`TUXCONFIG` は 2 つの要素からなる文字ポインタの配列で、それぞれ要素によってマシン上の `TUXCONFIG` ファイルの絶対パス名を識別します。

フィールド・テーブル 属性フィールド識別子にアクセスするには、フィールド・テーブル `tpadm` が必要です。そのためには、次のようにシェルで入力します。

```
$ FIELDTBLS=tpadm
$ FLDTBLDIR=${TUXDIR}/udataobj
$ export FIELDTBLS FLDTBLDIR
```

ヘッダ・ファイル 次のヘッダ・ファイルがインクルードされます。

```
#include <atmi.h>
#include <fml32.h>
#include <tpadm.h>
```

ライブラリ

```
${TUXDIR}/lib/libtmib.a, ${TUXDIR}/lib/libqm.a,
${TUXDIR}/lib/libtmib.so.<rel>, ${TUXDIR}/lib/libqm.so.<rel>,
${TUXDIR}/lib/libtmib.lib
```

buildclient を使用するときには、ライブラリを手動でリンクする必要があります。この場合は、-L${TUXDIR}/lib -ltmib -lqm を指定する必要があります。

初期コンフィギュレーション

以下のコードでは、FML32 バッファを作成して設定しています。この FML32 バッファは、処理のために tpadmcall() に渡されます。この例ではさらに、tpadmcall() のリターン・コードの解釈を示しています。ここで示す要求により、アプリケーションの初期コンフィギュレーションが作成されます。

```
 /* バッファの割り当ておよび初期化 */
 ibuf = (FBFR32 *)tpal loc("FML32", NULL, 4000);
 obuf = (FBFR32 *)tpalloc("FML32", NULL, 4000);
/* 要求タイプを定義する MIB(5) 属性を設定 */
 Fchg32(ibuf, TA_OPERATION, 0, "SET", 0);
 Fchg32(ibuf, TA_CLASS, 0, "T_DOMAIN", 0);
 Fchg32(ibuf, TA_STATE, 0, "NEW", 0);
 /* T_DOMAIN クラス・オブジェクトに設定する TM_MIB(5) 属性を設定 */
 Fchg32(ibuf, TA_OPTIONS, 0, "LAN,MIGRATE", 0);
 Fchg32(ibuf, TA_IPCKEY, 0, (char *)&ipckey, 0);
 Fchg32(ibuf, TA_MASTER, 0, "LMID1", 0);
 Fchg32(ibuf, TA_MODEL, 0, "MP", 0);
 /* TA_MASTER T_MACHINE クラス・オブジェクトの TM_MIB(5) 属性を設定 */
 Fchg32(ibuf, TA_LMID, 0, "LMID1", 0);
 Fchg32(ibuf, TA_PMID, 0, pmid[0], 0);
 Fchg32(ibuf, TA_TUXCONFIG, 0, tuxconfig[0], 0);
 Fchg32(ibuf, TA_TUXDIR, 0, tuxdir[0], 0);
 Fchg32(ibuf, TA_APPDIR, 0, appdir[0], 0);
 Fchg32(ibuf, TA_ENVFILE, 0, envfile[0], 0);
 Fchg32(ibuf, TA_ULOGPFX, 0, ulogpfx[0], 0);
 Fchg32(ibuf, TA_BRIDGE, 0, "/dev/tcp", 0);
 Fchg32(ibuf, TA_NADDR, 0, naddr[0], 0);
 Fchg32(ibuf, TA_NLSADDR, 0, nlsaddr[0], 0);
 /* tpadmcall() を使用して処理を実行 */
 if (tpadmcall(ibuf, obuf, 0) 0) {
 fprintf(stderr, "tpadmcall failed: %s\n", tpstrerror(tperrno));
 /* 追加のエラー処理 */
 }
```

2 つ目のマシンの追加

以下のコードでは、前のセクションで割り当てたバッファを再利用して要求バッファを作成しています。以下に示す要求により、先に確立したコンフィギュレーションに別のマシンが追加されます。

```
/* 要求バッファをクリア */ Finit32(ibuf, Fsizeof32(ibuf));
/* 要求タイプを定義する MIB(5) 属性を設定 */
```

```
 Fchg32(ibuf, TA_OPERATION, 0, "SET", 0);
 Fchg32(ibuf, TA_CLASS, 0, "T_MACHINE", 0);
 Fchg32(ibuf, TA_STATE, 0, "NEW", 0);
 /* T_MACHINE クラス・オブジェクトに設定する TM_MIB(5) 属性を設定 */
 Fchg32(ibuf, TA_LMID, 0, "LMID2", 0);
 Fchg32(ibuf, TA_PMID, 0, pmid[1], 0);
 Fchg32(ibuf, TA_TUXCONFIG, 0, tuxconfig[1], 0);
 Fchg32(ibuf, TA_TUXDIR, 0, tuxdir[1], 0);
 Fchg32(ibuf, TA_APPDIR, 0, appdir[1], 0);
 Fchg32(ibuf, TA_ENVFILE, 0, envfile[1], 0);
 Fchg32(ibuf, TA_ULOGPFX, 0, ulogpfx[1], 0);
 Fchg32(ibuf, TA_BRIDGE, 0, "/dev/tcp", 0);
 Fchg32(ibuf, TA_NADDR, 0, naddr[1], 0);
 Fchg32(ibuf, TA_NLSADDR, 0, nlsaddr[1], 0);

 tpadmcall(...) /* 詳細なエラー処理については上記の例を参照のこと */
```

2 つ目のマシンのバックアップ・マスタの作成

既存のバッファを再利用して、新たにコンフィギュレーションした 2 つ目のマシンをこのアプリケーションのバックアップ・マスタ・サイトとして識別します。

```
/* 要求バッファをクリア */ Finit32(ibuf, Fsizeof32(ibuf));

/* 要求タイプを定義する MIB(5) 属性を設定 */
 Fchg32(ibuf, TA_OPERATION, 0, "SET", 0);
 Fchg32(ibuf, TA_CLASS, 0, "T_DOMAIN", 0);

 /* Set TM_MIB(5) T_DOMAIN attributes changing *
 Fchg32(ibuf, TA_MASTER, 0, "LMID1,LMID2", 0);

 tpadmcall(...); /* 詳細なエラー処理については上記の例を参照 */
```

2 つのサーバ・グループの追加

バッファを再利用して、コンフィギュレーション済みのアプリケーションにサーバ・グループを 1 つずつ追加する要求を 2 つ作成します。2 つ目の要求は、単に既存の入力バッファの必要なフィールドを変更するためのものです。

```
/* 要求バッファをクリア */ Finit32(ibuf, Fsizeof32(ibuf));

/* 要求タイプを定義する MIB(5) 属性を設定 */
 Fchg32(ibuf, TA_OPERATION, 0, "SET", 0);
 Fchg32(ibuf, TA_CLASS, 0, "T_GROUP", 0);
 Fchg32(ibuf, TA_STATE, 0, "NEW", 0);

/* 1 つ目のグループを定義する MIB(5) 属性を設定 */
 Fchg32(ibuf, TA_SRVGRP, 0, "GRP1", 0);
 Fchg32(ibuf, TA_GRPNO, 0, (char *)&grpno[0], 0);
```

```
 Fchg32(ibuf, TA_LMID, 0, "LMID1,LMID2", 0);

 tpadmcall(...); /* 詳細なエラー処理については上記の例を参照 */

/* 2 つ目のグループを定義する MIB(5) 属性を設定 */
 Fchg32(ibuf, TA_SRVGRP, 0, "GRP2", 0);
 Fchg32(ibuf, TA_GRPNO, 0, (char *)&grpno[1], 0);
 Fchg32(ibuf, TA_LMID, 0, "LMID2,LMID1", 0);

 tpadmcall(...); /* 詳細なエラー処理については上記の例を参照 */
```

グループごとにサーバを 1 つずつ追加

割り当て済みのバッファを再利用し、グループごとに 1 つのサーバをコンフィギュレーション済みのアプリケーションに追加します。

```
/* 要求バッファをクリア */ Finit32(ibuf, Fsizeof32(ibuf));

/* 要求タイプを定義する MIB(5) 属性を設定 */
 Fchg32(ibuf, TA_OPERATION, 0, "SET", 0);
 Fchg32(ibuf, TA_CLASS, 0, "T_SERVER", 0);
 Fchg32(ibuf, TA_STATE, 0, "NEW", 0);

/* 1 つ目のサーバを定義する MIB(5) 属性を設定 */
 Fchg32(ibuf, TA_SRVGRP, 0, "GRP1", 0);
 Fchg32(ibuf, TA_SRVID, 0, (char *)&srvid[0], 0);
 Fchg32(ibuf, TA_SERVERNAME, 0, "ECHO", 0)

 tpadmcall(...); /* 詳細なエラー処理については上記の例を参照 */

/* 2 つ目のサーバを定義する MIB(5) 属性を設定 */
 Fchg32(ibuf, TA_SRVGRP, 0, "GRP2", 0);
 Fchg32(ibuf, TA_SRVID, 0, (char *)&srvid[1], 0);

 tpadmcall(...); /* 詳細なエラー処理については上記の例を参照 */
```

ルーティング基準の追加

ルーティング基準の定義を追加します。ルーティング基準は、tpcall() インターフェイスを使用して同様の操作を行うことで、動作中のアプリケーションに動的に追加することもできます。

```
/* 要求バッファをクリア */ Finit32(ibuf, Fsizeof32(ibuf));

/* 要求タイプを定義する MIB(5) 属性を設定 */
 Fchg32(ibuf, TA_OPERATION, 0, "SET", 0);
 Fchg32(ibuf, TA_CLASS, 0, "T_ROUTING", 0);
 Fchg32(ibuf, TA_STATE, 0, "NEW", 0);

/* ルーティング基準を定義する MIB(5) 属性を設定 */
 Fchg32(ibuf, TA_ROUTINGNAME, 0, "ECHOROUTE", 0);
```

```
 Fchg32(ibuf, TA_BUFTYPE, 0, "FML", 0);
 Fchg32(ibuf, TA_FIELD, 0, "LONG_DATA", 0);
 Fchg32(ibuf, TA_RANGES, 0, "MIN-100:GRP1,100-MAX:GRP2", 26);

 tpadmcall(...); /* 詳細なエラー処理については上記の例を参照 */
```

サービス定義の追加

上記で定義したルーティング基準に、宣言されたサービス名をマップするサービス・オブジェクトを定義します。

```
/* 要求バッファをクリア */ Finit32(ibuf, Fsizeof32(ibuf));

/* 要求タイプを定義する MIB(5) 属性を設定 */
 Fchg32(ibuf, TA_OPERATION, 0, "SET", 0);
 Fchg32(ibuf, TA_CLASS, 0, "T_SERVICE", 0);
 Fchg32(ibuf, TA_STATE, 0, "NEW", 0);

/* サービス・エントリを定義する MIB(5) 属性を設定 */
 Fchg32(ibuf, TA_SERVICENAME, 0, "ECHO", 0);
 Fchg32(ibuf, TA_ROUTINGNAME, 0, "ECHOROUTE", 0);

 tpadmcall(...); /* 詳細なエラー処理については上記の例を参照 */
```

マスタ・サイトの管理プロセスのアクティブ化

T_DOMAIN クラス・オブジェクトの状態を ACTIVE に設定して、マスタ・サイトの管理プロセス (DBBL、BBL、BRIDGE) をアクティブにします。

```
/* 要求バッファをクリア */ Finit32(ibuf, Fsizeof32(ibuf));

/* 要求タイプを定義する MIB(5) 属性を設定 */
 Fchg32(ibuf, TA_OPERATION, 0, "SET", 0);
 Fchg32(ibuf, TA_CLASS, 0, "T_DOMAIN", 0);
 Fchg32(ibuf, TA_STATE, 0, "ACT", 0);

 tpadmcall(...); /* 詳細なエラー処理については上記の例を参照 */
```

アクティブなアプリケーション管理への切り替え

これでアプリケーションがアクティブになりました。次は、アプリケーションに参加して、tpcall() インターフェイスを使用して AdminAPI 要求を作成します。

```
/* システムがアクティブなので、システムに管理者として参加 */ tpinfo =
(TPINIT *)tpalloc("TPINIT", NULL, TPINITNEED(0));
 sprintf(tpinfo->usrname, "appadmin");
 sprintf(tpinfo->cltname, "tpsysadm");
 if (tpinit(tpinfo) < 0) {
 fprintf(stderr, "tpinit() failed: %s\n", tpstrerror(tperrno));
 /* 追加のエラー処理 */
 }
```

```
 /* バッファを型付きバッファとして再初期化 */
 Finit32(ibuf, Fsizeof32(ibuf));
 Finit32(obuf, Fsizeof32(obuf));
```

残りのアプリケーションのアクティブ化

アプリケーションの残り部分をアクティブにします。管理者ユーザは、要求の TA_FLAGS 属性にある TMIB_NOTIFY フラグを設定して、各サーバを起動しようとする直前または直後に任意通知型メッセージを送信するよう要求します。この例では、tpcall() からのエラー・リターンの処理を示しています。

```
/* 要求バッファをクリア */ Finit32(ibuf, Fsizeof32(ibuf));

/* 要求タイプを定義する MIB(5) 属性を設定 */
 Fchg32(ibuf, TA_OPERATION, 0, "SET", 0);
 Fchg32(ibuf, TA_CLASS, 0, "T_MACHINE", 0);
 Fchg32(ibuf, TA_STATE, 0, "RAC", 0);

/* マシンを識別する MIB(5) 属性を設定 */
 Fchg32(ibuf, TA_LMID, 0, "LMID1", 0);

/* /AdminAPI を呼び出して結果を解釈 */
 if (tpcall(".TMIB", (char *)ibuf, 0, (char **)&obuf, &olen, 0) < 0) {
 fprintf(stderr, "tpcall failed: %s\n", tpstrerror(tperrno));
 if (tperrno == TPESVCFAIL) {
 Fget32(obuf,TA_ERROR,0,(char *)&ta_error,NULL);
 ta_status = Ffind32(obuf, TA_STATUS, 0, NULL);
 fprintf(stderr, "Failure: %ld, %s\n",
 ta_error, ta_status);

 /* 追加のエラー処理 */
 }
```

サーバ状態の問い合わせ

アクティブなサーバの状態に関する問い合わせを生成します。

```
/* 要求バッファをクリア */ Finit32(ibuf, Fsizeof32(ibuf));

/* 要求タイプを定義する MIB(5) 属性を設定 */
 Fchg32(ibuf, TA_OPERATION, 0, "GET", 0);
 Fchg32(ibuf, TA_CLASS, 0, "T_SERVER", 0);
 flags = MIB_LOCAL;
 Fchg32(ibuf, TA_FLAGS, 0, (char *)&flags, 0);

/* マシンを識別する MIB(5) 属性を設定 */
 Fchg32(ibuf, TA_SRVGRP, 0, "GRP1", 0);
 Fchg32(ibuf, TA_SRVID, 0, (char *)&srvid[0], 0);

 tpcall(...); /* 詳細なエラー処理については上記の例を参照 */
```

アプリケーションの非アクティブ化

各マシンの状態を INACTIVE に設定してアプリケーションを非アクティブ化します。この操作でも TMIB_NOTIFY フラグを使用できます。

```
/* 要求バッファをクリア */ Finit32(ibuf, Fsizeof32(ibuf));

 /* 最初にリモート・マシンをシャットダウン */
/* 要求タイプを定義する MIB(5) 属性を設定 */
 Fchg32(ibuf, TA_OPERATION, 0, "SET", 0);
 Fchg32(ibuf, TA_CLASS, 0, "T_MACHINE", 0);
 Fchg32(ibuf, TA_LMID, 0, "LMID2", 0);
 Fchg32(ibuf, TA_STATE, 0, "INA", 0);

 tpcall(....); /* 詳細なエラー処理については上記の例を参照 */

 /* つづいてマスタ・マシンのアプリケーション・サーバをシャットダウン */
 flags = TMIB_APPONLY;
 Fchg32(ibuf, TA_FLAGS, 0, (char *)&flags, 0);
 Fchg32(ibuf, TA_LMID, 0, "LMID1", 0);

 tpcall(...); /* 詳細なエラー処理については上記の例を参照 */

 /* アクティブなアプリケーション・アクセスを終了 */
 tpterm();

/* 最後にマスタ管理プロセスを終了 */
 Finit32(ibuf, Fsizeof32(ibuf));
 Fchg32(ibuf, TA_OPERATION, 0, "SET", 0);
 Fchg32(ibuf, TA_CLASS, 0, "T_DOMAIN", 0);
 Fchg32(ibuf, TA_STATE, 0, "INA", 0);

 tpadmcall(...); /* 詳細なエラー処理については上記の例を参照 */
```

ファイル

${TUXDIR}/include/tpadm.h, ${TUXDIR}/udataobj/tpadm

関連項目

tpacall(3c)、tpalloc(3c), tpcall(3c)、tpdequeue(3c), tpenqueue(3c)、tpgetrply(3c)、tprealloc(3c)、FML 関数の紹介、Fadd、Fadd32(3fml)、Fchg、Fchg32(3fml), Ffind、Ffind32(3fml)、MIB(5)、WS_MIB(5)

『BEA Tuxedo アプリケーションの設定』

『BEA Tuxedo アプリケーション実行時の管理』

『C 言語を使用した BEA Tuxedo アプリケーションのプログラミング』

『FML を使用した BEA Tuxedo アプリケーションのプログラミング』

TMFFNAME(5)

形式 FactoryFinder およびサポートする NameManager サービスを実行するサーバ

構文

```
TMFFNAME SRVGRP="identifier" SRVID="number"
     [CLOPT="[-A] [servopts options]
[-- [-F ] [-N | -N -M [-f filename]]]"]
```

機能説明 TMFFNAME は BEA Tuxedo が提供するサーバで、FactoryFinder およびサポートする NameManager サービス (アプリケーションが提供する名前とオブジェクト・リファレンスとのマッピングを管理する) を実行します。

パラメータ

`-A`
: サーバに組み込まれているすべてのサービスを宣言します。

`-F`
: FactoryFinder サービス。

`-N`
: スレーブ NameManager サービス。これがデフォルトです。

`--m`
: マスタ NameManager サービス。

`-f filename`
: FactoryFinder のインポート・ファイルおよびエクスポート・ファイルの場所。

FactoryFinder サービスは CORBA から派生したサービスです。このサービスは、アプリケーション固有の検索基準に対応するアプリケーション・ファクトリの検出機能をクライアント・アプリケーションに提供します。FactoryFinder API の詳細については、『BEA Tuxedo CORBA プログラミング リファレンス』を参照してください。ファクトリの登録および登録解除については、『BEA Tuxedo CORBA サーバ・アプリケーションの開発方法』を参照してください。CLOPT でサービスが指定されていない場合、FactoryFinder サービスが「デフォルト」のサービスになります。

NameManager は、アプリケーションが提供する名前とオブジェクト・リファレンスとのマッピングを管理する BEA Tuxedo- 固有のサービスです。このサービスの用途の 1 つとして、アプリケーション・ファクトリ名とオブジェクト・リファレンス間の対応関係を示すリストを維持することが挙げられます NameManager サービスは、-M オプションを使用してマスタ・ロールとして起動できます。-M オプションを指定しない場合、NameManager はスレーブになります。スレーブの NameManager はマスタから更新データを取得します。1 つのアプリケーションでマスタとして指定できる NameManager は 1 つだけです。

マスタ NameManager は、リモート・ドメインにあるファクトリ・オブジェクトをローカル・ドメインでアクセスできるようにコンフィギュレーションできます。また、ローカル・ドメインにあるファクトリ・オブジェクトをリモート・ドメインからアクセスできるようにコンフィギュレーションすることもできます。これらのコンフィギュレーションはいずれも、FactoryFinder Domains コンフィギュレーション・ファイル、factory_finder.ini で指定します。

factory_finder.ini ファイルの場所は、マスタ NameManager の -f コマンド行オプションで指定します。-f オプションを指定しても factory_finder.ini ファイルが見つからない場合、マスタ NameManager の初期化が失敗します。-f オプションを指定しない場合、ローカル・アプリケーションからはローカルに登録されたファクトリ・オブジェクトにしかアクセスできず、リモート・ドメイン内のアプリケーションからはローカル・ファクトリ・オブジェクトにアクセスできません。

注記　同じサービスを実行する TMFFNAME プロセスを 1 つまたは複数起動できます。信頼性を高めるには、異なるマシンで少なくとも 2 つ以上の NameManager サービスをコンフィギュレーションする必要があります。

相互運用性　TMFFNAME サーバは、BEA Tuxedo 4.0 以降のソフトウェアで動作します。

注意　アプリケーションの UBBCONFIG (TMFFNAME -N) でコンフィギュレーションされている NameManager サービスが 2 つに満たない場合、サーバは起動中に終了し、ユーザ・ログにエラー・メッセージを書き込みます。

アプリケーションの UBBCONFIG ファイルでコンフィギュレーションされていないマスタ NameManager サービスがスレーブ NameManager サービスの開始時に実行されていると、サーバは起動中に終了し、ユーザ・ログにエラー・メッセージを書き込みます。また、マスタがダウンしている場合は、マスタが再起動するまでファクトリを登録および登録解除できません。

TMSYSEVT サーバがアプリケーションの UBBCONFIG ファイルでコンフィギュレーションされておらず、NameManager サービスの開始時に実行されていない場合、サーバは起動中に終了し、ユーザ・ログにエラー・メッセージを書き込みます。

NameManager サービスがアプリケーションの UBBCONFIG ファイルでコンフィギュレーションされていない場合に FactoryFinder サービスが開始されると、サーバは起動中に終了し、ユーザ・ログにエラー・メッセージを書き込みます。

使用例

```
*SERVERS
TMSYSEVT SRVGRP=ADMIN1 SRVID=44 RESTART=Y
          CLOPT="-A"

TMFFNAME SRVGRP=ADMIN1 SRVID=45 RESTART=Y
CLOPT="-A -- -F"

TMFFNAME SRVGRP=ADMIN1 SRVID=46 RESTART=Y
CLOPT="-A -- -N -M -f c:\appdir\import_factories.ini"
TMFFNAME SRVGRP=ADMIN2 SRVID=47 RESTART=Y
CLOPT="-A -- -N"
TMFFNAME SRVGRP=ADMIN3 SRVID=48 RESTART=Y
CLOPT="-A -- -F"
TMFFNAME SRVGRP=ADMIN4 SRVID=49 RESTART=Y
CLOPT="-A -- -F"
```

関連項目

『BEA Tuxedo CORBA プログラミング リファレンス』の factory_finder.ini(5)、TMSYSEVT(5)、UBBCONFIG(5)、userlog(3c) および "TP Framework"

TMIFRSVR(5)

名前 インターフェイス・リポジトリ・サーバ

形式

```
TMIFRSVR SRVGRP="identifier" SRVID="number" RESTART=Y GRACE=0
CLOPT="[servopts options] -- [-f repository_file_name]"
```

機能説明 TMIFRSVR サーバは、インターフェイス・リポジトリにアクセスするために BEA が提供するサーバです。API は、CORBA で定義されるインターフェイス・リポジトリ API のサブセットです。インターフェイス・リポジトリ API については、『BEA Tuxedo CORBA プログラミング リファレンス』を参照してください。

パラメータ

[-f *repository_file_name*]

インターフェイス・リポジトリ・ファイルの名前。このファイルは、あらかじめ idl2ir コマンドを使用して作成しておく必要があります。このパラメータが指定されていない場合、マシン用のアプリケーション・ディレクトリ (APPDIR) に置かれたデフォルトのリポジトリ・ファイル名 repository.ifr が使用されます。リポジトリ・ファイルが読み取れない場合、サーバは起動できません。

使用例

```
*SERVERS

# このサーバはデフォルトのリポジトリ TMIFRSVR を使用する
SRVGRP="IFRGRP" SRVID=1000 RESTART=Y GRACE=0

# このサーバはデフォルト以外のリポジトリ TMIFRSVR を使用する
SRVGRP="IFRGRP" SRVID=1001 RESTART=Y GRACE=0
CLOPT="-- -f /nfs/repository.ifr"
```

関連項目 ir2idl(1)、UBBCONFIG(5)、servopts(5)

TMQFORWARD(5)

名前 TMQFORWARD— メッセージ転送サーバ

形式
```
TMQFORWARD SRVGRP="identifier" SRVID="number" REPLYQ=N CLOPT="
[-A] [servopts options] -- -q queuename[,queuename...]
[-t trantime ] [-i idletime] [-e] [-d] [-n] [-f delay] "
```

機能説明 メッセージ転送サーバは BEA Tuxedo システムが提供するサーバで、tpenqueue() で格納されたメッセージを後処理のために転送します。アプリケーション管理者は、SERVERS セクションでこのサーバをアプリケーション・サーバとして指定することにより、アプリケーション・サーバのメッセージ処理を自動化できます。

位置指定、サーバ・グループ、サーバ識別子、その他の汎用サーバ関連パラメータは、サーバ用に定義されているコンフィギュレーション・ファイル機構を使用して、このサーバに関連付けられます。次に、カスタマイズに使用できる追加コマンド行オプションの一覧を示します。

-q *queuename[,queuename...]*
: このサーバのメッセージの転送先である 1 つまたは複数のキュー / サービスの名前を指定する場合に使用します。キューおよびサービスの名前は、15 文字以内の文字列です。このオプションは必須です。

-t *trantime*
: キューからメッセージを取り出し、そのメッセージをアプリケーション・サーバに転送するトランザクションについて、tpbegin() で使用されるトランザクション・タイムアウト値を指定する場合に使用します。指定されない場合、デフォルトは 60 秒です。

-i *idletime*
: サーバが読み取るキューを排出した後に、サーバがアイドル状態になる時間を指定する場合に使用します。負の値は、ミリ秒単位の時間を表します。たとえば、-i -10 の場合、アイドル時間は 10 ミリ秒となります。

  値が 0 の場合、サーバがキューを連続して読み取ることを示しますが、これを指定した場合は、キューのメッセージが連続していない

と効率が低下する可能性があります。指定されない場合、デフォルトは 30 秒です。

-e

キュー上にメッセージがない状況でサーバを終了させる場合に使用します。このオプションをキューに関連付けられたしきい値コマンドと組み合わせて使用することにより、キューに登録されたメッセージの変化に応じて TMQFORWARD サーバを開始および停止できます。

-d

トランザクションのロールバック後に、サービスを異常終了させ、応答メッセージ (長さがゼロ以外) を持つメッセージをキューから削除する場合に使用します。つまり、サービスが失敗し、かつ長さがゼロ以外の応答メッセージがサーバから受信された場合、元の要求メッセージはキューに返されるのではなく削除されます。

応答メッセージは、異常終了キューがメッセージに関連付けられていて、かつ存在していれば、そのキューに登録されます。キューに設定されている再試行回数の上限に達すると同時にメッセージが削除されるようになっている場合、元の要求メッセージはエラー・キューに移されます。

-n

TPNOTRAN フラグを使ってメッセージを送信する場合に使用します。このフラグを指定すると、リソース・マネージャに関連付けられていないサーバ・グループに転送できるようになります。

-f *delay*

サーバが tpcall を使う代わりにサービスにメッセージを転送するよう指定する場合に使用します。メッセージが送信され、サービスからの応答は期待されません。TMQFORWARD サーバは、サービスからの応答を待つときにブロックすることなく、キューの次のメッセージを続けて処理できます。TMQFORWARD がシステムを要求でいっぱいにしないようにするには、*delay* 値に、処理する要求間の遅延を秒単位で指定します。ゼロを指定すると、遅延は設定されません。

メッセージは、それが読み取られるキューと同じ名前のサービスを提供するサーバに送信されます。キューへのメッセージ登録時に優先順位を指定した場合は、その優先順位がメッセージの優先順位になります。指定していない場合、優先順位はコンフィギュレーション・ファイルで定義されているサービスの優先順位か、またはデフォルト (50) になります。

メッセージは、1 つのトランザクションの中でキューから取り出され、サーバに送信されます。サービスが正常終了すると、トランザクションはコミットされ、メッセージはキューから削除されます。メッセージが応答キューに関連付けられている場合は、サービスからの応答はいずれも、返された tpurcode と共に応答キューに登録されます。. 応答キューが存在しない場合、応答はドロップされます。

元のメッセージをキューに登録する場合、アプリケーションではメッセージに対する応答のサービス品質を指定することができます。応答のサービス品質が指定されていない場合、応答キューに指定されているデフォルトの配信ポリシーが使用されます。デフォルトの配信ポリシーは、メッセージに対する応答がキューに登録されるときに決定される点に注意してください。つまり、元のメッセージがキューに登録されてから応答が登録されるまでの間に、応答キューのデフォルトの配信ポリシーが変更された場合、応答が最後に登録される時点で有効な方針が使用されます。

サービスが異常終了した場合、そのキューに対する再試行制限によって指定されている回数の範囲内でトランザクションがロールバックされ、メッセージがキューに戻されます。メッセージがキューに戻される際、そのメッセージが最初にキューに登録されたときに適用された順位付けルールおよびキューからの取り出しルールは、*delay* 秒の間、一時的に効力を失います。これにより、たとえば、順位付けの低いメッセージが、キューに戻されたメッセージより先に取り出される可能性が出てきます。

-d オプションを指定した場合、サービスが異常終了し、かつサーバから応答メッセージが受信されるとメッセージはキューから削除されます。また、応答メッセージ (および関連する tpurcode) は、異常終了キューがそのメッセージに関連付けられており、かつ存在していればそのキューに登録されます。キューに設定されている再試行の制限に達すると同時にメッセージが削除されるようになっている場合、元の要求メッセージはエラー・キューに移されます。

コンフィギュレーションの条件によっては、TMQFORWARD がキューからメッセージを取り出せないか、メッセージを転送できないことがあり、その場合はサーバを起動できなくなります。こうした条件では、次のことが必要です。

- SRVGRP では、TMSNAME が TMS_QM. に設定されている必要があります。
- OPENINFO に、関連するデバイスおよびキューの名前が設定されていなければなりません。

- SERVER エントリは、MSSQ セットの一部であってはなりません。
- REPLYQ は N に設定されている必要があります。
- -q オプションがコマンド行オプションで指定されている必要があります。
- サーバは、サービスを宣言してはなりません (つまり -s オプションを指定してはなりません)。

アプリケーション・バッファ・タイプの処理

TMQFORWARD は、BEA Tuxedo に用意されている標準バッファ・タイプを処理します。これ以外のアプリケーション・バッファ・タイプが必要な場合は、buildserver(1) をカスタマイズ・タイプ・スイッチと共に使用して、TMQFORWARD のカスタマイズ・バージョンを構築する必要があります。詳細については、『BEA Tuxedo /Q コンポーネント』を参照してください。

呼び出し元によって組み込まれるファイルには、アプリケーション・バッファ・タイプ・スイッチおよびサポートされる必須のルーチンのみを入れてください。buildserver は、サーバ・オブジェクト・ファイル $TUXDIR/lib/TMQFORWARD.o とアプリケーション・タイプ・スイッチのファイル (複数可) を結合し、これに必要な BEA Tuxedo システム・ライブラリをリンクするために使用されます。次の例を使用して、詳細を説明します。

```
buildserver -v -o TMQFORWARD -r TUXEDO/QM -f
${TUXDIR}/lib/TMQFORWARD.o -f apptypsw.o
```

buildserver オプションは次のとおりです。

-v
: buildserver を冗長モードで動作させます。cc コマンドの実行結果が、標準出力へ書き込まれます。

-o *name*
: 出力ロード・モジュールのファイル名を指定します。このオプションで指定された名前は、コンフィギュレーション・ファイルの SERVERS セクション中にも指定しなければなりません。一貫性を保つために、この名前として TMQFORWARD を使用することをお勧めします。このコマンドのアプリケーション固有バージョンは $APPDIR にインストールでき、インストールすると $TUXDIR/bin にあるコマンドの代わりにそれが起動されます。

-r TUXEDO/QM
: このサーバのリソース・マネージャを指定します。値 TUXEDO/QM は、$TUXDIR/udataobj/RM に配置されているリソース・マネージャ・テーブルにあり、BEA Tuxedo システムのキュー・マネージャ用のライブラリを含んでいます。

-f $TUXDIR/lib/TMQFORWARD.o
: TMQFORWARD サービスが入っているオブジェクト・ファイルを指定します。このファイルは、-f オプションの最初の引数として指定してください。

-f *firstfiles*
: buildserver のコンパイル段階やリンク段階で組み込まれる 1 つまたは複数のユーザ・ファイルを指定します。ソース・ファイルは、cc コマンド、または CC 環境変数を通して指定されるコンパイル・コマンドのいずれかを使用してコンパイルされます。これらのファイルは、TMQFORWARD.o オブジェクト・ファイルを組み込んだ後に指定しなければなりません。複数のファイルを指定する場合には、ファイル名を空白類 (スペースまたはタブ) で区切り、全体を引用符で囲みます。このオプションは、何回も指定することができます。

サービスを宣言するための -s を指定してはなりません。

移植性 TMQFORWARD は、サポートされているすべてのサーバ・プラットフォームで BEA Tuxedo システム提供のサーバとしてサポートされます。

相互運用性 TMQFORWARD は相互運用するアプリケーションで実行できますが、BEA Tuxedo リリース 4.2 以降のノードで実行する必要があります。

使用例

```
*GROUPS # Windows の場合、:myqueue を ;myqueue にする
TMQUEUEGRP LMID=lmid GRPNO=1 TMSNAME=TMS_QM
   OPENINFO="TUXEDO/QM:/dev/device:myqueue"
# CLOSEINFO は必要ない

*SERVERS # 推奨値は RESTART=Y GRACE=0
TMQFORWARD SRVGRP="TMQUEUEGRP" SRVID=1001 RESTART=Y GRACE=0
   CLOPT=" -- -qservice1,service2" REPLYQ=N
TMQUEUE SRVGRP="TMQUEUEGRP" SRVID=1000 RESTART=Y GRACE=0
   CLOPT="-s ACCOUNTING:TMQUEUE"
```

関連項目 buildserver(1)、tpdequeue(3c)、tpenqueue(3c)、servopts(5)、TMQUEUE(5)、UBBCONFIG(5)

『BEA Tuxedo アプリケーションの設定』

『C 言語を使用した BEA Tuxedo アプリケーションのプログラミング』

TMQUEUE(5)

名前 TMQUEUE― メッセージ・キュー・マネージャ

形式
```
TMQUEUE
SRVGRP="identifier"
SRVID="number" CLOPT=" [-A][servopts options] -- [-t timeout]"
```

機能説明 メッセージ・キュー・マネージャは BEA Tuxedo システムが提供するサーバで、tpenqueue() および tpdequeue() を呼び出すプログラムの代わりにキューへのメッセージの登録とキューからのメッセージの取り出しを実行します。アプリケーション管理者は、SERVERS セクションでこのサーバをアプリケーション・サーバとして指定することにより、アプリケーションでキューに対するメッセージの登録および取り出しを行うことができます。

位置指定、サーバ・グループ、サーバ識別子、その他の汎用サーバ関連パラメータは、サーバ用に定義されているコンフィギュレーション・ファイル機構を使用して、このサーバに関連付けられます。次に、カスタマイズに使用できる追加コマンド行オプションの一覧を示します。

-t *timeout*
: トランザクション・モード以外でのキュー操作で使用するタイムアウトを指定するために使用します。たとえば、tpenqueue() または tpdequeue() がトランザクション・モード以外の呼び出し元から呼び出されたり、TPNOTRAN フラグを指定して呼び出されたりする場合です。この値は、TPQWAIT オプションが指定された、キューからの取り出し要求にも影響します。この値に基づいて操作がタイムアウトし、エラーが要求者に返されるからです。指定されない場合、デフォルトは 30 秒です。

TMQUEUE サーバはアプリケーションの一部として起動され、アプリケーションから関連するキュー・スペースへのアクセスを容易にします。キュー・スペースはキューの集まりです。

コンフィギュレーションの条件によっては、TMQUEUE がキューにメッセージを登録できないか、キューからメッセージを取り出せないことがあり、その場合 TMQUEUE サーバは起動時に異常終了します。SRVGRP では、TMSNAME が TMS_QM に設定されている必要があり、OPENINFO には関連するデバイスおよびキュー・スペースの名前が設定されている必要があります。

メッセージ要求時のキューの名前

tpenqueue() および tpdequeue() 関数は、第 1 引数としてキュー・スペースの名前をとります。この名前は、TMQUEUE によって宣言されたサービスの名前でなければなりません。デフォルトでは、TMQUEUE はサービス "TMQUEUE" だけを提供します。キュー・スペースを 1 つだけ使用するアプリケーションの場合はこれで十分ですが、複数のキュー・スペースを持つアプリケーションでは、異なるキュー・スペース名を必要とすることがあります。また、アプリケーションによっては、キュー・スペースと同じ名前のより説明的なサービス名を指定したい場合もあります。追加サービス名の宣言は、使用例にもあるように、標準のサーバ・コマンド行オプション -s を使用して行うことができます。また、この後の項で説明するように、カスタム TMQUEUE プログラムを生成するときにサービスをハード・コード化してこれを行うこともできます。

これらの方法 (サーバ・コマンド行オプション、またはカスタマイズされたサーバ) は、キュー・スペースにメッセージの静的ルーティングを行う場合に使用できますが、データ依存型ルーティングを使用して動的なルーティングを行うこともできます。この場合、各 TMQUEUE サーバは同じサービス名を宣言しますが、コンフィギュレーション・ファイルの ROUTING フィールドが使用されて、待機メッセージ内のアプリケーション・データに基づくルーティング基準が指定されます。ルーティング関数は、サービス名とアプリケーションの型付きバッファ・データに基づいて、GROUP を返します。この GROUP を使用して、指定のグループにあるサービスにメッセージが転送されます (なお、キュー・スペースは、OPENINFO 文字列に基づいて GROUP につき 1 つしか存在できません)。

アプリケーション・バッファ・タイプの処理

TMQUEUE は、BEA Tuxedo に用意されている標準バッファ・タイプを処理します。これ以外のアプリケーション・バッファ・タイプが必要な場合は、buildserver(1) を使用して、TMQUEUE のカスタマイズ・バージョンを構築する必要があります。詳細については、『BEA Tuxedo /Q コンポーネント』を参照してください。

buildserver で記述されるカスタマイズは、サーバ用にサービス名をハード・コード化する場合にも使用できます。

呼び出し元によって組み込まれるファイルには、アプリケーション・バッファ・タイプ・スイッチおよびサポートされる必須のルーチンのみを入れてください。buildserver は、サーバ・オブジェクト・ファイル

$TUXDIR/lib/TMQUEUE.o とアプリケーション・タイプ・スイッチのファイル (複数可) を結合し、これに必要な BEA Tuxedo システム・ライブラリをリンクするために使用されます。次の例を使用して、詳細を説明します。

```
buildserver -v -o TMQUEUE -s qspacename:TMQUEUE -r TUXEDO/QM \
-f ${TUXDIR}/lib/TMQUEUE.o -f apptypsw.o
```

buildserver オプションは次のとおりです。

-v
: buildserver を冗長モードで動作させます。cc コマンドの実行結果が、標準出力へ書き込まれます。

-o *name*
: 出力ロード・モジュールのファイル名を指定します。このオプションで指定された名前は、コンフィギュレーション・ファイルの SERVERS セクション中にも指定しなければなりません。一貫性を保つために、この名前として TMQUEUE を使用することをお勧めします。

-s *qspacename,qspacename* :TMQUEUE
: サーバの起動時に宣言できるサービスの名前を指定します (servopts(5) を参照)。このサーバでは、サービス名は要求の送信先となるキュー・スペースの名前のエイリアスとして使用されます。カンマとカンマの間に空白を入れてはいけません。関数名 TMQUEUE の前にはコロンを付けます。-s オプションは何回使用してもかまいません。

-r TUXEDO/QM
: このサーバのリソース・マネージャを指定します。値 TUXEDO/QM は、$TUXDIR/udataobj/RM に配置されているリソース・マネージャ・テーブルにあり、BEA Tuxedo システムのキュー・マネージャ用のライブラリを含んでいます。

-f $TUXDIR/lib/TMQUEUE.o
: TMQUEUE サービスが入っているオブジェクト・ファイルを指定します。このファイルは、-f オプションの最初の引数として指定してください。

-f *firstfiles*
: buildserver のコンパイル段階やリンク段階で組み込まれる 1 つまたは複数のユーザ・ファイルを指定します。ソース・ファイルは、cc コ

マンド、または CC 環境変数を通して指定されるコンパイル・コマンドのいずれかを使用してコンパイルされます。これらのファイルは、TMQUEUE.o オブジェクト・ファイルを組み込んだ後に指定しなければなりません。複数のファイルを指定する場合には、ファイル名を空白類 (スペースまたはタブ) で区切り、全体を引用符で囲みます。このオプションは、何回も指定することができます。

移植性 TMQUEUE は、サポートされているすべてのサーバ・プラットフォームで BEA Tuxedo システム提供のサーバとしてサポートされます。

相互運用性 TMQUEUE は相互運用するアプリケーションで実行できますが、BEA Tuxedo リリース 4.2 以降のノードで実行する必要があります。

使用例

```
*GROUPS
# Windows の場合、:myqueue を ;myqueue にする
TMQUEUEGRP1 GRPNO=1 TMSNAME=TMS_QM
   OPENINFO="TUXEDO/QM:/dev/device1:myqueue"
# Windows の場合、:myqueue を ;myqueue にする
TMQUEUEGRP2 GRPNO=2 TMSNAME=TMS_QM
   OPENINFO="TUXEDO/QM:/dev/device2:myqueue"

*SERVERS
# この例では、キュー・スペース名 myqueue に ACCOUNTING というエイリアスが付けられている
TMQUEUE SRVGRP="TMQUEUEGRP1" SRVID=1000 RESTART=Y GRACE=0
    CLOPT="-s ACCOUNTING:TMQUEUE"
TMQUEUE SRVGRP="TMQUEUEGRP2" SRVID=1000 RESTART=Y GRACE=0
    CLOPT="-s ACCOUNTING:TMQUEUE"
TMQFORWARD SRVGRP="TMQUEUEGRP1" SRVID=1001 RESTART=Y GRACE=0 REPLYQ=N
    CLOPT=" -- -qservice1"
TMQFORWARD SRVGRP="TMQUEUEGRP2" SRVID=1001 RESTART=Y GRACE=0 REPLYQ=N
    CLOPT=" -- -qservice1"
*SERVICES
ACCOUNTING ROUTING="MYROUTING"
*ROUTING
MYROUTING FIELD=ACCOUNT BUFTYPE="FML"
RANGES="MIN - 60000:TMQUEUEGRP1,60001-MAX:TMQUEUEGRP2"
```

この例では、2 つのキュー・スペースを使用できます。どちらの TMQUEUE サーバも同じサービスを提供し、ルーティングはアプリケーションの型付きバッファの ACCOUNT フィールドを介して行われます。

関連項目 buildserver(1)、tpdequeue(3c)、tpenqueue(3c)、servopts(5)、TMQFORWARD(5)、UBBCONFIG(5)

『BEA Tuxedo アプリケーションの設定』

『BEA Tuxedo アプリケーション実行時の管理』

『C 言語を使用した BEA Tuxedo アプリケーションのプログラミング』

TMSYSEVT(5)

名前 TMSYSEVT— システム・イベント通知プロセス

形式

```
TMSYSEVT SRVGRP="identifier" SRVID="number"
  [CLOPT="[-A] [servopts options]
  [-- [-S] [-p poll-seconds] [-f control-file]]"]
```

機能説明 TMSYSEVT は BEA Tuxedo システムが提供するサーバで、システム・エラーや潜在的なエラー状態に関するイベント通知を処理します。イベント・レポートはフィルタを通され、1 つまたは複数の通知アクションを開始できます。

フィルタや通知に関するルールは、制御ファイル *control-file* に格納されます。デフォルトの名前は ${APPDIR}/tmsysevt.dat です。control-file の構文は EVENT_MIB(5) に関する追加情報 で定義します。特に、EVENT_MIB のクラス属性を設定することで、通知ルールの範囲内でサブスクリプションをアクティブ化できます。

1 つまたは複数の二次的な TMSYSEVT プロセスを起動して、可用性を高めることができます。追加サーバは、「セカンダリ・サーバ」であることを示すコマンド行オプション -S を指定して起動する必要があります。

EVENT_MIB(5) に関する追加情報 のコンフィギュレーションが更新されるときには、プライマリ TMSYSEVT サーバがその制御ファイルに書き込みを行います。セカンダリ・サーバは、プライマリ・サーバの制御ファイルが更新されているかどうかをポーリングを通じてチェックし、必要であれば各自の (ローカルの) 制御ファイルを更新します。ポーリングの間隔は -p オプションで指定できます。デフォルトは 30 秒です。

相互運用性 TMSYSEVT は、BEA Tuxedo リリース 6.0 以降のマシンで実行する必要があります。

注意事項 プライマリ TMSYSEVT サーバを別のマシンに移行するには、システム管理者は現在の制御ファイルのコピーを提供する必要があります。セカンダリ TMSYSEVT サーバは、最新のコピーを自動的に維持します。

TMSYSEVT は、システム・イベントを定義した FML32 フィールド・テーブルへのアクセス手段が必要になります。FLDTBLDIR32 には $TUXDIR/udataobj が、FIELDTBLS32 には evt_mib がそれぞれ含まれている必要があります。これらの環境変数は、マシンまたはサーバの環境設定ファイルで設定できます。

使用例

```
*SERVERS
TMSYSEVT SRVGRP=ADMIN1 SRVID=100 RESTART=Y GRACE=900 MAXGEN=5
  CLOPT="-A --"
TMSYSEVT SRVGRP=ADMIN2 SRVID=100 RESTART=Y GRACE=900 MAXGEN=5
  CLOPT="-A -- -S -p 90"
```

関連項目 tpsubscribe(3c), EVENTS(5)、EVENT_MIB(5) に関する追加情報、TMUSREVT(5)

tmtrace(5)

名前 tmtrace— 実行時のトレース機能

機能説明 実行時トレース機能を使用すると、アプリケーション管理者および開発者は、BEA Tuxedo アプリケーションの実行をトレースできます。

実行時トレースは、アプリケーション実行時の注意すべき条件または遷移をマークする、*trace point* の概念に基づいています。トレース・ポイントの例としては、tpcall などの ATMI 関数へのエントリ、BEA Tuxedo メッセージの到着、トランザクションの開始などがあります。

トレース・ポイントに到達すると、次のことが発生します。まず、フィルタ *filter* が適用され、トレース・ポイントを処理すべきかどうかを決定します。トレース・ポイントを処理すべき場合、*trace record* が *receiver* に発行されます。receiver はファイルか (将来は) バッファです。最後に、プロセスの中止などの *action* が起動されます。レシーバへの発行およびトリガはどちらもオプションであり、トレース・ポイントがフィルタを通らない場合には発生しません。

フィルタ、レシーバ、およびトリガは、*trace specification* で指定します。構文について以下に記述します。トレース指定は、TMTRACE 環境変数から初期化されます。実行中のプロセスのトレース指定は、トリガー・アクションによってか、tmadmin(1) の changetrace コマンドを使用することで変更できます。

トレース・ポイントは、以下に列挙するような *trace categories* に分類されます。各トレース・ポイントは 1 つのカテゴリに属します。フィルタは、処理するトレース・カテゴリを記述し、フィルタを通らないトレース・ポイントには最小限の処理が行われます。

また、実行時トレーシングは、クライアントがサーバに送信したメッセージ (またはそのサーバから他のサーバへ) を *dye* 設定する機能を提供します。プロセスがメッセージをダイ設定するように選択した場合、発信元プロセスによってダイ設定が自動的に、発信元プロセスから直接または間接的にメッセージを受信するすべてのプロセスに渡されます。プロセスがダイ設定され

たメッセージを受け取ると、atmi トレース・カテゴリが自動的にオンになり、ユーザ・ログへのトレース・レコードの発行が開始されます (発行が行われていなかった場合)。

ダイ設定は、トレース指定の dye トリガと undye トリガによって明示的にオンまたはオフにできます。また、ダイ設定は、ダイ設定されたメッセージが受信されたときに暗黙的にオンになり、tpreturn() および tpforward() によって暗黙的にオフにできます。ダイ設定が暗黙的にオフされた場合、ダイ設定がオンであったときに有効であったトレース指定が復元されます。

トレース・カテゴリ

トレース・カテゴリは以下のとおりです。

atmi
: ATMI インターフェイスおよび TX インターフェイスへの明示的なアプリケーション呼び出し (つまり tp 関数や tx_ 関数に対する呼び出し) およびアプリケーション・サービス起動のためのトレース・ポイント。いくつか例外があります。最初の呼び出し tpinit() で ATMI 呼び出しが処理される際の tpinit に対する暗黙的な呼び出し、およびエラーが発生して tpreturn が呼び出される場合には、暗黙的な呼び出しは TX インターフェイスが ATMI インターフェイスを直接呼び出すこのカテゴリーに出力されます。

iatmi
: ATMI および TX インターフェイスの暗黙的な呼び出しのトレース・ポイント。これらのトレース・ポイントは、アプリケーションの要求の処理中に呼び出される内部呼び出し、および管理を目的として呼び出される内部呼び出しのすべてを示します。この値をセットすることは、atmi レベル、すなわち ATMI または TX インターフェイスのすべての呼び出し (明示的な呼び出しと暗黙的な呼び出しの両方) がトレースされることを意味します。

xa
: XA インターフェイス (トランザクション・マネージャとリソース・マネージャ間のインターフェイス) のすべての呼び出しのトレース・ポイント。

trace
: メッセージのダイ設定を含む、トレース機能自体に関連するトレース・ポイント。

トレース指定

トレース指定は、*filter-spec*: *receiver-spec* [: *trigger-spec*] という構文で指定する文字列です。*filter-spec* は、検査または無視するトレース・カテゴリを記述します。*receiver-spec* は、トレース・レコードのレシーバです。オプションの *trigger-spec* は、実行するアクションを記述します。

ヌル文字列も有効なトレース指定です。ヌル文字は、他の指定がない場合のすべての BEA Tuxedo プロセスのデフォルトです。

文字列 `on` と `off` も指定できます。`on` は `atmi:ulog:dye` のエイリアスで、`off` は `: :undye` と等価です。

フィルタ指定

トレース指定の最初の要素であるフィルタ指定の構文は次のとおりです。

[{ + | - }] [*category*] . . .

ここで *category* は、上記にリストしたカテゴリの 1 つです。*category* の位置に記号 * を使用すると、すべてのカテゴリを表すことができます。接頭辞 + または - は、後続のカテゴリを現在有効なカテゴリのセットに追加またはそのセットから削除することを示します。+ または - の次にカテゴリがない場合、現在有効なカテゴリは変更されません。

フィルタが空の場合、カテゴリが 1 つも選択されず、トレースが使用不可になります。

トレース・ポイントが発生すると、そのカテゴリがフィルタ指定と比較されます。カテゴリがフィルタ指定に含まれている場合、トレース・ポイントは、レシーバおよびトリガ指定に従ってさらに処理されます。カテゴリが含まれていない場合、トレース・ポイントの処理はこれ以上発生しません。

レシーバ指定

レシーバは、トレース・レコードの送信先となるエンティティです。各トレース・レコードには、最大 1 つのレシーバがあります。

トレース指定の 2 番目の要素であるレシーバ指定の構文は次のとおりです。

[/ *regular-expression* /] *receiver*

ここでは、オプションの正規表現を使用して、フィルタを通過するトレース・ポイントの一部を選択できます。正規表現は、トレース・レコードと比較されます。空のレシーバ指定も有効であり、この場合、トレース・レコードは発行されません。

現在のところ、*receiver* の有効値は次の 1 つだけです。

`ulog`
　トレース・レコードをユーザ・ログへ発行します。

トリガ指定　トリガは、トレース・レコードの発行後に実行されるオプションのアクションです。フィルタを通過した各トレース・レコードに対して、最大 1 つのアクションが実行されます。

トレース指定の 3 番目の要素であるトリガ指定の構文は次のとおりです。

`[/ regular-expression /] action`

ここでは、オプションの正規表現を使用して、フィルタを通過するトレース・ポイントの一部に対してだけトリガが実行されるよう設定できます。正規表現は、トレース・レコードと比較されます。

有効なアクションは次のとおりです。

`abort`
　abort() を呼び出してプロセスを終了します。

ulog(*`message`*)
　ユーザ・ログに *`message`* を書き込みます。

system(*`command`*)
　`system`(3) を使用して *`command`* を実行します (これは Windows クライアントではサポートされません)。%A は、トレース・レコードの値に展開されます。

trace(*`trace-spec`*)
　トレース指定を標準の *`trace-spec`* にリセットします。

`dye`
　メッセージのダイ設定をオンにします。

`undye`
　メッセージのダイ設定をオフにします。

sleep(*`seconds`*)
　指定の秒数だけスリープ状態にします (これは Windows クライアントではサポートされません)。

トレース・レコード　トレース・レコードは、次の形式の文字列です。

`cc``:`*`data`*

ここで *cc* はトレース・カテゴリの最初の 2 文字で、*data* はトレース・ポイントに関する追加情報です。

トレース・レコードがユーザ・ログに表示されるとき、その行は次のようになります。

hhmmss.system-name!*process-name.pid*: TRACE:*cc*:*data*

注意事項 MAC プラットフォームで動作するワークステーション・クライアントのレシーバやトリガに対しては、マッチ・パターンを指定することはできません。

`tmadmin changetrace` コマンドは、/WS クライアントに対するトレース・レベルを変更するために使用できません。

使用例 クライアントをトレースし、アプリケーション・サーバがそのクライアントの代わりに行ったすべての ATMI 呼び出しをトレースするには、クライアントの環境に `TMTRACE=on` を設定してエクスポートします。この指定により、クライアント内のすべての明示的な ATMI トレース・ポイントがログに記録され、メッセージのダイ設定がオンになります。こクライアントの代わりにサービスを実行するアプリケーション・サーバ・プロセスは、すべての明示的な ATMI トレース・ポイントを自動的にログに記録します。

すべての (明示的および暗黙的な) クライアント・トレース・ポイントを確認するには、次のように設定してエクスポートします。

```
TMTRACE="*:ulog:dye:"
```

前述の例で、クライアントからのサービス要求はトレースし、クライアントからの出力のトレースを `tpcall` 要求についての最低限の情報に制限するには、次のように設定しエクスポートします。

```
TMTRACE=atmi:/tpacall/ulog:dye
```

この指定により、クライアントで行われるすべての `tpacall` 呼び出しがログに記録され、メッセージのダイ設定がオンになります。クライアントの代わりにサービスを実行するアプリケーション・サーバ・プロセスは、すべての ATMI トレース・ポイントを自動的にログに記録します。`tpacall()` トレース・レコードに含まれるクライアントの識別子は、クライアントの代わりに呼び出されたサービス・ルーチンに渡される `TPSVCINFO` パラメータの値と相互に関連させることができます。

アプリケーション・サーバによって実行されたすべてのサービス要求の呼び出しをトレースするには、次のように設定します。

```
TMTRACE=atmi:/tpservice/ulog
```

これは、参加しているすべてのマシン上のサーバ *ENVFILE* に設定します。

メッセージのダイ設定をオンにして、アプリケーションを通してすべてのトレース・カテゴリの実行時トレースを有効にするには、次のように設定してエクスポートします。

```
TMTRACE=*:ulog:dye
```

これは、すべてのクライアント環境および参加しているすべてのマシン上のサーバ *ENVFILE* に設定します。この設定では、BBL と DBBL を含むすべてのプロセスがトレース・レコードを発行するので、管理できない量の出力が生成される可能性があります。

グループ GROUP1 内の実行中のすべてのサーバで、起動後に ATMI のトレースをオンにするには、次のように tmadmin の changetrace コマンドを呼び出します。

```
changetrace -g GROUP1 on
```

changetrace は現在存在しているプロセスに対してのみ影響します。つまり、グループ GROUP1 内で起動していないサーバのトレース・コンフィギュレーションは変更されません (サーバのデフォルトのトレース・コンフィギュレーションを設定するには、サーバの *ENVFILE* に TMTRACE を設定します)。

現在実行中のすべてのアプリケーション・プロセスでトレーシングをオフにするには、次のように changetrace を使用します。

```
changetrace -m all off
```

グループ GROUP1 内で識別子が 1 の実行中のサーバ・プロセスを tpreturn の実行時に中止するには、tmadmin に対して次のように指定します。

```
changetrace -i 1 -g GROUP1 "atmi::/tpreturn/abort"
```

関連項目 tmadmin(1)、userlog(3c)

TMUSREVT(5)

名前 TMUSREVT— ユーザ・イベント通知プロセス

形式

```
TMUSREVT SRVGRP="identifier" SRVID="number"
  [CLOPT="[-A] [servopts options]
  [-- [-S] [-p poll-seconds] [-f control-file]]"]
```

機能説明 TMUSREVT は BEA Tuxedo システムが提供するサーバで、tppost(3c) からのイベント・レポート・メッセージ・バッファを処理し、それらにフィルタを適用して転送するイベント・ブローカとして動作します。

フィルタや通知に関するルールは、制御ファイル *control-file* に格納されます。デフォルトの名前は ${APPDIR}/tmusrevt.dat です。control-file の構文は EVENT_MIB(5) に関する追加情報 で定義します。特に、EVENT_MIB のクラス属性を設定することで、通知ルールの範囲内でサブスクリプションをアクティブ化できます。

1 つまたは複数の二次的な TMUSREVT プロセスを起動して、可用性を高めることができます。追加サーバは、「セカンダリ・サーバ」であることを示すコマンド行オプション -S を指定して起動する必要があります。

EVENT_MIB(5) に関する追加情報 のコンフィギュレーションが更新されるときには、プライマリ TMUSREVT サーバがその制御ファイルに書き込みを行います。セカンダリ・サーバは、プライマリ・サーバの制御ファイルが更新されているかどうかをポーリングを通じてチェックし、必要であれば各自の (ローカルの) 制御ファイルを更新します。ポーリングの間隔は -p オプションで指定できます。デフォルトは 30 秒です。

相互運用性 TMUSREVT は、BEA Tuxedo リリース 6.0 以降のマシンで実行する必要があります。

注意事項 プライマリ TMUSREVT サーバを別のマシンに移行するには、システム管理者は現在の制御ファイルのコピーを提供する必要があります。セカンダリ TMUSREVT サーバは、最新のコピーを自動的に維持します。

tppost() をトランザクション・モードで呼び出す場合は、すべての TMUSREVT サーバ・グループがトランザクション機能 (TMS プロセス) を備えている必要があります。

TMUSREVT サーバの環境変数は、メッセージにフィルタを適用したりフォーマットする際に必要となる FML フィールド・テーブルや VIEW ファイルを使用できるように設定しておく必要があります。これらの環境変数は、マシンまたはサーバの環境設定ファイルで設定できます。

使用例

```
*SERVERS
TMUSREVT SRVGRP=ADMIN1 SRVID=100 RESTART=Y MAXGEN=5 GRACE=3600
 CLOPT="-A --"
TMUSREVT SRVGRP=ADMIN2 SRVID=100 RESTART=Y MAXGEN=5 GRACE=3600
 CLOPT="-A -- -S -p 120"
```

関連項目

tppost(3c)、tpsubscribe(3c)、EVENTS(5)、EVENT_MIB(5) に関する追加情報、TMSYSEVT(5)

tperrno(5)

名前 tperrno—BEA Tuxedo システム・エラー・コード

形式 #include <atmi.h>

機能説明 エラー条件のシンボル名によって表される数値は、BEA Tuxedo システム・ライブラリ・ルーチンの実行時に発生するエラー用の tperrno に割り当てられます。

tperrno は、int 型の変更可能な *lvalue* の拡張です。lvalue の値は、複数の BEA Tuxedo システム・ライブラリ・ルーチンによって正のエラー番号に設定されます。tperrno はオブジェクトの識別子である必要はなく、*lvalue* 関数呼び出しの結果、変更可能な lvalue に拡張される場合があります。tperrno がマクロであるかまたは外部リンクで宣言される識別子であるかは特定されていません。実際のオブジェクトをアクセスするための tperrno マクロの定義が抑止されている場合、または、あるプログラムが名前 tperrno を使用して識別子を定義している場合、動作は不確定です。

BEA Tuxedo システム・ライブラリ・ルーチンのリファレンス・ページには、各ルーチンのエラー条件とそのコンテキストにおけるエラーの意味が掲載されています。掲載されているエラーの順番は重要ではなく、優先順位を示すものでもありません。tperrno の値は、エラーが指摘された後にのみ検査します。すなわち、構成要素の戻り値がエラーを示していて、構成要素の定義で tperrno のエラー時の設定が指定されている場合です。tperrno の値を検査するアプリケーションは、ヘッダ・ファイル <atmi.h> をインクルードしなければなりません。

以下に、各エラーの一般的な意味を示します。

TPEABORT
: イニシエータまたは 1 つ以上のパーティシパントによって実行される処理がコミットできなかったために、トランザクションがコミットできませんでした。

TPEBADDESC
: 呼び出し記述子が無効であるか、会話型サービスを起動したときに使用した記述子ではありません。

TPEBLOCK
: ブロッキング状態のため、TPNOBLOCK が指定されました。

TPEDIAGNOSTIC
: 指定されたキューへのメッセージの登録が異常終了しました。異常終了の原因は、ctl を介して返される診断値によって判別できます。

TPEEVENT
: イベントが発生しました。イベントのタイプは *revent* で返されます。

TPEGOTSIG
: シグナルを受け取りましたが、TPSIGRSTRT が指定されていません。

TPEHAZARD
: ある種の障害のため、トランザクションの一部としてなされた作業がヒューリスティックに完了している可能性があります。

TPEHEURISTIC
: ヒューリスティックな決定により、トランザクションの代わりに行われた処理は部分的に完了し、部分的にアボートされました。

TPEINVAL
: 無効な引数が検出されました。

TPEITYPE
: 入力バッファのタイプおよびサブタイプは、サービスが扱うタイプおよびサブタイプの 1 つではありません。

TPELIMIT
: 未終了の要求数または接続数が最大数に達したために、呼び出し側の要求が送信されませんでした。

TPEMATCH
: *svcname* は、既にこのサーバについて宣言されていますが、それは、*func* 以外の関数で行われました。

TPEMIB
: 管理要求が失敗しました。*outbuf* が更新され、MIB(5) および TM_MIB(5) で説明するエラーの原因を示す FML32 のフィールドが設定され、呼び出し側に返されました。

TPENOENT
svc が存在していないか、または正しいサービス型でないため、svc に送信できませんでした。

TPEOS
オペレーティング・システムのエラーが発生しました。

TPEOTYPE
応答のタイプおよびサブタイプは、呼び出し側に認識されていません。

TPEPERM
クライアントはアプリケーションに参加できません。クライアントがアプリケーションへの参加を許可されていないか、または正しいアプリケーション・パスワードが提供されていないためです。

TPEPROTO
ライブラリ・ルーチンが不正なコンテキストで呼び出されました。

TPERELEASE
TPACK が指定され、ターゲットは承認プロトコルをサポートしない旧リリースの BEA Tuxedo システムのクライアントです。

TPERMERR
リソース・マネージャがオープンまたはクローズに失敗しました。

TPESVCERR
サービス・ルーチンが、tpreturn() あるいは tpforward() でエラーを検出しました (たとえば、誤った引数が渡された場合など)。

TPESVCFAIL
呼び出し側の応答を送信するサービス・ルーチンが、TPFAIL で tpreturn() を呼び出しました。これは、アプリケーション・レベルの障害です。

TPESYSTEM
BEA Tuxedo システム・エラーが発生しました。

TPETIME
このエラー・コードは、タイムアウトが発生したか、現在のトランザクションが既に「ロールバックのみ」としてマークされているにもかかわらずトランザクション ATMI 関数が試行されたことを示します。

呼び出し側がトランザクション・モードの場合、トランザクションに「ロールバックのみ」のマークが付けられているか、またはトランザクション・タイムアウトが発生しました。このトランザクションは、「アボートのみ」とマークされます。呼び出し側がトランザクション・モードでない場合、ブロッキング・タイムアウトが発生します。ブロッキング・タイムアウトは、TPNOBLOCK または TPNOTIME が指定されている場合は発生しません。いずれの場合も、*odata、その内容、*olen はどれも変更されません。

トランザクション・タイムアウトが発生した場合、トランザクションがアボートされない限り、新しい要求の送信や未処理の応答の受信はできず、TPETIME が発生します。ただし、例外が 1 つあります。その例外とは、ブロックされず、応答を期待せず、かつ呼び出し側のトランザクションの代わりに送信されない (つまり、TPNOTRAN、TPNOBLOCK および TPNOREPLY を設定して tpacall() を呼び出した場合の) 要求です。

サービスがトランザクションの内部で失敗した場合、そのトランザクションは TX_ROLLBACK_ONLY 状態に置かれます。ほとんどの場合、この状態はタイムアウトと同様に取り扱われます。このトランザクションに対する後続のすべての ATMI 呼び出し (前述の環境で発行された呼び出しを除く) は失敗し、TPETIME が発生します。

TPETRAN
: 呼び出し側がトランザクション・モードになりません。

使用法 ルーチンには、エラーの戻り値がないものもあります。tperrno をゼロに設定するルーチンはないため、アプリケーションは、tperrno をゼロに設定し、ルーチンを呼び出してから、エラーが発生したかを調べるために再度 tperrno をチェックできます。

関連項目 個々の BEA Tuxedo ライブラリ・ルーチンの ERRORS の項を参照してください。

tpurcode(5)

名前 tpurcode—アプリケーションが指定する戻りコードのための BEA Tuxedo システムのグローバル変数

形式

```
#include <atmi.h>
```

機能説明 tpurcode は、atmi.h で定義されるグローバル変数です。その値は、tpreturn() の rcode 引数の値として使用されているものと同じ長さの整数です。tpurcode はアプリケーションで使用され、アプリケーション・サービスを呼び出すプロセスに追加的な情報を返す場合があります。詳細については、tpreturn() を参照してください。

アプリケーションが tpurcode の値に意味を割り当てます。

使用例 以下に、tpurcode の使用例を示します。

アプリケーション・サービスで rcode により値 myval を返す場合、次のようになります。

```
.
.
.
tpreturn(TPSUCCESS, myval, rqst->data, 0L, 0);
.
.
.
```

モジュールのコードは、次のようになります。

```
.
.
.
ret = tpcall("TOUPPER", (char *)sendbuf, 0, (char **)&rcvbuf, \
&rcvlen, (long)0);
.
.
.
(void) fprintf(stdout, "Returned string is: %s\n", rcvbuf);
(void) fprintf(stdout, "Returned tpurcode is:%d\n", tpurcode);
```

サンプル・クライアント simpcl を "My String," という値で呼び出した場合、次のように出力されます。

```
%simpcl "My String"
Returned string is: MY STRING
Returned tpurcode is: myval
```

myval の意味はアプリケーションで定義する必要があります。

関連項目 tpreturn(3c)

tuxenv(5)

名前 tuxenv—BEA Tuxedo システムでの環境変数のリスト

機能説明 アプリケーションのクライアントとサーバをコンパイルし、BEA Tuxedo システムを実行するには、正しい環境変数の設定とエクスポートが重要です。ここでは、最も使用頻度の高い変数について説明します。

環境変数は、以下のセクションにグループ分けされています。

- オペレーティング・システム変数
- キー BEA Tuxedo システム変数
- フィールド・テーブル・ファイルおよび VIEW ファイル用の変数
- ファイル・システムおよび TLOG 変数
- ワークステーション変数
- BEA Tuxedo /Q 変数
- COBOL 変数
- DEBUG 変数
- その他の変数

オペレーティング・システム変数

CC
: buildserver およびその他の BEA Tuxedo コマンドで使われる標準 C コンパイラ。

CFLAGS
: C コンパイラで使用するフラグ。

EDITOR
: BEA Tuxedo によって呼び出されるエディタの指定。

LANG
: 言語の設定をするロケールの指定。nl_types(5) を参照してください。

LOGNAME
: エラー・メッセージで使うユーザ名。

LD_LIBRARY_PATH
: 実行時共用ライブラリのパス名に設定する必要があります。

NLSPATH
: メッセージ・カタログのパス名。指定されない場合はデフォルトのパス名が使用されます。See nlpaths(5).

PAGER
: qmadmin(1)、tmadmin(1) でページング出力のために使うページング・コマンド。この指定により、システムのデフォルト (UNIX オペレーティング・システムの pg(1)) は無効になます。

PATH
: 実行可能ファイルを検索するためのパス名。

SHELL
: BEA Tuxedo によって呼び出されるシェル・プログラム。

TERM
: 端末を使用する場合、その端末のタイプ。

TMPDIR
: 一時ファイルが書き込まれるディレクトリのパス名。一時ファイルは、tmpnam( ) 関数 (BEA Tuxedo MIB およびその他の BEA Tuxedo コードで呼び出される) で指定される、オペレーティング・システム固有の場所に書き込むこともできます。tmpnam( ) の呼び出しが行われると、BEA Tuxedo システムは TMPDIR 変数を無視します。

  BEA Tuxedo 6.5 以前のリリースでは、サービス・キューがいっぱいになってメッセージ・ファイルを保持できなくなると、クライアントからのメッセージ・ファイルをサービス・キューに転送する BEA Tuxedo コードは、tmpnam( ) 関数で指定される一時的な場所にメッセージ・ファイルを書き込み、この一時的な場所のパス名をサービス・キューに配置します。BEA Tuxedo 7.1 以降のリリースでも、このコードは旧リリースと同様に機能します。ただし、一時的な場所は TMPDIR によって指定されるディレクトリのパス名になり (この変数が設定されている場合)、TMPDIR が設定されていない場合はオペレー

ティング・システムによって指定されるディレクトリのパス名になります。

TZ
: ANSI C mktime 関数が存在しないシステムでは、BEA Tuxedo gp_mktime(3c) 関数を使用するために TZ を設定する必要があります。

これらの変数についての詳細は、UNIX システムのリファレンス・ページ environ(5) を参照してください。

キー BEA Tuxedo システム変数

通常、以下の環境変数を設定およびエクスポートします。

APPDIR
: アプリケーション・ファイルの基本ディレクトリの絶対パス名。

APP_PW
: セキュリティが稼動している場合にアプリケーション・パスワードの入力を求めるシステム・クライアント用のパスワードを指定します。変数にパスワードを設定すると、そのパスワードは手動による入力ではなくスクリプトから提供されます。

ENVFILE
: tmloadcf(1) で使用される変数。通常、他の BEA Tuxedo システム環境変数 (自動設定される) を含みます。

TLOGDEVICE
: トランザクション・ログ用のパス名。これは、アプリケーションのコンフィグレーション・ファイルで指定されている TLOGDEVICE と同じでなければなりません。

TUXCONFIG
: tmloadcf(1) でロードされるバイナリ・コンフィギュレーション・ファイルのパス名。

TUXDIR
: BEA Tuxedo ソフトウェアがインストールされている基本ディレクトリ。

ULOGPFX
: 中央イベント・ログのファイル名に付ける接頭辞。デフォルトは ULOG です。

TPMBENC

BEA Tuxedo 8.1 以降が実行されているアプリケーション・サーバまたはクライアントが割り当て済み型付きバッファ MBSTRING に追加するコード・セット符号化名。アプリケーション・サーバまたはクライアント・プロセスが MBSTRING バッファを割り当てて送信すると、TPMBENC に定義されているコード・セット符号化名がバッファの属性として自動的に付加され、バッファ・データと一緒に目的のプロセスに送信されます。

アプリケーション・サーバまたはクライアント・プロセスが MBSTRING バッファを受信し、TPMBACONV という別の環境変数が設定されている場合、TPMBENC に定義されているコード・セット符号化名が受信バッファ内のコード・セット符号化名と比較されます。符号化名が同じでない場合、MBSTRING バッファのデータは TPMBENC に定義されている符号化に変換されてからサーバまたはクライアント・プロセスに渡されます。

TPMBENC のデフォルト値はありません。MBSTRING 型付きバッファを使用するアプリケーション・サーバまたはクライアントでは、TPMBENC を定義する必要があります。

注記　TPMBENC は、FML32 型付きバッファの FLD_MBSTRING フィールドと同じように使用されます。

TPMBACONV

BEA Tuxedo 8.1 以降が実行されているアプリケーション・サーバまたはクライアントが、受信した MBSTRING バッファ内のデータを TPMBENC に定義されている符号化に自動変換するかどうかを指定します。デフォルトでは、自動変換は無効です。つまり、受信した MBSTRING バッファ内のデータは符号化変換されず、そのままの状態で目的のサーバまたはクライアント・プロセスに渡されます。― 自動変換を有効にするには、TPMBACONV をヌル以外の任意の値、たとえば Y (yes) などに設定します。

注記　TPMBACONV は、FML32 型付きバッファの FLD_MBSTRING フィールドと同じように使用されます。

URLENTITYCACHING

BEA Tuxedo 8.1 以降のソフトウェアが実行されているアプリケーション・サーバまたはワークステーションが文書型定義 (DTD)、XML スキーマ、およびエンティティ・ファイルをキャッシュするかどうかを

指定します。特に、アプリケーション・サーバまたはワークステーションで実行されている Apache Xerces-C++ パーサが、検証が必要なときに DTD および XML スキーマ・ファイルをキャッシュするかどうか、または DTD で指定された外部エンティティ・ファイルをキャッシュするかどうかを指定します。デフォルトでは、キャッシュは無効 (Y) です。URLENTITYCACHING を N (no) に設定するとキャッシュが有効になります。

URLENTITYCACHEDIR

URLENTITYCACHING=Y (yes) か、または設定されていない場合にのみ適用されます。詳細については、URLENTITYCACHING を参照してください。

BEA Tuxedo 8.1 以降のソフトウェアが実行されているアプリケーション・サーバまたはワークステーションが DTD、XML スキーマ、およびエンティティ・ファイルをキャッシュするかどうかを指定します。特に、アプリケーション・サーバまたはワークステーションで実行されている Apache Xerces-C++ パーサが DTD、XML スキーマ、およびエンティティ・ファイルをキャッシュするかどうかを指定します。URLENTITYCACHEDIR 変数には、キャッシュするファイルの絶対パス名を指定します。URLENTITYCACHEDIR を指定しない場合、デフォルト・ディレクトリは URLEntityCachedir になります。このディレクトリは、アプリケーション・サーバまたはワークステーション・クライアント・プロセスの現在の作業ディレクトリに作成されます (適切な書き込みパーミッションが設定されている場合)。

これらの変数の詳細については、『C 言語を使用した BEA Tuxedo アプリケーションのプログラミング』、『BEA Tuxedo アプリケーションの設定』、および『BEA Tuxedo アプリケーション実行時の管理』を参照してください。

フィールド・テーブル・ファイルおよび VIEW ファイル用の変数

FML および VIEWS で使用される環境変数は以下のとおりです。

FIELDTBLS

カンマで区切ったフィールド・テーブル・ファイルのリスト。

VIEWFILES

カンマで区切ったバイナリ VIEW ファイルのリスト。

FLDTBLDIR
: FIELDTBLS ファイルの検索先ディレクトリのコロン区切りのリスト。

VIEWDIR
: VIEWFILES ファイルの検索先ディレクトリのコロン区切りのリスト。

これらの変数の詳細については、『BEA Tuxedo アプリケーションの設定』、『BEA Tuxedo アプリケーション実行時の管理』、『C 言語を使用した BEA Tuxedo アプリケーションのプログラミング』、および『FML を使用した BEA Tuxedo アプリケーションのプログラミング』を参照してください。

ファイル・システムおよび TLOG 変数

以下の変数は、BEA Tuxedo システムのファイル・システムおよびトランザクション・ログで使用します。

FSCONFIG
: 汎用デバイス・リストのパス名。

FSMAXCOMMIT
: コミット・バッファの最大サイズを設定します。

FSMAXUPDATE
: 更新リストのサイズと更新回数の最大値を設定します。

FSMSGREP
: メッセージを繰り返す間隔を設定します。

FSOFFSET
: 汎用デバイス・リスト内のオフセットを指定します。

ワークステーション変数

以下の変数は、/WS クライアント・マシンで使用します。

TPMBENC
: 585 ページの「キー BEA Tuxedo システム変数」を参照してください。

TPMBACONV
: 585 ページの「キー BEA Tuxedo システム変数」を参照してください。

URLENTITYCACHING
: 585 ページの「キー BEA Tuxedo システム変数」を参照してください。

URLENTITYCACHEDIR
: 585 ページの「キー BEA Tuxedo システム変数」を参照してください。

WSALLOWPRE71
: BEA Tuxedo リリース 7.1 以降のソフトウェアを実行しているワークステーション・マシンと、BEA Tuxedo リリース 7.1 より前のアプリケーションを相互運用できるかどうかを指定します。変数を Y に設定すると (WSALLOWPRE71=Y)、相互運用が可能になります。

WSBUFFERS
: アプリケーションごとのパケット数。

WSDEVICE
: ネットワークのアクセスで使用するネットワーク・デバイス。BEA Tuxedo リリース 6.4 以降のワークステーション・クライアントの場合、この変数は必要ありません。

WSENVFILE
: ワークステーション・クライアント環境変数を含むファイルのパス名。

WSFADDR
: 別のマシンに接続する際にワークステーション・クライアントが使用するネットワーク・アドレス。この変数は、WSFRANGE 変数と共に、アウトバウンド接続を確立する前にプロセスがバインドを試みる TCP/IP ポートの範囲を決定します。

WSFRANGE
: アウトバウンド接続を確立する前にネイティブ・プロセスがバインドを試行する TCP/IP ポートの範囲。WSFADDR 変数は、この範囲のベース・アドレスを指定します。

WSNADDR
: ネイティブ・サイト・ネットワーク・リスナのネットワーク・アドレス

WSRPLYMAX
: メッセージを転送用ファイルにダンプする前の最大メッセージサイズ。

WSTYPE
: ワークステーションのマシン・タイプ。

これらの変数の詳細については、『BEA Tuxedo Workstation コンポーネント』を参照してください。

BEA Tuxedo /Q 変数

以下の変数は、BEA Tuxedo /Q で使用します。

`QMCONFIG`
: BEA Tuxedo /Q でキュー・スペースを使用できるデバイスを設定します。

詳細については、『BEA Tuxedo /Q コンポーネント』を参照してください。

COBOL 変数

以下の環境変数は、COBOL で使用します。

`ALTCC`
: COBOL のコンパイルに使用するコンパイラを指定します。

`ALTCFLAGS`
: COBOL コンパイラに渡すフラグ。

注記　Windows システムでは、環境変数 `ALTCC` および `ALTCFLAGS` は使用できません。これらを設定すると予期しない結果が生じます。最初に COBOL コンパイラを使用してアプリケーションをコンパイルし、次に生成されたオブジェクト・ファイルを `buildclient(1)` または `buildserver(1)` コマンドに渡す必要があります。

`COBCPY`
: COBOL の複写ファイルを検索するディレクトリ。

`COBDIR`
: COBOL のコンパイラ・ソフトウェアを入れるディレクトリを指定します。

`COBOPT`
: COBOL コンパイラ用のコマンド行引数を指定します。

これらの変数の詳細については、『COBOL を使用した BEA Tuxedo アプリケーションのプログラミング』を参照してください。

その他の変数

以下の環境変数も使用できます。

`MHSCACHE`
: オープンしておくメッセージ・カタログ (BEA Tuxedo システム・メッセージのみ) の数を指定します。デフォルト値は 3 です。

PMID

MP モードで、物理マシン ID を指定できます。また、可用性の高い (HA) 環境では、PMID を使用して、UBBCONFIG ファイルで指定されたマシン名を代替マシン名に置き換えることができます。これにより、マスタ・マシンをマスタから HA クラスタ内のバックアップに移行できます。

TAGENTLOG

tlisten(1) ログのパス名を指定します。

TMCMPLIMIT

メッセージを圧縮するかどうかを指定し、ローカル・メッセージとリモート・メッセージのしきい値を設定します。この変数の構文は次のとおりです。

TMCMPLIMIT=[*remote_threshold*[,*local_threshold*]]

しきい値は、0 から MAXLONG までの数字です。この数字により、データ圧縮を行うメッセージの最小バイト・サイズが設定されます。

TMCMPPRFM

プロセスの圧縮レベルを設定します。有効値は整数 1 から 9 までです。1 を設定すると、圧縮レベルは若干落ちますが、処理時間が短くなります。プロセスが TMCMPPRFM を読み取ると、ULOG メッセージが書き込まれます。

TMNETLOAD

ネットワークのロード・バランシングを設定します。この値は、リモート・サービスのロード・ファクタに加えられる任意のユニット数です。この変数を使用すると、ローカル・サービスが強制的に使用されます。

TMNOTHREADS

マルチスレッド処理をオフにするには、この変数を yes に設定します。スレッドを使用しないアプリケーションの場合、マルチスレッド処理をオフにすると、ミューテックス関数の呼び出しが減り、パフォーマンスが大幅に向上します。

TMSICACHEENTRIESMAX

プロセス単位でキャッシュするサービスおよびインターフェイスの量を指定します。有効値は 0 から 32,767 までの整数です。この変数の設定値は、UBBCONFIG ファイルに指定される値に優先します。

UIMMEDSIGS
信号の遅れを無効にするには、この変数を Y に設定します。

関連項目 buildclient(1)、buildserver(1)、viewc、viewc32(1)

UNIX システムのリファレンス・マニュアルの cc(1)、environ(5)

tuxtypes(5)

名前 tuxtypes— バッファ・タイプ・スイッチ、BEA Tuxedo システムによって提供されるバッファ・タイプの説明

形式 デフォルトのバッファ・タイプ・スイッチ

```
/*
 * 以下の定義は、
 * $TUXDIR/lib/tmtypesw.c に指定されている
 */

#include <stdio.h>
#include <tmtypes.h>

/*
 * バッファ・タイプ・スイッチの初期化
 */

struct tmtype_sw_t tm_typesw[] = {
{
      "CARRAY",     /* type */
      "*",          /* subtype */
      0             /* dfltsize */
      NULL,         /* initbuf */
      NULL,         /* reinitbuf */
      NULL,         /* uninitbuf */
      NULL,         /* presend */
      NULL,         /* postsend */
      NULL,         /* postrecv */
      NULL,         /* encdec */
      NULL,         /* route */
      NULL,         /* filter */
      NULL,         /* format */
      NULL,         /* presend2 */
      NULL          /* multibyte code-set encoding conversion */
},
{
      "STRING",        /* type */
      "*",             /* subtype */
      512,             /* dfltsize */
      NULL,            /* initbuf */
      NULL,            /* reinitbuf */
      NULL,            /* uninitbuf */
      _strpresend,     /* presend */
```

```
        NULL,            /* postsend */
        NULL,            /* postrecv */
        _strencdec,      /* encdec */
        NULL,            /* route */
        _sfilter,        /* filter */
        _sformat,        /* format */
        NULL,            /* presend2 */
        NULL            /* multibyte code-set encoding conversion */
    },
    {
        "FML",           /* type */
        "*",             /* subtype */
        1024,            /* dfltsize */
        _finit,          /* initbuf */
        _freinit,        /* reinitbuf */
        _funinit,        /* uninitbuf */
        _fpresend,       /* presend */
        _fpostsend,      /* postsend */
        _fpostrecv,      /* postrecv */
        _fencdec,        /* encdec */
        _froute,         /* route */
        _ffilter,        /* filter */
        _fformat,        /* format */
        NULL,            /* presend2 */
        NULL            /* multibyte code-set encoding conversion */
    },
    {
        "VIEW",          /* type */
        "*",             /* subtype */
        1024,            /* dfltsize */
        _vinit,          /* initbuf */
        _vreinit,        /* reinitbuf */
        NULL,            /* uninitbuf */
        _vpresend,       /* presend */
        NULL,            /* postsend */
        NULL,            /* postrecv */
        _vencdec,        /* encdec */
        _vroute,         /* route */
        _vfilter,        /* filter */
        _vformat,        /* format */
        NULL,            /* presend2 */
        NULL             /* マルチバイト・コード・セット符号化変換 */
    },
    {
        /* XATMI - CARRAY と同じ */
        "X_OCTET",       /* type */
        "*",             /* subtype */
        0                /* dfltsize */
    },
```

```
{    /* XATMI - VIEW と同じ */

      {'X','_','C','_','T','Y','P','E'},      /* type */
      "*",              /* subtype */
      1024,             /* dfltsize */
      _vinit,           /* initbuf */
      _vreinit,         /* reinitbuf */
      NULL,             /* uninitbuf */
      _vpresend,        /* presend */
      NULL,             /* postsend */
      NULL,             /* postrecv */
      _vencdec,         /* encdec */
      _vroute,          /* route */
      _vfilter,         /* filter */
      _vformat,         /* format */
      NULL,             /* presend2 */
      NULL              /* マルチバイト・コード・セット符号化変換 */
},
{
     /* XATMI - VIEW と同じ */
      {'X','_','C','O','M','M','O','N'},      /* type */
      "*",              /* subtype */
      1024,             /* dfltsize */
      _vinit,           /* initbuf */
      _vreinit,         /* reinitbuf */
      NULL,             /* uninitbuf */
      _vpresend,        /* presend */
      NULL,             /* postsend */
      NULL,             /* postrecv */
      _vencdec,         /* encdec */
      _vroute,          /* route */
      _vfilter,         /* filter */
      _vformat,         /* format */
      NULL,             /* presend2 */
      NULL              /* マルチバイト・コード・セット符号化変換 */
},
{
      "FML32",          /* type */
      "*",              /* subtype */
      1024,             /* dfltsize */
      _finit32,         /* initbuf */
      _freinit32,       /* reinitbuf */
      _funinit32,       /* uninitbuf */
      _fpresend32,      /* presend */
      _fpostsend32,     /* postsend */
      _fpostrecv32,     /* postrecv */
      _fencdec32,       /* encdec */
      _froute32,        /* route */
```

```
        _ffilter32,      /* filter */
        _fformat32,      /* format */
        _fpresend232     /* presend2 */


        _fmbconv32       /* マルチバイト・コード・セット符号化変換 */
    },
    {
        "VIEW32",        /* type */
        "*",             /* subtype */
        1024,            /* dfltsize */
        _vinit32,        /* initbuf */
        _vreinit32,      /* reinitbuf */
        NULL,            /* uninitbuf */
        _vpresend32,     /* presend */
        NULL,            /* postsend */
        NULL,            /* postrecv */
        _vencdec32,      /* encdec */
        _vroute32,       /* route */
        _vfilter32,      /* filter */
        _vformat32,      /* format */
        NULL,            /* presend2 */
        NULL             /* マルチバイト・コード・セット符号化変換 */
    },
    {
       "XML",            /* type */
        "*",             /* subtype */
        0,               /* dfltsize */
        NULL,            /* initbuf */
        NULL,            /* reinitbuf */
        NULL,            /* uninitbuf */
        NULL,            /* presend */
        NULL,            /* postsend */
        NULL,            /* postrecv */
        NULL,            /* encdec */
        _xroute,         /* route */
        NULL,            /* filter */
        NULL,            /* format */
        NULL,            /* presend2 */
        NULL             /* マルチバイト・コード・セット符号化変換 */
    },
    {
        "MBSTRING",     /* type */
        "*",            /* subtype */
        0,              /* dfltsize */
        _mbsinit,     /* initbuf */
        NULL,           /* reinitbuf */
        NULL,           /* uninitbuf */
        NULL,           /* presend */
```

```
        NULL,           /* postsend */
        NULL,           /* postrecv */
        NULL,           /* encdec */
        NULL,           /* route */
        NULL,           /* filter */


        NULL,           /* format */
        NULL,           /* presend2 */
        _mbsconv      /* マルチバイト・コード・セット符号化変換 */
},
{
""
}
};

struct tmtype_sw_t _TM_FAR *
_TMDLLENTRY
_tmtypeswaddr(void)
{
  return(tm_typesw);
}
```

機能説明　次の表に、BEA Tuxedo システムの 11 のバッファ・タイプを示します。

CARRAY	送信時に符号化も復号化も行われない文字配列 (多くの場合 NULL 文字を含む)。
STRING	NULL で終了する文字配列。
FML	FML フィールド化バッファ。
VIEW	C 構造体または FML VIEW。
X_OCTET	CARRAY に相当するもので、XATMI との互換性のために提供されています。
X_C_TYPE	VIEW に相当するもので、XATMI との互換性のために提供されています。
X_ COMMON	VIEW に相当するもので、XATMI との互換性のために提供されています。
FML32	32 ビット識別子またはオフセットを使用した、FML32 フィールド化バッファ。

VIEW32	32 ビット識別子、カウンタ変数、およびサイズ変数を使用した、C 構造体または FML32 VIEW。
XML	XML 文書用のバッファ
MBSTRING	マルチバイト文字用の文字配列。

すべての VIEW、X_C_TYPE、および X_COMMON バッファは同じルーチン・セットで処理され、特定の VIEW の名前はそのサブタイプの名前です。

カスタム・バッファ・タイプを指定する場合には、上記の tm_typesw 配列にインスタンスを追加します。新しいバッファ・タイプを追加したり、既存のバッファ・タイプを削除したりする場合は、上に示したように配列の最後に NULL エントリを残しておく必要があります。バッファ・タイプに NULL 名を指定することはできません。

デフォルトの配列のコピーは $TUXDIR/lib/tmtypesw.c で配布され、開始点として使用されます。新しいバッファ・タイプ・スイッチをインストールする手順として推奨されるのは、tmtypesw.c をコンパイルし、libbuft というライブラリ内に唯一の要素として格納することです。

共有オブジェクト機能を持つシステムでは、$TUXDIR/lib の下に libbuft.so. という新しいインスタンスを作成およびインストールします。WSH などの BEA Tuxedo システム・プロセスを含むすべてのプロセスは、再コンパイルしなくても新しいタイプ・スイッチに自動的にアクセスできるようになります。Windows のワークステーションでは、バッファ・タイプ・スイッチのための共有オブジェクトは WBUFT.DLL です。これは、$TUXDIR\bin に格納される必要があります。

共有オブジェクト機能をもたないシステムでは、$TUXDIR/lib の下に libbuft.a という新しいインスタンスを作成およびインストールします。新しいタイプについて知る必要があるすべてのプロセスは、buildclient(1) または buildserver(1) を使用して再作成する必要があります。WSH のようなシステム・プロセスは、buildwsh(1) などの特別なコマンドで再作成する必要があります。

バッファ・タイプ・スイッチの要素とルーチンについては、buffer(3c) を参照してください。そこでは、システム標準のバッファ・タイプを変更するときにアプリケーションが使用できる BEA Tuxedo システムのデフォルト・ルーチン (_finit() など) についても説明されています。

システムに用意されている _froute()、_vroute()、および _xroute() の 3 つのルーティング関数は、それぞれ FML バッファ、VIEW バッファ、および XML バッファのデータ依存型ルーティングに使用されます。これら 3 つの関数で使用するルーティング基準の定義方法については、UBBCONFIG(5) を参照してください。

ファイル　$TUXDIR/tuxedo/include/tmtypes.h― タイプ・スイッチの定義
$TUXDIR/lib/tmtypesw.c― デフォルト・タイプ・スイッチのインスタンス
$TUXDIR/lib/libbuft.so.― タイプ・スイッチ共有オブジェクト
$TUXDIR/lib/libbuft.a― タイプ・スイッチ・アーカイブ・ライブラリ

関連項目　buffer(3c)、typesw(5)、UBBCONFIG(5)

typesw(5)

名前 typesw— バッファ・タイプ・スイッチ構造体、各バッファ・タイプに必要なパラメータとルーチン

形式 バッファ・タイプ構造体

```
/*
 * 以下の定義は、$TUXDIR/include/tmtypes.h に存在する
 */
#define TMTYPELEN  ED_TYPELEN
#define TMSTYPELEN ED_STYPELEN

struct tmtype_sw_t {
   char type[TMTYPELEN];      /* バッファのタイプ */
   char subtype[TMSTYPELEN];  /* バッファのサブタイプ */
   long dfltsize;             /* バッファのデフォルト・サイズ */
   /* バッファ初期化関数ポインタ */
   int (_TMDLLENTRY *initbuf) _((char _TM_FAR *, long));

   /* バッファ再初期化関数ポインタ */
   int (_TMDLLENTRY *reinitbuf) _((char _TM_FAR *, long));

   /* バッファ非初期化関数ポインタ */
   int (_TMDLLENTRY *uninitbuf) _((char _TM_FAR *, long));

   /* バッファ送信前操作関数ポインタ */
   long (_TMDLLENTRY *presend) _((char _TM_FAR *, long, long));

   /* バッファ送信後操作関数ポインタ */
   void (_TMDLLENTRY *postsend) _((char _TM_FAR *, long, long));

   /* バッファ受信後操作関数ポインタ */
   long (_TMDLLENTRY *postrecv) _((char _TM_FAR *, long, long));

   /* XDR 符号化 / 復号化関数ポインタ */
  long (_TMDLLENTRY *encdec) _((int, char _TM_FAR *, long, char _TM_FAR *, long));

   /* ルーティング関数ポインタ */
   int (_TMDLLENTRY *route) _((char _TM_FAR *, char _TM_FAR *, char _TM_FAR *,
       long, char _TM_FAR *));

   /* バッファ・フィルタ関数ポインタ */
   int (_TMDLLENTRY *filter) _((char _TM_FAR *, long, char _TM_FAR *, long));
```

```
    /* バッファ・フォーマット関数ポインタ */
    int (_TMDLLENTRY *format) _((char _TM_FAR *, long, char _TM_FAR *,
        char _TM_FAR *, long));

    /* 送信前のバッファの処理、コピーを生成する場合がある */
    long (_TMDLLENTRY *presend2) _((char _TM_FAR *, long,
        long, char _TM_FAR *, long, long _TM_FAR *));

    /* マルチバイト・コード・セット符号化変換関数ポインタ */
    long (_TMDLLENTRY *mbconv) _((char _TM_FAR *, long,
        char _TM_FAR *, char _TM_FAR *, long, long _TM_FAR *));


    /* この領域は将来の拡張のために予約されている */
    void (_TMDLLENTRY *reserved[8]) _((void));
};
/*
 * アプリケーション・タイプ・スイッチ・ポインタ
 * テーブルへのアクセス時には常にこのポインタを使用
 */
extern struct tmtype_sw_t *tm_typeswp;
```

機能説明　バッファ・タイプとサブタイプには、バッファが操作されるときに適切なルーチンが呼び出されるようにするため、`tm_typesw` 配列内にエントリが必要です。BEA Tuxedo システムに用意されているバッファ・タイプについては、`tuxtypes(5)` を参照してください。

カスタマイズしたバッファ・タイプを指定する場合は、`$TUXDIR/lib/tmtypesw.c` の `tm_typesw` 配列にインスタンスを追加します。この方法については、`tuxtypes(5)` を参照してください。新しいタイプを追加する場合に指定する必要があるルーチンのセマンティクスについては、`buffer(3c)` を参照してください。

ファイル　`$TUXDIR/tuxedo/include/tmtypes.h`― タイプ・スイッチの定義
`$TUXDIR/lib/tmtypesw.c`― タイプ・スイッチのインスタンス

関連項目　`buffer(3c)`, `tuxtypes(5)`

UBBCONFIG(5)

名前 UBBCONFIG— テキスト形式の BEA Tuxedo コンフィギュレーション・ファイル

機能説明 BEA Tuxedo アプリケーションが起動するとき、tmboot コマンドは TUXCONFIG というバイナリ・コンフィギュレーション・ファイルを参照して、アプリケーション・サーバの起動処理と掲示板の初期化処理を順番に行うために必要な情報を取得します。このバイナリ・ファイルは直接作成できるものではなく、UBBCONFIG と呼ばれるテキスト・ファイルから作成する必要があります。アプリケーションをコンフィギュレーションするには、管理者はテキスト・エディタで UBBCONFIG ファイルを作成し、次に tmloadcf(1) コマンドを実行してそのファイルをバイナリ形式の TUXCONFIG にロードします。アプリケーションが実行されている間、TUXCONFIG ファイルはさまざまな BEA Tuxedo 管理ツールによって使用されます。tmadmin(1) は、システムの監視活動時にコンフィギュレーション・ファイル (またはそのコピー) を使用します。また、tmshutdown(1) はコンフィギュレーション・ファイルを参照して、アプリケーションをシャットダウンするために必要な情報を調べます。

BEA Tuxedo UBBCONFIG ファイルにはどのような名前を付けることもできますが、ファイルの内容はこのリファレンス・ページで説明する形式に準拠している必要があります。また、TUXCONFIG ファイルにも任意の名前を付けることができますが、実際の名前は TUXCONFIG 環境変数で指定されたデバイス・ファイル名またはシステム・ファイル名になります。

UBBCONFIG ファイル全体の詳細については、669 ページの「UBBCONFIG(5) に関する追加情報」を参照してください。

定義 サーバとは、要求を受け付け、その応答をクライアントや別のサーバに送信するプロセスです。一方、クライアントは要求を送信し、その応答を受け取ります。

リソース・マネージャとは、情報やプロセス (またはその両方) の集まりへのアクセスを提供するインターフェイスおよび関連ソフトウェアです。リソース・マネージャの例としては、データベース管理システムがあります。リソース・マネージャのインスタンスとは、DBMS によって制御される、特定のデータベースのインスタンスのことです。分散トランザクションとは、

複数のリソース・マネージャのインスタンスにまたがるトランザクションのことで、tpbegin() で開始され、tpcommit() または tpabort() で完了します。

サーバ・グループはリソース・マネージャのインスタンスであり、特定のマシン上に配置されたこのリソース・マネージャ・インスタンスへのアクセスを提供するサーバやサービスの集合です。このサーバ・グループに関連付けられている XA インターフェイスは、トランザクション管理に使用されます。サーバがリソース・マネージャのインスタンスにアクセスしないか、または分散型トランザクションの一部としてリソース・マネージャのインスタンスにアクセスしない場合は、そのサーバはサーバ・グループにあって、空の XA インターフェイスを持つ必要があります。NULL 同様に、クライアントは GROUPS セクションで指定する必要のない、特別なクライアント・グループ内で動作します。このクライアント・グループはリソース・マネージャには関連付けられません。

リモート・ドメインは、この BEA Tuxedo システム・コンフィギュレーションの掲示板が使用できない環境として定義されます。リモート・ドメインは UBBCONFIG コンフィギュレーション・ファイルには指定せず、ホスト固有のリファレンス・ページに指定されているホスト固有の環境変数を介して定義します。

コンフィギュレーション・ファイルの形式

UBBCONFIG ファイルは、9 つの指定セクションで構成されます。先頭にアスタリスク (*) が付いている行は、指定セクションの始まりを示します。このような行にはそれぞれ、* のすぐ後にセクション名が含まれています。使用可能なセクションは次のとおりです。

- RESOURCES
- MACHINES
- GROUPS
- NETGROUPS
- NETWORK
- SERVERS
- SERVICES
- INTERFACES

■ ROUTING

RESOURCES および MACHINES セクションは、この順序で最初に置く必要があります。GROUPS セクションは、SERVERS、SERVICES、および ROUTING セクションの前になければなりません。NETGROUPS セクションは、NETWORK セクションの前になければなりません。

RESOURCES セクション以外のパラメータは、一般に *KEYWORD* = *value* という形式で指定します。等号 (=) の両側で、空白類 (スペースやタブ文字) を使用できます。この形式により、*KEYWORD* が *value* に設定されます。有効なキーワードについては、以下の各セクションで説明します。

予約語の DEFAULT で始まる行にはパラメータ指定が含まれており、セクション内の以降の該当するすべての行に対して適用されます。デフォルトの指定は RESOURCES セクション以外のすべてのセクションで使用することができ、また 1 つのセクションで複数回使用することもできます。これらの行のフォーマットは次のとおりです。

DEFAULT: [optional *KEYWORD*=*value* pairs]

この行で設定した値は、別の DEFAULT 行によってリセットされるか、セクションが終わるまで有効です。これらの値は、DEFAULT でない行のオプション・パラメータによって無効になる場合もあります。DEFAULT でない行におけるパラメータ設定は、その行でのみ有効です。以降の行ではデフォルト設定に戻ります。DEFAULT が行頭に表示されると、それ以前に設定されたすべてのデフォルト値はクリアされ、システムのデフォルト値に戻ります。

値が *numeric* の場合は、C の標準表記法を使用して基数を示します。つまり、基数 16 (16 進) の接頭辞は 0x、基数 8 (8 進) の接頭辞は 0、基数 10 (10 進) には接頭辞が付きません。数値パラメータに指定できる値の範囲は、そのパラメータの説明の下に示されています。

値が *identifier* (SECURITY パラメータの APP_PW のように BEA Tuxedo システムにとって既知の文字列値) の場合、一般的に標準 C 規則が使用されます。標準 C の *identifier* の先頭には英字またはアンダースコア (_) を使用し、以降の識別子には英数字またはアンダースコアを使用する必要があります。identifier の長さは最大 30 バイトです (最後のヌルを除く)。

注記　識別子を二重引用符で囲む必要はありません。

整数でも識別子でもない値は、二重引用符で囲む必要があります。この値はユーザ定義の *string* です。ユーザ定義の文字列は、最後のヌル文字を除き最大 78 文字 (バイト) です。この規則には、以下のような例外があります。

- CLOPT、BUFTYPE、OPENINFO、および CLOSEINFO パラメータの長さは最大 256 文字です。
- The TUXCONFIG,TUXDIR、APPDIR、TLOGDEVICE、ULOGPFX、ENVFILE、TMSNAME、RCMD、NADDR、NLSADDR、FADDR、および AOUT (SERVERS セクション) パラメータの長さは、BEA Tuxedo リリース 8.1 では最大 256 文字です。BEA Tuxedo 8.0 以前では、これらの値の文字列値は 78 文字に制限されます。
- SEC_PRINCIPAL_NAME、SEC_PRINCIPAL_LOCATION、および SEC_PRINCIPAL_PASSVAR パラメータの長さは最大 511 文字です (最後のヌルを除く)。
- RANGES パラメータの長さは最大 2,048 文字です (ただし Domains の場合は最大 4,096 文字)。

ROUTING セクションの RANGES パラメータでは、特定の文字はバックスラッシュを用いることによって文字列の中でエスケープすることができます。

"\\" は 1 つのバックスラッシュ
"\"" は二重引用符
"\n" は復帰改行
"\t" はタブ
"\f" は用紙送り
"\O+" は 8 進数の値が 0+ である文字と解釈

O+ は 1 桁、2 桁、または 3 桁の 8 進文字を表します。"\0" は、埋め込みヌル文字と解釈されます。"\xH+" または "\XH+" は、16 進数の値が H+ である文字と解釈されます。H+ は 1 桁または複数桁の 16 進文字です。"\y" (‘y’ は上記以外のすべての文字) は、‘y’ と解釈されます。

"#" (シャープ記号) はコメントを示します。復帰改行文字でコメントを終了します。

識別子または数値定数には、常に空白類 (スペースまたはタブ文字)、復帰改行文字、または句読文字 (シャープ記号、等号、アスタリスク、コロン、カンマ、バックスラッシュ、またはピリオド) が付加されます。

空白行とコメントは無視されます。

コメントは任意の行の最後に自由に入力できます。

行は、復帰改行の後に最低 1 つのタブを置いて継続できます。コメントを継続することはできません。

RESOURCES セクション

このセクションでは、サーバ数やサービス領域に存在できるサービス数など、システム全体のリソースを指定します。RESOURCES セクションの行は、*KEYWORD value* という形式を取ります。*KEYWORD* はパラメータの名前、*value* はそれに対応する値です。有効な *KEYWORD* は以下のとおりです。

IPCKEY *numeric_value*

BEA Tuxedo システムの掲示板における既知のアドレスの数値キーを指定します。単一のプロセッサを使用している環境では、このキーは掲示板を指定します。複数のプロセッサからなる環境では、このキーは DBBL のメッセージ・キューを指定します。また、このキーは、既知のアドレスのほか、マルチプロセッサ全体の掲示板などのリソースの名前を取り出す基準としても使用されます。IPCKEY の値は、32,768 より大きく 262,143 未満でなければなりません。このパラメータは必須です。

MASTER *string_value1*[,*string_value2*]

TUXCONFIG ファイルのマスタ・コピーのあるマシンを指定します。また、アプリケーションが MP モードで動作している場合、MASTER は DBBL が実行されるマシンを指定します。*string_value2* は、プロセスの再配置および起動時に使用される LMID の代替位置を指定します。本来の位置が使用できない場合、DBBL はこの代替位置で起動し、その位置にある代替 TUXCONFIG ファイルが使用されます。LMID の値はどちらも MACHINES セクションにあるマシンを指定する必要があり、またどちらも 30 文字以下でなければなりません。このパラメータは必須です (SHM モードの場合でも)。

異なったマシ上で BEA Tuxedo の複数のリリース・レベルをサポートするアプリケーションでは、MASTER と BACKUP は常にすべてのマシンよりも上位のリリースを持っていなければなりません。この規則は "Hot Upgrade." の間は強制されません。

MODEL {SHM | MP}

コンフィギュレーションのタイプを指定します。このパラメータは必須で、この 2 つの値のうちいずれか 1 つしか指定できません。SHM (共用メモリ) は、単一マシン用のコンフィギュレーションを指定します。MACHINES セクションに指定できるマシンは 1 つだけです。MP は、複数マシン用のコンフィギュレーションを指定します。ネットワーク化

されたアプリケーションを定義する場合は、MP を指定する必要があります。注記 : 再リンクせずに *value* を変更するには、必要なモデルをサポートするようにサーバを構築しておく必要があります (buildserver(1) を参照)。

DOMAINID *string_value*

ドメイン ID 文字列を指定します。指定しない場合は "" が使用されます。DOMAINID の値が文字列の場合、後続のヌル文字も含め最大 30 文字まで使用できます。DOMAINID の値が 16 進数の文字列の場合、最大 30 オクテットまで使用できます。DOMAINID が指定されている場合、その値は、特定のドメインに関連付けられているプロセスで通知される任意のコマンド出力 (ps コマンドの出力など) に、パラメータ (-C dom=*domainid*) として含まれます。このコメントは、複数のドメインを管理する管理者にとっては役に立ちます。このコメントがないと、複数のドメインを参照する単一の出力ストリームを解釈するのが難しくなる場合があります。

UID *numeric_value*

掲示板用に作成された IPC 構造体に関連付ける数値ユーザ ID を指定します。この値は、ローカルの UNIX システム上のユーザ ID です。このパラメータの指定がない場合は、tmloadcf(1) を実行するユーザの有効ユーザ ID となる値が取られます。*RESOURCES* セクションの値は、プロセッサごとに MACHINES セクションで変更できます。

GID *numeric_value*

掲示板用に作成された IPC 構造体に関連付ける数値グループ ID を指定します。この値は、ローカルの有効な UNIX システム上のグループ ID です。GID の指定がない場合は、tmloadcf(1) を実行するユーザの有効グループ ID が取られます。RESOURCES セクションの値は、プロセッサごとに MACHINES セクションで変更できます。

PERM *numeric_value*

掲示板をインプリメントする IPC 構造体に関連付ける数値パーミッションを指定します。このパラメータは、通常の UNIX システム形式 (0600 のような 8 進数値) で、プロセスに対する読み取りまたは書き込みパーミッションを指定するために使用します。このパラメータを指定しない場合、IPC 構造体に対するパーミッションのデフォルトは 0666 (同じユーザ、同じグループ、およびその他すべてのユーザによる読み取り / 書き込みアクセス) です。値は 0001 以上 0777 以下の範

囲で指定できます。RESOURCES セクションの値は、プロセッサごとに MACHINES セクションで変更できます。

MAXACCESSERS *numeric_value*

このアプリケーション内の特定のマシンの掲示板に同時接続できるクライアントおよびサーバのデフォルトの最大数を指定します。この値は 0 より大きく、32,768 未満でなければなりません。指定しない場合、デフォルトの最大数は 50 になります。このパラメータの RESOURCES セクションの値は、マシンごとに MACHINES セクションで変更できます。

BBL、restartsrv、cleanupsrv、tmshutdown()、tmadmin() などのシステム管理プロセスはこの数に入れる必要はありません。ただし、DBBL、すべてのブリッジ・プロセス、すべてのシステム提供サーバ・プロセスとアプリケーション・サーバ・プロセス、および特定のサイトで使用する可能性があるクライアント・プロセスは数に入れる必要があります。システム提供のサーバには、AUTHSVR、TMQUEUE、TMQFORWARD、TMUSREVT、TMSYSEVT、TMS などがあります。GROUPS セクションの TMSNAME パラメータ、TMS_QM、GWTDOMAIN、および WSL を参照してください。特定のサイトでアプリケーションがワークステーション・リスナ (WSL) を起動する場合は、起動される WSL と使用する可能性があるワークステーション・ハンドラ (WSH) の両方をこの数に入れる必要があります。

BEA Tuxedo リリース 7.1 より前のリリース (6.5 以前) では、アプリケーションの MAXACCESSERS および MAXSERVERS パラメータは、ユーザ・ライセンス数をチェックする仕組みで利用されます。つまり、アプリケーションで実行中の 1 台以上のマシンの MAXACCESSERS の数と、特定のマシンの MAXACCESSERS の数の合計が、MAXSERVERS の数とユーザ・ライセンス数の合計より大きい場合、マシンを起動することはできません。したがって、アプリケーションの MAXACCESSERS パラメータには、MAXSERVERS の数とユーザ・ライセンス数の合計か、またはそれより小さい値を指定しなければなりません。

また、BEA Tuxedo リリース 7.1 以降のユーザ・ライセンス数をチェックする仕組みで考慮されるのは、アプリケーションのユーザ・ライセンス数とそのアプリケーションで現在使用中のラインセンス数の 2 つだけです。すべてのユーザ・ライセンスが使用中になると、ア

プリケーションに新しいクライアントが参加することはできなくなります。

MAXSERVERS *numeric_value*

このアプリケーションで掲示板のサーバ・テーブルに登録できるサーバの最大数を指定します。この値は、0 より大きく 8,192 未満でなければなりません。指定しない場合のデフォルトは 50 です。

アプリケーションで利用可能なシステム提供のサーバおよびアプリケーション・サーバのすべてのインスタンスを、掲示板のサーバ・テーブルで指定する必要があります。このテーブルはグローバル・テーブルであり、同じサーバ・テーブルがアプリケーションの各マシン上にあります。システム提供のサーバには、AUTHSVR、TMQUEUE、TMQFORWARD、TMUSREVT、TMSYSEVT、TMS などがあります。GROUPS セクションの TMSNAME パラメータ、TMS_QM、GWTDOMAIN、および WSL を参照してください。

BEA Tuxedo システムを使用しているサイトを管理するには、1 サイトあたりほぼ 1 つのサーバが必要です。さらに、DBBL プロセスとすべての BBL、ブリッジ、および WSH プロセスも MAXSERVERS の値に入れてください。

MAXSERVICES *numeric_value*

掲示板のサービス・テーブルに登録されるサービスの最大総数を指定します。この値は 0 より大きく、32,768 未満でなければなりません。適切な値を計算するには、アプリケーション・サーバとシステム・サーバ (BBL、DBBL、BRIDGE、TMS など) で使用されるサービスの数と、アプリケーションで必要なそれ以外のシステム・サーバで使用されるサービスの数を数えます。指定しない場合、デフォルトは 100 です。

MAXGROUPS *numeric_value*

掲示板のグループ・テーブルに登録するコンフィギュレーション済みのサーバ・グループの最大数を指定します。この値は、100 以上 32,768 未満でなければなりません。指定しない場合のデフォルトは 100 です。

MAXNETGROUPS *numeric_value*

TUXCONFIG ファイルの NETWORK セクションに指定されるコンフィギュレーション済みのネットワーク・グループの最大数を指定します。こ

の値は、1 以上 8,192 未満でなければなりません。指定しない場合のデフォルトは 8 です。

MAXMACHINES *numeric_value*

掲示板のマシン・テーブルに登録するコンフィギュレーション済みのマシンの最大数を指定します。この値は、256 以上 8,191 未満でなければなりません。指定しない場合のデフォルトは 256 です。

MAXQUEUES *numeric_value*

掲示板のグループ・テーブルに登録するサーバ要求キューの最大数を指定します。この値は 1 以上、8,192 未満でなければなりません。指定しない場合は、MAXSERVERS の値に設定されます。5.0 より前のリリースと相互運用するためには、この値は MAXSERVERS の値と等しくなければなりません。

MAXACLGROUPS *numeric_value*

ACL のパーミッション・チェックに使用できるグループ ID の最大数を指定します。定義可能なグループ ID 数は、TA_MAXACLGROUPS - 1 です。この値は、1 以上 16,384 以下でなければなりません。指定しない場合のデフォルトは 16,384 です。

MAXGTT *numeric_value*

このアプリケーション内の特定のマシンが同時に関与できるグローバル・トランザクションの最大数を指定します。この値は 0 以上 32,768 未満でなければなりません。指定しない場合、デフォルトは 100 です。このパラメータの RESOURCES セクションの値は、マシンごとに MACHINES セクションで変更できます。

MAXCONV *numeric_value*

このアプリケーション内の特定のマシン上のクライアントおよびサーバが同時に関与できる会話の最大数を指定します。この値は 0 より大きく、32,768 未満でなければなりません。指定しない場合、SERVERS セクションに何らかの会話型サーバが定義されていればデフォルト値は 64 になり、それ以外のときは 1 になります。サーバ当たりの同時の会話の最大数は 64 です。このパラメータの RESOURCES セクションの値は、マシンごとに MACHINES セクションで変更できます。

MAXBUFTYPE *numeric_value*

掲示板のバッファ・タイプ・テーブルに登録できるバッファ・タイプの最大数を指定します。この値は、0 より大きく 32,768 未満でなければなりません。指定しない場合のデフォルトは 16 です。

MAXBUFSTYPE *numeric_value*

掲示板のバッファ・サブタイプ・テーブルに登録できるバッファ・サブタイプの最大数を指定します。この値は、0 より大きく 32,768 未満でなければなりません。指定しない場合のデフォルトは 32 です。

MAXDRT *numeric_value*

コンフィギュレーション済みのデータ依存型ルーティング基準のエントリの最大数を指定します。この値は 0 以上 32,768 未満でなければなりません。指定しない場合、デフォルトは ROUTING セクションの設定値です。

MAXRFT *numeric_value*

データ依存型ルーティング範囲フィールド・テーブル・エントリの最大数を指定します。この値は 0 以上 32,768 未満でなければなりません。指定しない場合、デフォルトは ROUTING セクションの設定値です。

MAXRTDATA *numeric_value*

データ依存型ルーティング範囲文字列用の掲示板の文字列プールに登録できる最大文字列プール領域を指定します。この値は 0 以上 32,761 未満でなければなりません。指定しない場合、デフォルトは ROUTING セクションの設定値です。

ROUTING セクションの RANGES 値に指定された文字列と carry は文字列プールに格納されます。実行時の拡大に備えて、領域を余分に割り当てる必要があります。

MAXSPDATA *numeric_value*

掲示板の共通文字列プールに登録できる最大文字列プール領域をバイトで指定します。この値は 0 以上 2,147,483,640 以下でなければなりません。デフォルトは 0 です。このパラメータは、BEA Tuxedo 8.1 以降のソフトウェアが実行されているアプリケーションにのみ適用されます。

ほとんどの場合、このパラメータのデフォルト値を使用すれば BEA Tuxedo システムは BEA Tuxedo で最大許容長が 256 バイトに拡張され

た TUXCONFIG パラメータ文字列用に十分な文字列プール領域を割り当てます。これらのパラメータは、TUXCONFIG、TUXDIR、APPDIR、TLOGDEVICE、ULOGPFX、ENVFILE、TMSNAME、RCMD、NADDR、NLSADDR、FADDR、および SERVERS セクションの AOUT です。

大幅な動的コンフィギュレーションが予想されるアプリケーションでは (6 台のマシンが BEA Tuxedo アプリケーションに追加されると予想される場合など)、管理者は MAXSPDATA パラメータを使用して共通文字列プールのサイズを拡大できます。共通文字列プールのサイズの変更は、MAXRTDATA パラメータで制御されるルーティング文字列プールのサイズに影響を与えません。2 つの文字列プールはそれぞれ別個のものです。

MAXSPDATA の指定値に関係なく、BEA Tuxedo システムは、TUXCONFIG ファイルで実際に指定されている文字列、および 256 バイト対応の文字列が指定された場合に必要な文字列プール・サイズに基づいて計算された範囲を外れた文字列プール・サイズを割り当てません。tmloadcf(1) コマンドは、指定された値がこの範囲を外れている場合に警告を表示し、その値を最も近い許容値に設定します。

最大許容長が 256 バイトに拡張された TUXCONFIG パラメータのうち、GROUPS セクションの TMSNAME パラメータと SERVERS セクションの AOUT および RCMD パラメータだけが実際に掲示板に格納されます。それ以外のパラメータは、プロセス起動時に読み込まれるか、またはプロセス・メモリに格納されます。

MAXTRANTIME *numeric_value*

この BEA Tuxedo アプリケーションで開始され、または受け取られたトランザクションの最大許容タイムアウトを秒単位で指定します。この値は 0 以上 2,147,483,647 以下でなければならず、デフォルトは 0 です (グローバル・トランザクション・タイムアウト制限は無効)。このパラメータは、BEA Tuxedo 8.1 以降のソフトウェアが実行されているアプリケーションにのみ適用されます。

MAXTRANTIME タイムアウトの値が AUTOTRAN サービスの TRANTIME タイムアウトの値またはトランザクションを開始するための tpbegin(3c) 呼び出しで渡されるタイムアウト値より小さい場合、トランザクションのタイムアウトは MAXTRANTIME の値に短縮されます。MAXTRANTIME は、BEA Tuxedo 8.0 以前のリリースが実行されているマシンで起動さ

れたトランザクションに対しては無効です。ただし、BEA Tuxedo 8.1 以降のソフトウェアが実行されているマシンがトランザクションの影響を受ける場合、トランザクション・タイムアウトは必要な場合はそのマシン用に設定されている MAXTRANTIME 値に短縮されます。

UBBCONFIG ファイルの SERVICES セクションに指定されている TRANTIME の値が MAXTRANTIME の値より大きい場合、tmloadcf(1) コマンドはエラーなしでコンフィギュレーションをロードします。AUTOTRAN トランザクションの影響を受ける BEA Tuxedo 8.1 以降の各マシンは、そのマシン用に設定されている MAXTRANTIME 値にトランザクション・タイムアウトを自動的に短縮します。

CMTRET {COMPLETE | LOGGED}

BEA Tuxedo システム・アプリケーションのすべてのクライアント・プロセスおよびサーバ・プロセスの TP_COMMIT_CONTROL 特性の初期設定を指定します。*value* が LOGGED の場合、TP_COMMIT_CONTROL 特性は TP_CMT_LOGGED に初期化され、それ以外の場合は TP_CMT_COMPLETE に初期化されます。CMTRET を指定しない場合、デフォルト COMPLETE です。この特性の設定の詳細については、BEA Tuxedo システム ATMI 関数 tpscmt の説明を参照してください。

LDBAL {Y | N}

ロード・バランシングを実行するかどうかを指定します。LDBAL を指定しない場合、デフォルトは Y です。各サービスが 1 つのキューにしかマップされない場合は、ロード・バランシングが自動になっているため、LDBAL を N に設定してください。

LDBAL を Y に設定すると、ロード・バランシングが自動的に実行されます。各インターフェイス要求は、負荷の合計が最も低いサーバにルーティングされます。ルーティング先のサーバの負荷は、ルーティングされた CORBA インターフェイスのロード・ファクタ (LOAD) だけ増加します。

ロード・バランシングが無効になっており、複数のサーバが同じ CORBA インターフェイスを提供する場合、使用可能な最初のキューが要求を受け取ります。

SYSTEM_ACCESS {FASTPATH | PROTECTED}[,NO_OVERRIDE]

アプリケーション・プロセス内で BEA Tuxedo システム・ライブラリが BEA Tuxedo システムの内部テーブルへのアクセス権を取得するた

めに使用するデフォルトのモードを指定します。FASTPATH を指定した場合、BEA Tuxedo システム・ライブラリは高速アクセス用のプロテクトされていない共用メモリを介して内部テーブルにアクセスできます。PROTECTED を指定した場合、BEA Tuxedo システム・ライブラリが共用メモリを介して内部テーブルにアクセスできる間、BEA Tuxedo システム・ライブラリの外部からはそれらのテーブルの共用メモリにアクセスできません。NO_OVERRIDE を単独あるいは FASTPATH または PROTECTED と共に指定した場合、tpinit(3c) または TPINITIALIZE(3cbl) で使用可能なフラグを使ってアプリケーション・プロセスで選択モードを変更することはできません。SYSTEM_ACCESS を指定しない場合、デフォルトのモードは FASTPATH です。

制限事項 :SYSTEM_ACCESS を PROTECTED に設定しても、マルチスレッド・サーバには効果がない場合があります。これは、あるスレッドが BEA Tuxedo コードを実行しているときに (つまりスレッドが掲示板にアタッチされているとき) 別のスレッドがユーザ・コードを実行できるからです。BEA Tuxedo では、このような状況を防止することはできません。

OPTIONS {[LAN | MIGRATE | NO_XA | NO_AA],*}

使用するオプションを指定します。2 つ以上のオプションを指定する場合は、それらをカンマで区切ります。識別子 LAN は、ネットワーク・アプリケーションであることを示します。識別子 MIGRATE は、サーバ・グループを移行できることを示します。MIGRATE を指定する場合は、LAN も指定する必要があります (このコンフィギュレーションが単一のマルチプロセッサ・コンピュータ上で動作する場合を除く)。識別子 NO_XA は、XA トランザクションが使用できないことを示します。識別子 NO_AA は、監査用および認可用の関数が呼び出されないことを示します。このパラメータはオプションなので、デフォルト値はありません。

USIGNAL {SIGUSR1 | SIGUSR2}

SIGNAL ベースの通知方法が使用される場合に用いられるシグナルを指定します。このパラメータの有効値は、SIGUSR1 と SIGUSR2 です。デフォルトは SIGUSR2 です。SIGNAL ベースの通知方法が NOTIFY パラメータで選択されていない場合でも、USIGNAL を指定できます。これは、tpinit() の呼び出し側がシグナル・ベースの通知方法を選択する場合があるためです。

SECURITY {NONE | APP_PW | USER_AUTH | ACL | MANDATORY_ACL}

使用するアプリケーション・セキュリティの種類を指定します。指定しない場合、このパラメータのデフォルトは NONE です。APP_PW を指定した場合、アプリケーションのパスワード・セキュリティ機能が使用されます (クライアントは初期化時にアプリケーション・パスワードを渡す必要があります)。APP_PW を指定すると、tmloadcf はアプリケーション・パスワードの入力を要求します。値 USER_AUTH は APP_PW とよく似ていますが、クライアントの初期化時にさらにユーザごとの認証が行われることも意味します。値 ACL は USER_AUTH とよく似ていますが、さらにサービス名、キュー名、およびイベント名に対してアクセス制御チェックが行われることを意味します。名前に対応する ACL が見つからなかった場合は、パーミッションが与えられているものとみなされます。値 MANDATORY_ACL は ACL とよく似ていますが、その名前に対応する ACL が見つからなかった場合にはパーミッションが与えられないことを意味します。

AUTHSVC *string_value*

システムに参加しているクライアントごとに、システムが呼び出すアプリケーション認証サービスの名前を指定します。このパラメータでは、SECURITY 識別子が USER_AUTH、ACL、または MANDATORY_ACL に設定されている必要があります。上位互換性のために、SECURITY APP_PW と AUTHSVC を両方セットすることは SECURITY USER_AUTH を意味します。パラメータ値の文字長は 15 文字以下です。SECURITY レベルが USER_AUTH の場合、デフォルトのサービス名は (指定しなかった場合は) AUTHSVC です。SECURITY レベルが ACL または MANDATORY_ACL の場合、デフォルトのサービス名は (指定しなかった場合は) ..AUTHSVC です。

システム提供の認証サーバ AUTHSVR は、SECURITY が USER_AUTH に設定されているときには認証サービスを AUTHSVC として宣言し、SECURITY が ACL または MANDATORY_ACL に設定されているときには ..AUTHSVC として宣言します。AUTHSVC と ..AUTHSVC は、同じ認証サービスを指します。

文字列値 AUTHSVC と ..AUTHSVC は識別子です。つまり、AUTHSVC または ..AUTHSVC を二重引用符で囲む必要はありません。

SCANUNIT *numeric_value*

サービス要求内で古いトランザクションやタイムアウト・ブロッキング呼び出しを見つけるために BBL が定期的なスキャンを行う間隔 (秒単位)。この値は BBL によるスキャン処理の基本単位として使用されます。この値は、tpbegin() で指定できるトランザクション・タイムアウト値と、BLOCKTIME パラメータで指定されるブロッキング・タイムアウト値に影響します。SANITYSCAN パラメータ、BBLQUERY パラメータ、DBBLWAIT パラメータ、および BLOCKTIME パラメータは、システム内のその他のタイムアウト値を指定するパラメータであり、SCANUNIT の倍数で指定されます。SCANUNIT は、5 の倍数で 0 より大きく 60 秒以下でなければなりません。デフォルト値は 10 秒です。

SANITYSCAN *numeric_value*

システムの正常性チェックを行う間隔を、基本 SCANUNIT の乗数で指定します。SCANUNIT の値は 0 より大きくなければなりません。このパラメータが指定されていない場合、デフォルト値は (SCANUNIT * SANITYSCAN) が約 120 秒になるように設定されます。正常性チェックは、掲示板のデータ構造体のほか、サーバに対しても実行されます。各 BBL は、そのマシン上のサーバがすべて実行可能であるかどうか、すなわち、サーバの異常終了やループが発生していないかどうかをチェックします。実行可能な状態でないと判断されたプロセスは、起動時に指定されたオプションに応じて、クリーンアップまたは再起動されます。それに引き続き、BBL は、メッセージ (応答なし) を DBBL に送信して、BBL が OK であることを示します。

DBBLWAIT *numeric_value*

DBBL がタイムアウトまでそのすべての BBL からの応答を待機する最大待ち時間を基本 SCANUNIT の乗数で設定します。DBBL は、BBL に要求を転送した後、すべての BBL から肯定応答を受信した後で要求元に応答を返します。このオプションは、動作不能の、または異常な BBL を適宜通知するために使用できます。DBBLWAIT は 0 より大きくなければなりません。このパラメータを指定しない場合、(SCANUNIT * DBBLWAIT) が SCANUNIT より大きくなるか、または 20 秒となるようにデフォルト値が設定されます。

BBLQUERY *numeric_value*

すべての BBL の DBBL により実行される状態チェックの間隔を基本 SCANUNIT の乗数で指定します。DBBL は、すべての BBL が BBLQUERY サイクル内で報告したことをチェックします。BBL からの報告がない

場合、DBBL はその BBL にメッセージを送信し、状態を照会します。応答がない場合、BBL は分断されます。BBLQUERY の値は 0 より大きくなければなりません。このパラメータの指定がない場合は、(SCANUNIT * BBLQUERY) が約 300 秒になるようにデフォルト値が設定されます。

BLOCKTIME *numeric_value*

ブロッキング呼び出し (応答の受信など) の後、タイムアウトするまでの時間を基本 SCANUNIT の乗数で指定します。BLOCKTIME の値は 0 より大きくなければなりません。このパラメータの指定がない場合は、(SCANUNIT * BLOCKTIME) が約 60 秒になるようにデフォルト値が設定されます。

NOTIFY {DIPIN | SIGNAL | THREAD | IGNORE}

クライアント・プロセスに送信される任意通知型メッセージのために、システムが使用するデフォルトの通知検出方法を指定します。このデフォルト値は、適切な tpinit() フラグ値を使用して、クライアントごとに変更できます。任意通知型メッセージが検出されると、tpsetunsol() 関数 (tpnotify()) で指定されたアプリケーション定義の任意通知型メッセージ処理ルーチンを使用して、アプリケーションからメッセージを使用できるようになります。

DIPIN を指定した場合、ディップ・イン方式の通知検知手段が使用されます。これは、システムが ATMI 呼び出しの中でクライアント・プロセスに代わって通知メッセージだけを検出することを意味します。特定の ATMI 呼び出し中の検出ポイントは、システムによっては定義されず、ブロック中のシステム呼び出しがディップ・イン検出によって中断されることはありません。DIPIN は、デフォルトの通知検出手段です。

SIGNAL を指定した場合、シグナル・ベースの通知検知手段が使用されます。これは、通知メッセージが使用可能になると、システムがターゲットのクライアント・プロセスにシグナルを送出することを意味します。システムは、通知手段を選択したクライアントに代わってシグナルを検出するルーチンをインストールします。

ネイティブ・クライアント・プロセスのすべてのシグナル処理は、アプリケーション・プロセスではなく管理システム・プロセスによって行われます。したがって、SIGNAL 方式を使用して通知できるのは、アプリケーション管理者と同じ UNIX システムのユーザ識別子で実行さ

れているネイティブ・クライアントだけです。ワークステーション・クライアントの場合、どのユーザ識別子で動作しているかに関係なく、SIGNAL 方式を使用できます。

注記　SIGNAL 通知方法は、MS-DOS クライアント、およびマルチスレッド・クライアントまたはマルチコンテキスト・クライアントに対しては使用できません。

THREAD を指定した場合、THREAD・通知検知手段が使用されます。この通知方法では、任意通知型メッセージを受け取るための専用のスレッドが使用され、そのスレッドに任意通知型メッセージ・ハンドラがディスパッチされます。1 つの BEA Tuxedo アプリケーション関連付けで一度に実行できる任意通知型メッセージ・ハンドラは 1 つだけです。この値は、マルチスレッド処理をサポートするプラットフォームでのみ使用できます。COBOL クライアントは、THREAD 通知を使用できません。COBOL で記述されているクライアントまたはスレッドをサポートしていないプラットフォーム上で実行されているクライアントでは、UBBCONFIG デフォルト通知方法を使用することができ、UBBCONFIG デフォルト通知方法が THREAD に設定されている場合、通知方法が DIPIN に変更されます。また、そのようなクライアントでは、tpinit() または TPINITIALIZE() へのパラメータで明示的にスレッド通知が指定されていると、この関数を呼び出したときにエラーが返されます。

IGNORE を指定した場合、デフォルトで通知メッセージがアプリケーション・クライアントに無視されます。これは、tpinit() 時の通知を要求するクライアントのみ任意通知型メッセージを受信するアプリケーションに適しています。

SEC_PRINCIPAL_NAME *string_value* [0..511]

BEA Tuxedo 7.1 以降のソフトウェアを実行するアプリケーションで認証に使用するためのセキュリティ・プリンシパル名の識別文字列を指定します。このパラメータには、最後のヌル文字を除いて 511 文字まで指定できます。このパラメータに指定するプリンシパル名は、このアプリケーションで実行される 1 つまたは複数のシステム・プロセスの識別子として使用されます。

SEC_PRINCIPAL_NAME は、コンフィギュレーション階層の 4 つのレベル (RESOURCES セクション、MACHINES セクション、GROUPS セクション、および SERVERS セクション) のいずれでも指定できます。特定の

コンフィギュレーション・レベルでのプリンシパル名は、それより下位レベルで変更可能です。どのレベルにも SEC_PRINCIPAL_NAME が指定されていない場合、アプリケーションのプリンシパル名のデフォルト値には、このアプリケーションの RESOURCES セクションに指定されている DOMAINID 文字列が設定されます。

SEC_PRINCIPAL_NAME のほかにも、SEC_PRINCIPAL_LOCATION と SEC_PRINCIPAL_PASSVAR というパラメータがあります。後の 2 つのパラメータは、アプリケーション起動時に、BEA Tuxedo 7.1 以降のアプリケーションで実行されるシステム・プロセス用の復号化キーを開く処理に関するものです。特定のレベルで SEC_PRINCIPAL_NAME だけが指定されている場合には、これ以外の 2 つのパラメータはそれぞれ、長さゼロの NULL 文字列に設定されます。

SEC_PRINCIPAL_LOCATION *string_value* [0..511]

SEC_PRINCIPAL_NAME で指定されたプリンシパルの復号化 (秘密) キーを収めるファイルまたはデバイスの場所を指定します。このパラメータには、最後のヌル文字を除いて 511 文字まで指定できます。

SEC_PRINCIPAL_LOCATION は、コンフィギュレーション階層の 4 つのレベル (RESOURCES セクション、MACHINES セクション、GROUPS セクション、および SERVERS セクション) のいずれでも指定できます。このパラメータは、どのレベルで指定する場合でも SEC_PRINCIPAL_NAME パラメータと対になっている必要があり、それ以外の場合には無視されます。SEC_PRINCIPAL_PASSVAR はオプションです。これが指定されていない場合、システムによって長さゼロの NULL 文字列が設定されます。

SEC_PRINCIPAL_PASSVAR *string_value* [0..511]

SEC_PRINCIPAL_NAME で指定されたプリンシパルのパスワードが格納される変数を指定します。このパラメータには、最後のヌル文字を除いて 511 文字まで指定できます。

SEC_PRINCIPAL_PASSVAR は、コンフィギュレーション階層の 4 つのレベル (RESOURCES セクション、MACHINES セクション、GROUPS セクション、および SERVERS セクション) のいずれでも指定できます。このパラメータは、どのレベルで指定する場合でも SEC_PRINCIPAL_NAME パラメータと対になっている必要があり、それ以外の場合には無視されます。SEC_PRINCIPAL_LOCATION はオプションです。これが指定され

ていない場合、システムによって長さゼロの NULL 文字列が設定されます。

初期化処理中、管理者は SEC_PRINCIPAL_PASSVAR で設定された各復号化キーのパスワードを入力する必要があります。パスワードの入力は、tmloadcf(1) で求められます。管理者が入力したパスはシステム側で自動的に暗号化され、暗号化されたそれぞれのパスワードは対応するパスワード変数に割り当てられます。

SIGNATURE_AHEAD *numeric_value* (1 <= *num* <= 2147483647)
ローカル・マシンの時刻から見て、その時刻からどれだけ先のデジタル署名のタイムスタンプが許容されるかを秒数で指定します。指定しない場合、デフォルトは 3600 秒 (1 時間) です。このパラメータは、BEA Tuxedo 7.1 以降のソフトウェアが実行されているアプリケーションにのみ適用されます。

SIGNATURE_BEHIND *numeric_value* (1 <= *num* <= 2147483647)
ローカル・マシンの時刻から見て、その時刻からどれだけ前のデジタル署名のタイムスタンプが許容されるかを秒数で指定します。指定しない場合、デフォルトは 604800 秒 (1 週間) です。このパラメータは、BEA Tuxedo 7.1 以降のソフトウェアが実行されているアプリケーションにのみ適用されます。

SIGNATURE_REQUIRED {Y | N}
このアプリケーションで実行するすべてのプロセスで、その入力メッセージ・バッファに対してデジタル署名が必要かどうかを指定します。指定しない場合、デフォルトは N です。このパラメータは、BEA Tuxedo 7.1 以降のソフトウェアが実行されているアプリケーションにのみ適用されます。

SIGNATURE_REQUIRED は、コンフィギュレーション階層の 4 つのレベル (RESOURCES セクション、MACHINES セクション、GROUPS セクション、および SERVICES セクション) のいずれでも指定できます。特定のレベルで SIGNATURE_REQUIRED を Y に設定すると、そのレベル以下で実行するすべてのプロセスで署名が必要となります。

ENCRYPTION_REQUIRED {Y | N}
このアプリケーションで実行するすべてのプロセスで、暗号化された入力メッセージ・バッファが必要かどうかを指定します。指定しない場合、デフォルトは N です。このパラメータは、BEA Tuxedo 7.1 以降

のソフトウェアが実行されているアプリケーションにのみ適用されます。

ENCRYPTION_REQUIRED は、コンフィギュレーション階層の 4 つのレベル (RESOURCES セクション、MACHINES セクション、GROUPS セクション、および SERVICES セクション) のいずれでも指定できます。特定のレベルで ENCRYPTION_REQUIRED を Y に設定すると、そのレベル以下で実行するすべてのプロセスで暗号化が必要となります。

MACHINES セクション

MACHINES セクションでは、物理マシンの論理名を指定します。また、このセクションではマシン固有のパラメータも指定します。MACHINES セクションには、アプリケーションで使用される物理プロセッサごとのエントリが必要です。エントリの形式は次のとおりです。

ADDRESS required_parameters [*optional_parameters*]

ADDRESS はプロセッサの物理名です。たとえば、UNIX システムの uname -n コマンドで生成される値です。Windows システムの場合、この値は [コントロールパネル] にあるネットワークのコンピュータ名の値で設定でき、大文字で指定する必要があります。ADDRESS のエントリの長さは 30 文字以下でなければなりません。この名前が識別子でない場合は、二重引用符で囲まなければなりません。

LAN オプションが指定されていない場合は、このセクションにはマシン名は 1 つしか指定できません。必須 *KEYWORD* の 1 つに LMID があります。これは、物理マシンに割り当てられる論理マシン *string_value* です。LMID *string_value* は、コンフィギュレーション・ファイルの MACHINES セクションの中で一意でなければなりません。

LMID = *string_value*

string_value を、別のセクションで *ADDRESS* のシンボル名として使用することを指定します。この名前にはカンマを指定できません。名前は 30 文字以内で指定します。このパラメータは必須です。コンフィギュレーションで使用されるすべてのマシンには、LMID 行を指定する必要があります。

必須パラメータは以下のとおりです。

TUXCONFIG = *string_value*[2..256] (BEA Tuxedo 8.0 以前では最大 64 バイト)

このマシン上のバイナリ形式の TUXCONFIG ファイルが存在するファイルまたはデバイスの絶対パス名。管理者が保守する必要があるのは、MASTER マシン上の TUXCONFIG 環境変数によって指定されている TUXCONFIG ファイルだけです。他のマシン上にあるこのマスタ TUXCONFIG ファイルのコピーは、システムの起動時に自動的に MASTER と同期されます。このパラメータは各マシンごとに指定しなければなりません。TUXOFFSET が指定されている場合、BEA Tuxedo ファイル・システムは、TUXCONFIG デバイスの最初からそのブロック数だけずれ

たところで起動します (後述の TUXOFFSET を参照)。この値がどのように使用されるかについては、MACHINES セクションの ENVFILE を参照してください。

注記　このパラメータに指定するパス名は、TUXCONFIG 環境変数に指定されているパス名と、大文字 / 小文字を含め正確に一致していなければなりません。そうでない場合、tmloadcf(1) は正常に実行されません。

TUXDIR = *string_value*[2..256] (BEA Tuxedo 8.0 以前では最大 78 バイト)
このマシン上の BEA Tuxedo システム・ソフトウェアが存在するディレクトリの絶対パス名。このパラメータはマシンごとに指定する必要があります。パス名は、各マシンにローカルなものでなければなりません。つまり、TUXDIR はリモート・ファイル・システムにあってはいけません。マルチプロセッサ・アプリケーションのマシンに異なるリリースの BEA Tuxedo システムがインストールされている場合、新しいリリースの 『BEA Tuxedo リリース・ノート』を参照して必要な機能を得るようにしてください。この値がどのように使用されるかについては、MACHINES セクションの ENVFILE を参照してください。

APPDIR = *string_value*[2..256] (BEA Tuxedo 8.0 以前では最大 78 バイト)
アプリケーション・ディレクトリの絶対パス名。この値はこのマシンで起動されるすべてのアプリケーションと管理サーバのカレント・ディレクトリです。この絶対パス名の後ろには、コロンで区切った他のパス名を追加できます。SECURITY が設定されているコンフィギュレーションでは、各アプリケーションは独自の APPDIR を持たなければなりません。この値がどのように使用されるかについては、MACHINES セクションの ENVFILE を参照してください。

オプション・パラメータは以下のとおりです。

UID = *number*
掲示板用に作成された IPC 構造体に関連付ける数値ユーザ ID を指定します。有効な値は 0 ～ 2147483647 です。値を指定しない場合、デフォルトで RESOURCES セクションの値が指定されます。

GID = *number*
掲示板用に作成された IPC 構造体に関連付ける数値グループ ID を指定します。有効な値は 0 ～ 2147483647 です。値を指定しない場合、デフォルトで RESOURCES セクションの値が指定されます。

PERM = *number*

掲示板をインプリメントする IPC 構造体に関連付ける数値パーミッションを指定します。このパラメータは、通常の UNIX システム形式 (0600 のような 8 進数値) で、プロセスに対する読み取りまたは書き込みパーミッションを指定するために使用します。値は 0001 以上 0777 以下の範囲で指定できます。値を指定しない場合、デフォルトで RESOURCES セクションの値が指定されます。

BRTHREADS = {Y | N}

このマシンのブリッジ・プロセスがマルチスレッド実行向け (Y) かシングル・スレッド実行向け (N) のどちらにコンフィギュレーションされているかを指定します。デフォルトは N です。このパラメータは、BEA Tuxedo 8.1 以降のソフトウェアが実行されているアプリケーションにのみ適用されます。

BRTHREADS を Y に設定することに意味があるのは、マシンに複数の CPU が存在する場合だけです。ただし、複数の CPU が存在することが BRTHREADS を Y に設定できることの前提条件というわけではありません。.

ローカル・マシンで BRTHREADS を Y に設定し、リモート・マシンで BRTHREADS を N に設定 (デフォルトを使用) することも可能ですが、マシン間のスループットはシングル・スレッドのブリッジ・プロセスのスループット以下になります。

シングル・スレッドまたはマルチスレッド実行用にコンフィギュレーションされたブリッジ・プロセスは、BEA Tuxedo または WebLogic Enterprise の旧リリース (BEA Tuxedo リリース 8.0 以前、WebLogic Enterprise release 5.1 以前) と相互運用できます。一般に、スレッド・ブリッジは非スレッド・ブリッジと相互運用できます。これは、スレッド化による機能および動作の変更が存在しないからです。

注記　BRTHREADS=Y で、ブリッジ環境に TMNOTHREADS=Y が含まれている場合、ブリッジはスレッド・モードで起動し、ブリッジが TMNOTHREADS 設定を無視することを示す警告メッセージをログに書き込みます。TMNOTHREADS 環境変数は、BEA Tuxedo リリース 8.0 に追加されました。

MAXACCESSERS = *number*

このマシンの掲示板に同時に接続できるクライアントおよびサーバの最大数を指定します。この値は 0 より大きく、32,768 未満でなければなりません。指定しない場合、デフォルトは RESOURCES セクションで指定される MAXACCESSERS です。

BBL、restartsrv、cleanupsrv、tmshutdown()、tmadmin() などのシステム管理プロセスはこの数に入れる必要はありません。ただし、DBBL、すべてのブリッジ・プロセス、すべてのシステム提供サーバ・プロセスとアプリケーション・サーバ・プロセス、およびこのサイトで使用する可能性があるクライアント・プロセスは数に入れる必要があります。システム提供のサーバには、AUTHSVR、TMQUEUE、TMQFORWARD、TMUSREVT、TMSYSEVT、TMS などがあります。GROUPS セクションの TMSNAME パラメータ、TMS_QM、GWTDOMAIN、および WSL を参照してください。このサイトでアプリケーションがワークステーション・リスナ (WSL) を起動する場合は、起動される WSL と使用する可能性があるワークステーション・ハンドラ (WSH) の両方をこの数に入れる必要があります。

BEA Tuxedo リリース 7.1 より前のリリース (6.5 以前) では、アプリケーションの MAXACCESSERS および MAXSERVERS (RESOURCES セクションの MAXSERVERS を参照) パラメータは、ユーザ・ライセンス数をチェックする仕組みで利用されます。つまり、アプリケーションで実行中の 1 台以上のマシンの MAXACCESSERS の数と、特定のマシンの MAXACCESSERS の数の合計が、MAXSERVERS の数とユーザ・ライセンス数の合計より大きい場合、マシンを起動することはできませんでした。したがって、アプリケーションの MAXACCESSERS パラメータには、MAXSERVERS の数とユーザ・ライセンス数の合計か、またはそれより小さい値を指定しなければなりませんでした。

また、BEA Tuxedo リリース 7.1 以降のユーザ・ライセンス数をチェックする仕組みで考慮されるのは、アプリケーションのユーザ・ライセンス数とそのアプリケーションで現在使用中のラインセンス数の 2 つだけです。すべてのユーザ・ライセンスが使用中になると、アプリケーションに新しいクライアントが参加することはできなくなります。

MAXWSCLIENTS = *number*

(ネイティブ・クライアントに対して) ワークステーション・クライアント用に確保されるこのマシンへのアクセサ・エントリの数を指定します。指定する場合、値は 0 以上 32,768 未満でなければなりません。指定しない場合、デフォルトは 0 です。

ここで指定する値は、MAXACCESSERS で指定されたアクセサ・スロットの総数の一部です。つまり、MAXWSCLIENTS 用に確保されるアクセサ・スロットは、このマシン上の別のクライアントおよびサーバでは使用できません。この値を MAXACCESSERS より大きい値に設定した場合、エラーになります。

MAXWSCLIENTS パラメータが使用されるのは、BEA Tuxedo システムの Workstation 機能を使用するときだけです。システムへのワークステーション・クライアントのアクセスは、BEA Tuxedo システム提供の代替物、つまりワークステーション・ハンドラ (WSH) を通じて多重化されるため、このパラメータを適切な値に設定することでプロセス間通信 (IPC) リソースを節約できるようになります。

MAXACLCACHE = *number*

SECURITY が ACL または MANDATORY_ACL に設定されているときに ACL エントリのために使用されるキャッシュ内のエントリ数を指定します。このパラメータを適切に設定すると、共有メモリ上のリソースを節約しながら、ACL のチェックを行うためのディスクのアクセス回数を減らすことができます。この値は、10 以上 32,000 以下でなければなりません。デフォルトは 100 です。

MAXCONV = *number*

このマシン上のクライアントおよびサーバが同時に関与できる会話の最大数を指定します。この値は 0 より大きく、32,768 未満でなければなりません。指定しない場合、デフォルトは RESOURCES セクションで指定される MAXCONV の値です。サーバ当たりの同時会話の最大数は 64 です。

MAXPENDINGBYTES = *number*

プロセスによって送信されるのを待っているメッセージのために割り当てられスペースの量の限界を指定します。*number* は、100,000 から MAXLONG までの値でなければなりません。

MAXGTT = *number*

このマシンが同時に関与できるグローバル・トランザクションの最大数を指定します。この値は 0 より大きく、32,768 未満でなければなりません。指定しない場合、デフォルトは RESOURCES セクションで指定される値です。

TYPE = *string_value*

マシンをクラスにグループ分けするときに使用します。TYPE には、15 文字以下の任意の文字列を設定できます。2 つのマシンが同じ TYPE 値を持つ場合、それらのマシン間でデータを送信するときにデータの符号化および復号化は無視されます。TYPE には、任意の文字列値を指定できます。このパラメータは、比較のために使用します。TYPE パラメータは、アプリケーションが異機種ネットワークのマシンで構成されている場合、またはネットワーク内のマシン上でさまざまなコンパイラを使用している場合に使用します。このパラメータを指定しない場合、デフォルトは値が指定されていない他のすべてのエントリに一致するヌル文字列です。

CMPLIMIT = *string_value1*[,*string_value2*]

自動データ圧縮が実行されるリモート・プロセス (*string_value1*) およびローカル・プロセス (*string_value2*) に向けたメッセージのサイズのしきい値を指定します。どちらの値も、負以外の整数または文字列 MAXLONG でなければなりません。指定しない場合、デフォルトは MAXLONG です。

NETLOAD = *numeric_value*

このマシンから別のマシンにサービス要求を送信するコストを計算するときに追加する負荷を指定します。この値は、0 以上 32,768 未満でなければなりません。指定しない場合のデフォルトは 0 です。

SPINCOUNT = *numeric_value*

UNIX セマフォ上のプロセスをブロッキングする、ユーザ・レベルでの掲示板ロックの回数を指定します。この値は 0 以上でなければなりません。この値が 0 の場合は、配布されたバイナリに組み込まれているスピンカウントを使用しなければならないことを示します。この値を設定すると、TMSPINCOUNT 環境変数は無視されます。これは、プラットフォームによって異なります。このパラメータのデフォルト値は 0 です。

TLOGDEVICE = *string_value*[0..256] (BEA Tuxedo 8.0 以前では最大 64 バイト)

このマシンの DTP トランザクション・ログ (TLOG) を格納する BEA Tuxedo ファイル・システムを指定します。TLOG は、BEA Tuxedo システムの VTOC テーブルとしてデバイスに格納されています。このパラメータを指定しない場合、そのマシンには TLOG がないものと見なされます。

TLOGOFFSET = *offset*

このマシンの DTP トランザクション・ログを格納する BEA Tuxedo ファイル・システムの開始点までの (デバイス先頭からの) 数値オフセットをページ単位で指定します。このオフセットは 0 以上で、デバイス上のページ数より小さい値でなければなりません。デフォルト値は 0 です。

TLOGNAME = *string_value*

このマシンの DTP トランザクション・ログ名を指定します。指定しない場合、デフォルトは TLOG です。1 つの TLOGDEVICE に複数の TLOG がある場合、各 TLOG の名前は一意でなければなりません。TLOGNAME は、TLOG テーブルが作成されるコンフィギュレーション内の他のいかなるテーブルの名前とも異なっていなければなりません。名前は 30 文字以内でなければなりません。

TLOGSIZE = *size*

このマシンの DTP トランザクション・ログの数値サイズをページ単位で指定します。この値は、BEA Tuxedo ファイル・システム上の使用可能な容量に応じて、0 より大きく 2048 以下にする必要があります。指定しない場合、デフォルトは 100 ページです。

ULOGPFX = *string_value*[0..256] (BEA Tuxedo 8.0 以前では最大 78 バイト)

このマシンの userlog(3c) メッセージ・ファイルの絶対パス名の接頭辞を指定します。指定したマシンの ULOGPFX の値を使用して、そのマシン上で実行されるすべてのサーバ、クライアント、管理プロセスについての userlog(3c) エラー・メッセージ・ファイルを作成します。このパラメータの指定がない場合は、$APPDIR/ULOG が使用されます。"mmddyy" (月、日、年) をこの接頭辞に付加して、実際のログ・ファイル名を得ることができます。

TUXOFFSET = *offset*

このマシンの TUXCONFIG ファイルを格納する BEA Tuxedo ファイル・システムの開始点までの (デバイス先頭からの) 数値オフセットをページ単位で指定します。このオフセットは 0 以上で、デバイス上のページ数より小さい値でなければなりません。デフォルトのオフセットは 0 です。0 以外の値を指定した場合、TUXOFFSET の値はマシン上で起動するすべてのサーバの環境に置かれます。この値がどのように使用されるかについては、MACHINES セクションの ENVFILE を参照してください。

ENVFILE = *string_value*[0..256] (BEA Tuxedo 8.0 以前では最大 78 バイト)

マシン上のすべてのクライアントとサーバを指定されたファイルの環境で実行することを指定します。無効なファイル名が指定されると、環境に値は追加されません。環境ファイルの各行は、ident=*value* の形式で指定します。*ident* の先頭にはアンダースコア (_) または英文字を指定し、全体はアンダースコアまたは英数字のみで構成します。*value* 内で、${*env*} という形式の文字列は、ファイルの処理時に環境内の既存の変数を使用して展開されます (前方参照はサポートされていません。値が設定されていない場合、変数は空の文字列に置換されます)。バックスラッシュ (\) を使用して、ドル記号およびそれ自体をエスケープすることができます。その他すべてのシェルのクォーテーションとエスケープのメカニズムは無視され、展開された *value* がそのまま環境に入れられます。

クライアント・プログラムは、tpinit() の実行時に MACHINES ENVFILE のみを処理します。

サーバを起動する際、ローカル・サーバは tmboot(1) の環境を継承し、リモート・サーバ (MASTER) は tlisten(1) の環境を継承します。関連する MACHINES エントリの情報に基づいてサーバが起動するときは、TUXCONFIG、TUXDIR、および APPDIR も環境に組み込まれます。これら 3 つの変数をほかの値に変更しようとすると、警告メッセージが表示されます。tmboot および tlisten は、サーバを起動する前にマシンの ENVFILE を処理するため、環境には実行可能ファイルと動的にロードされるファイルの検索に必要なパス名が示されます。サーバが実行されると、(アプリケーションが tpsvrinit() で制御権を得る前の) サーバ初期化の一部として、マシンおよびサーバの ENVFILE ファイルから変数を読み取り、エクスポートします。マシンとサーバの両

方の ENVFILE で変数が設定されている場合、サーバの ENVFILE の値がマシンの ENVFILE の値に優先します。

PATH と LD_LIBRARY_PATH は、特殊な方法で処理されます。サーバがアクティブになる前に、マシンの ENVFILE がスキャンされ、最初に現れた PATH 変数または LD_LIBRARY_PATH 変数が検出されます。どちらの PATH 変数内でも、埋め込み型の環境変数は展開されません。PATH と LD_LIBRARY_PATH は、実行可能ファイルと動的にロードされるファイルのパス名を検索するために使用されます。PATH には常に次の接頭辞が付きます。

```
${APPDIR}:${TUXDIR}/bin:/bin:
```

ただし、既にこの接頭辞がある場合は付きません。この PATH は、単純なパス名または相対パス名で指定されたサーバの検索パスとして使用されます。LD_LIBRARY_PATH には常に次の接頭辞が付きます。

```
${APPDIR}:${TUXDIR}/lib:/lib:/usr/lib:
```

ただし、既にこの接頭辞がある場合は付きません。HPUX では SHLIB_PATH が設定され、AIX では LD_LIBRARY_PATH の代わりに LIBPATH が設定されます。

SEC_PRINCIPAL_NAME = *string_value* [0..511]

BEA Tuxedo 7.1 以降のソフトウェアを実行するアプリケーションで認証に使用するためのセキュリティ・プリンシパル名の識別文字列を指定します。このパラメータには、最後のヌル文字を除いて 511 文字まで指定できます。このパラメータに指定するプリンシパル名は、このマシンで実行される 1 つまたは複数のシステム・プロセスの識別子として使用されます。

SEC_PRINCIPAL_NAME は、コンフィギュレーション階層の 4 つのレベル (RESOURCES セクション、MACHINES セクション、GROUPS セクション、および SERVERS セクション) のいずれでも指定できます。特定のコンフィギュレーション・レベルでのプリンシパル名は、それより下位レベルで変更可能です。どのレベルにも SEC_PRINCIPAL_NAME が指定されていない場合、アプリケーションのプリンシパル名のデフォルト値には、このアプリケーションの RESOURCES セクションに指定されている DOMAINID 文字列が設定されます。

SEC_PRINCIPAL_NAME のほかにも、SEC_PRINCIPAL_LOCATION と SEC_PRINCIPAL_PASSVAR というパラメータがあります。後の 2 つのパラメータは、アプリケーション起動時に、BEA Tuxedo 7.1 以降のアプリケーションで実行されるシステム・プロセス用の復号化キーを開く処理に関するものです。特定のレベルで SEC_PRINCIPAL_NAME だけが指定されている場合には、これ以外の 2 つのパラメータはそれぞれ、長さゼロの NULL 文字列に設定されます。

SEC_PRINCIPAL_LOCATION = *string_value* [0..511]

SEC_PRINCIPAL_NAME で指定されたプリンシパルの復号化 (秘密) キーを収めるファイルまたはデバイスの場所を指定します。このパラメータには、最後のヌル文字を除いて 511 文字まで指定できます。

SEC_PRINCIPAL_LOCATION は、コンフィギュレーション階層の 4 つのレベル (RESOURCES セクション、MACHINES セクション、GROUPS セクション、および SERVERS セクション) のいずれでも指定できます。このパラメータは、どのレベルで指定する場合でも SEC_PRINCIPAL_NAME パラメータと対になっている必要があり、それ以外の場合には無視されます。SEC_PRINCIPAL_PASSVAR はオプションです。これが指定されていない場合、システムによって長さゼロの NULL 文字列が設定されます。

SEC_PRINCIPAL_PASSVAR = *string_value* [0..511]

SEC_PRINCIPAL_NAME で指定されたプリンシパルのパスワードが格納される変数を指定しますこのパラメータには、最後のヌル文字を除いて 511 文字まで指定できます。

SEC_PRINCIPAL_PASSVAR は、コンフィギュレーション階層の 4 つのレベル (RESOURCES セクション、MACHINES セクション、GROUPS セクション、および SERVERS セクション) のいずれでも指定できます。このパラメータは、どのレベルで指定する場合でも SEC_PRINCIPAL_NAME パラメータと対になっている必要があり、それ以外の場合には無視されます。SEC_PRINCIPAL_LOCATION はオプションです。これが指定されていない場合、システムによって長さゼロの NULL 文字列が設定されます。

初期化処理中、管理者は SEC_PRINCIPAL_PASSVAR で設定された各復号化キーのパスワードを入力する必要があります。パスワードの入力は、tmloadcf(1) で求められます。管理者が入力したパスはシステム

側で自動的に暗号化され、暗号化されたそれぞれのパスワードは対応するパスワード変数に割り当てられます。

SIGNATURE_REQUIRED = {Y | N}
: このマシンで実行するすべてのプロセスで、その入力メッセージ・バッファに対してデジタル署名が必要かどうかを指定します。指定しない場合、デフォルトは N です。このパラメータは、BEA Tuxedo 7.1 以降のソフトウェアが実行されているアプリケーションにのみ適用されます。

  SIGNATURE_REQUIRED は、コンフィギュレーション階層の 4 つのレベル (RESOURCES セクション、MACHINES セクション、GROUPS セクション、および SERVICES セクション) のいずれでも指定できます。特定のレベルで SIGNATURE_REQUIRED を Y に設定すると、そのレベル以下で実行するすべてのプロセスで署名が必要となります。

ENCRYPTION_REQUIRED = {Y | N}
: このマシンで実行するすべてのプロセスで、暗号化された入力メッセージ・バッファが必要かどうかを指定します。指定しない場合、デフォルトは N です。このパラメータは、BEA Tuxedo 7.1 以降のソフトウェアが実行されているアプリケーションにのみ適用されます。

  ENCRYPTION_REQUIRED は、コンフィギュレーション階層の 4 つのレベル (RESOURCES セクション、MACHINES セクション、GROUPS セクション、および SERVICES セクション) のいずれでも指定できます。特定のレベルで ENCRYPTION_REQUIRED を Y に設定すると、そのレベル以下で実行するすべてのプロセスで暗号化が必要となります。

SICACHEENTRIESMAX = *string_value*
: このマシン上で各プロセスが保持するサービス・キャッシュ・エントリの最大数を表します。この値は 0 以上、32,768 未満でなければなりません。この値を指定しない場合、デフォルトは 500 です。値を 0 に設定した場合は、このマシン上のどのプロセスもサービス・キャッシュを実行しません。この属性が受け付ける最大値は 32,767 です。このマシン上のすべてのクライアントがこの値を使用します。

注記　SERVERS セクションで対応する属性とは異なり、このパラメータでは文字列 DEFAULT を有効値として使用できません。

GROUPS セクション

このセクションでは、サーバ・グループに関する情報を指定します。このセクションには、最低 1 つのサーバ・グループが定義されている必要があります。サーバ・グループは、TUXCONFIG ファイルの作成後に tmconfig、wtmconfig(1) を使用して追加できます。サーバ・グループのエントリには、サーバ群およびマシン上のサービス群に対して、論理名を指定します。論理名は、SERVERS セクションの SRVGRP パラメータの値に使用されます。この値により、サーバはグループ内のサーバとして識別されます。SRVGRP は、SERVICES セクションで、グループ内の特定のサービス・インスタンスのオカレンスを識別する場合にも使用されます。GROUPS セクションのその他のパラメータは、このグループを特定のリソース管理インスタンスに関連付けます (社員データベースなど)。GROUPS セクション内にある各行の形式は次のとおりです。

GROUPNAME required_parameters [*optional_parameters*]

GROUPNAME は、グループの論理名 (*string_value*) です。グループ名は、GROUPS セクションのグループ名と MACHINES セクションの LMID の中で一意でなければなりません。グループ名の中にアスタリスク (*)、カンマ、コロンを入れることはできません。グループ前は 30 文字以内でなければなりません。

以下のパラメータは必須です。

LMID = *string_value1* [,*string_value2*]
: このサーバ・グループが MACHINES セクション (または SHM モードのデフォルト値) の *string_value1* によってシンボル名に指定されているマシンにあることを示します。LMID 値は、それぞれ 30 文字以下でなければなりません。論理マシン名は 2 つまで指定できます。2 つ目の論理名を指定する場合で、サーバ・グループが移行できる場合は、この 2 つ目の論理名は、サーバ・グループが移行できるマシンを示します。

GRPNO = *number*
: このサーバ・グループに関連する数値グループ番号を指定します。0 より大きく、30000 未満の番号を指定します。番号は、GROUPS セクションのすべてのエントリの中で一意でなければなりません。

オプション・パラメータは以下のとおりです。

TMSNAME = *string_value*[0..256] (BEA Tuxedo 8.0 以前では最大 78 バイト)
このグループに関連するトランザクション・マネージャ・サーバ a.out の名前を指定します。分散トランザクションに参加するサーバを持つグループに対しては、必ずこのパラメータを指定する必要があります。分散トランザクションとは、tpbegin() で開始し tpcommit()/ tpabort() で終了する、複数のリソース・マネージャ (場合によっては複数のマシン) 間で処理されるトランザクションのことです。このパラメータは、サーバ・グループをブートするときに tmboot(1) により実行されるファイル (*string_value*) を指定します。値 TMS は、非 XA インターフェイスを使用することを示すために予約されています。TMS 以外の空でない値が指定された場合は、このエントリの LMID 値に関連するマシンに対して、TLOGDEVICE を指定する必要があります。各 TM サーバに対して一意のサーバ識別子が自動的に選択され、サーバは何回でも再起動することができます。

ENVFILE = *string_value*[0..256] (BEA Tuxedo 8.0 以前では最大 78 バイト)
グループ内のすべてのサーバを指定されたファイルの環境で実行することを指定します。無効なファイル名が指定されると、環境に値は追加されません。環境ファイルの各行は、ident=*value* の形式で指定します。ident はアンダースコア (_) または英数字で構成します。*value* 内の ${*env*} という形式の文字列は、ファイルの処理時に、環境内の既存の変数を使用して展開されます。前方参照はサポートされていません。値が設定されていない場合、変数は空の文字列に置換されます。バックスラッシュ (\) を使用して、ドル記号とバックスラッシュをエスケープできます。他のすべてのシェルのクォートとエスケープは無視され、展開された *value* がそのまま環境に入れられます。

ENVFILE は、MACHINES セクションの ENVFILE (存在する場合) の後、SERVERS セクションの ENVFILE (指定されている場合) の前に読み取られます。

TMSCOUNT = *number*
TMSNAME が指定されている場合に、関連するグループに対して起動するトランザクション・マネージャ・サーバ数を指定します。このパラメータは省略可能で、デフォルトは 3 です。0 以外の値を指定した場合、最小値は 2 で、最大値は 10 です。サーバは MSSQ セットに自動的にセットアップされます。

SEC_PRINCIPAL_NAME = *string_value* [0..511]

BEA Tuxedo 7.1 以降のソフトウェアを実行するアプリケーションで認証に使用するためのセキュリティ・プリンシパル名の識別文字列を指定します。このパラメータには、最後のヌル文字を除いて 511 文字まで指定できます。このパラメータに指定するプリンシパル名は、このグループで実行される 1 つまたは複数のシステム・プロセスの識別子として使用されます。

SEC_PRINCIPAL_NAME は、コンフィギュレーション階層の 4 つのレベル (RESOURCES セクション、MACHINES セクション、GROUPS セクション、および SERVERS セクション) のいずれでも指定できます。特定のコンフィギュレーション・レベルでのプリンシパル名は、それより下位レベルで変更可能です。どのレベルにも SEC_PRINCIPAL_NAME が指定されていない場合、アプリケーションのプリンシパル名のデフォルト値には、このアプリケーションの RESOURCES セクションに指定されている DOMAINID 文字列が設定されます。

SEC_PRINCIPAL_NAME のほかにも、SEC_PRINCIPAL_LOCATION と SEC_PRINCIPAL_PASSVAR というパラメータがあります。後の 2 つのパラメータは、アプリケーション起動時に、BEA Tuxedo 7.1 以降のアプリケーションで実行されるシステム・プロセス用の復号化キーを開く処理に関するものです。特定のレベルで SEC_PRINCIPAL_NAME だけが指定されている場合には、これ以外の 2 つのパラメータはそれぞれ、長さゼロの NULL 文字列に設定されます。

SEC_PRINCIPAL_LOCATION = *string_value* [0..511]

SEC_PRINCIPAL_NAME で指定されたプリンシパルの復号化 (秘密) キーを収めるファイルまたはデバイスの場所を指定します。このパラメータには、最後のヌル文字を除いて 511 文字まで指定できます。

SEC_PRINCIPAL_LOCATION は、コンフィギュレーション階層の 4 つのレベル (RESOURCES セクション、MACHINES セクション、GROUPS セクション、および SERVERS セクション) のいずれでも指定できます。このパラメータは、どのレベルで指定する場合でも SEC_PRINCIPAL_NAME パラメータと対になっている必要があり、それ以外の場合には無視されます。SEC_PRINCIPAL_PASSVAR はオプションです。これが指定されていない場合、システムによって長さゼロの NULL 文字列が設定されます。

SEC_PRINCIPAL_PASSVAR = *string_value* [0..511]

SEC_PRINCIPAL_NAME で指定されたプリンシパルのパスワードが格納される変数を指定します。このパラメータには、最後のヌル文字を除いて 511 文字まで指定できます。

SEC_PRINCIPAL_PASSVAR は、コンフィギュレーション階層の 4 つのレベル (RESOURCES セクション、MACHINES セクション、GROUPS セクション、および SERVERS セクション) のいずれでも指定できます。このパラメータは、どのレベルで指定する場合でも SEC_PRINCIPAL_NAME パラメータと対になっている必要があり、それ以外の場合には無視されます。SEC_PRINCIPAL_LOCATION はオプションです。これが指定されていない場合、システムによって長さゼロの NULL 文字列が設定されます。

初期化処理中、管理者は SEC_PRINCIPAL_PASSVAR で設定された各復号化キーのパスワードを入力する必要があります。(パスワードの入力は、tmloadcf(1) で求められます。管理者が入力したパスはシステム側で自動的に暗号化され、暗号化されたそれぞれのパスワードは対応するパスワード変数に割り当てられます。

SIGNATURE_REQUIRED = {Y | N}

このグループで実行するすべてのプロセスで、その入力メッセージ・バッファに対してデジタル署名が必要かどうかを指定します。指定しない場合、デフォルトは N です。このパラメータは、BEA Tuxedo 7.1 以降のソフトウェアが実行されているアプリケーションにのみ適用されます。

SIGNATURE_REQUIRED は、コンフィギュレーション階層の 4 つのレベル (RESOURCES セクション、MACHINES セクション、GROUPS セクション、および SERVICES セクション) のいずれでも指定できます。特定のレベルで SIGNATURE_REQUIRED を Y に設定すると、そのレベル以下で実行するすべてのプロセスで署名が必要となります。

ENCRYPTION_REQUIRED = {Y | N}

このグループで実行するすべてのプロセスで、暗号化された入力メッセージ・バッファが必要かどうかを指定します。指定しない場合、デフォルトは N です。このパラメータは、BEA Tuxedo 7.1 以降のソフトウェアが実行されているアプリケーションにのみ適用されます。

ENCRYPTION_REQUIRED は、コンフィギュレーション階層の 4 つのレベル (RESOURCES セクション、MACHINES セクション、GROUPS セクション、および SERVICES セクション) のいずれでも指定できます。特定のレベルで ENCRYPTION_REQUIRED を Y に設定すると、そのレベル以下で実行するすべてのプロセスで暗号化が必要となります。

OPENINFO = *string_value*

このグループのリソース・マネージャをオープンするときに必要な、リソース・マネージャに依存する情報を指定します。この値は二重引用符で囲む必要があり、その長さは 256 文字以下でなければなりません。

このグループの TMSNAME パラメータが設定されていないか、または TMS に設定されている場合、この値は無視されます。TMSNAME パラメータが TMS 以外の値に設定されており、OPENINFO がヌル文字列 ("") が設定されているか、または何も設定されていない場合は、グループのリソース・マネージャは存在しますが、open 操作の実行に関する情報は必要ありません。

OPENINFO 文字列の形式は、基となるリソース・マネージャのベンダごとに異なります。ベンダ固有の情報の先頭には、トランザクション・インターフェイス (XA) インターフェイスの公開名とコロン (:) が付きます。

BEA Tuxedo /Q データベースでは、次のような形式になります。

# On UNIX #
OPENINFO = "TUXEDO/QM:*qmconfig*:*qspace*"

# On Windows #
OPENINFO = "TUXEDO/QM:*qmconfig*;*qspace*"

# In AS/400 environment #
OPENINFO = "TUXEDO/QM:*qmconfig*;*qspace*"

# In OpenVMS environment #
OPENINFO = "TUXEDO/QM,[a.b.c]*qmconfig*,*qspace*"

TUXEDO/QM は BEA Tuxedo /Q XA インターフェイスの公開名、*qmconfig* はキュー・スペースがある QMCONFIG (qmadmin(1) を参照) の名前、*qspace* はキュー・スペースの名前に相当します。Windows

および AS/400 では、*qmconfig* の後ろの区切りはセミコロン (;) でなければなりません。OpenVMS では、TUXEDO/QM の後ろと *qmconfig* の後ろの区切りはカンマ (,) でなければなりません。

その他のベンダのデータベースでは、OPENINFO 文字列の形式はリソース・マネージャを提供するベンダの指定によって異なります。たとえば、次の OPENINFO 文字列は、Oracle のリソース・マネージャをオープンするときに必要な情報を示します。

```
OPENINFO="Oracle_XA:Oracle_XA+Acc=P/Scott/*****+SesTm=30+Log
Dit=/tmp"
```

Oracle_XA は、Oracle XA インターフェイスの公開名です。OPENINFO 文字列内の 5 つの連続するアスタリスク (*) は、暗号化されたパスワードを示します。次に、パスワードについて説明します。

リソース・マネージャに渡される OPENINFO 文字列内のパスワードは、平文または暗号化された形式で格納されます。パスワードを暗号化するには、まず、OPENINFO 文字列内のパスワードが必要な場所に、5 つ以上の連続するアスタリスクを入れます。次に、tmloadcf(1) を実行して UBBCONFIG ファイルをロードします。tmloadcf() は、アスタリスクの文字列を検出すると、パスワードの作成をユーザに要求します。次に例を示します。

```
tmloadcf -y /usr5/apps/bankapp/myubbconfig
Password for OPENINFO (SRVGRP=BANKB3):
password
```

tmloadcf(1) は、パスワードを暗号化して TUXCONFIG ファイルに格納します。tmunloadcf() を使用して TUXCONFIG から UBBCONFIG ファイルを再生成すると、パスワードは @@ で区切られた暗号化形式でその UBBCONFIG ファイルに出力されます。次に例を示します。

```
OPENINFO="Oracle_XA:Oracle_XA+Acc=P/Scott/@@A0986F7733D4@@+S
esTm=30+LogDit=/tmp"
```

tmloadcf() の実行時に、tmunloadcf() によって生成された UBBCONFIG ファイル内で暗号化されたパスワードが検出されても、ユーザに対してパスワードの作成は要求されません。

CLOSEINFO = *string_value*

このグループのリソース・マネージャをクローズするときに必要な、リソース・マネージャに依存する情報を指定します。この値は二重引用符で囲む必要があり、その長さは 256 文字以下でなければなりません。CLOSEINFO 文字列は、BEA Tuxedo /Q データベースには使用されません。

このグループの TMSNAME パラメータが設定されていないか、または TMS に設定されている場合、この値は無視されます。TMSNAME パラメータが TMS 以外の値に設定されており、CLOSEINFO がヌル文字列 ("") に設定されているか、または何も設定されていない場合は、グループのリソース・マネージャは存在しますが、close 操作の実行に関する情報は必要ありません。

CLOSEINFO 文字列の形式は、基となるリソース・マネージャのベンダごとに異なります。ベンダ固有の情報の先頭には、トランザクション・インターフェイス (XA) インターフェイスの公開名とコロン (:) が付きます。

NETGROUPS セクション

NETGROUPS セクションでは、LAN 環境で使用可能なネットワーク・グループに関する情報を指定します。複数のネットワーク・グループに、複数のマシンのペアが存在できます。通信を行う 2 つのノードは、優先度メカニズムを使用して、そのグループのエレメント間での通信方法を決定します。

すべての LMID は、デフォルトのネットワーク・グループ (DEFAULTNET) のメンバでなければなりません。リリース 6.4 (NETGROUPS が使用可能) 以前の BEA Tuxedo リリースを実行するマシンは、DEFAULTNET ネットワーク・グループにのみ属することができます。DEFAULTNET のネットワーク・グループ番号 (NETGRPNO) は 0 (ゼロ) で、変更できません。ただし、DEFAULTNET のデフォルト優先順位は変更できます。

このセクションのエントリの一般的な形式は次のとおりです。

NETGROUP required_parameters [*optional_parameters*]

NETGROUP はネットワーク・グループ名です。*NETGROUP* が DEFAULTNET と同じである場合、エントリはデフォルトのネットワーク・グループを表します。

以下のパラメータは必須です。

NETGRPNO = *numeric_value*
: これは、一意のネットワーク・グループ番号です。この番号は、フェイルオーバーおよびフェイルバックで使用できるように管理者が割り当てる必要があります。このエントリが DEFAULTNET の場合には、数値を 0 (ゼロ) にする必要があります。

オプション・パラメータは以下のとおりです。

NETPRIO = *numeric_value*
: このネットワーク・グループの優先順位を指定します。同じ優先順位の複数のネットワーク・グループの一組のマシンは、より優先度の高いネットワーク・グループが使用可能でない限り、その優先順位バンドで並列通信を行います。ある優先バンドのすべてのネットワーク・リンクが管理者またはネットワーク状態によって分断されている場合、次の優先順位のバンドが使用されます。優先度の高いバンドから再試行されます。詳細については、『BEA Tuxedo アプリケーションの設定』を参照してください。この値は、0 以上 8,192 未満でなければ

なりません。指定しない場合、デフォルトは 100 です。これは、DEFAULTNET の唯一の変更可能なパラメータです。

注記　並列データ回線は、優先グループ番号中のネットワーク・グループ番号 (NETGRPNO) によって優先付けされます。

NETWORK セクション

NETWORK では、LAN 環境用のネットワーク・コンフィギュレーションを指定します。BRIDGE サーバが位置している各プロセッサごとに、NETWORK セクションにエントリを入れて、BRIDGE プロセスのネットワーク・アドレスを指定しなければなりません。このセクションが存在し、RESOURCES セクションの OPTIONS パラメータに LAN が指定されていない場合、エラーが発生します。

このセクションのエントリの一般的な形式は次のとおりです。

LMID required_parameters [*optional_parameters*]

LMID は、BRIDGE プロセスが存在する論理マシンです。*LMID* には、使用されるネットワーク・デバイスへの直接アクセス権が必要です (BRIDGE パラメータで指定される)。

以下のパラメータは必須です。

NADDR = *string_value*[0..256] (BEA Tuxedo 8.0 以前では最大 78 バイト)

この *LMID* の BRIDGE プロセスが使用する完全なネットワーク接続指示受け付けアドレスを指定します。BRIDGE の接続指示受け付けアドレスは、アプリケーションに参加している他の BRIDGE プロセスがこの BRIDGE プロセスと通信するための手段として使用されます。*string_value* の形式が "0x*hex-digits*" または "\\x*hex-digits*" の場合、偶数の有効な 16 進数桁を含める必要があります。。これらの形式は、TCP/IP アドレスを含む文字配列に内部変換され、以下の 2 つの形式のいずれかになります。

```
"//hostname:port_number"
"//#.#.#.#:port_number"
```

最初の形式では、オペレーティング・システム・コマンドでアクセスされたローカル設定の名前解決機能を使ってアドレスがバインドされるときに、*hostname* は TCP/IP ホスト・アドレスに解決されます。"*#.#.#.#*" はドットで区切った 10 進数の形式で、*#* は 0 から 255 までの 10 進数の値を表します。*Port_number* は 0 から 65535 までの 10 進数で、指定された文字列の 16 進表現です。

注記　一部のポート番号は、お使いのシステムで使用されるトランスポート・プロトコル (TCP/IP など) のために予約されている場合

があります。予約されているポート番号を確認するには、トランスポート・プロトコルのマニュアルを調べてください。

オプション・パラメータは以下のとおりです。

BRIDGE = *string_value*

LMID の BRIDGE プロセスがネットワークにアクセスするために使用するデバイス名を指定します。この値は、TLI ベースの BEA Tuxedo システム・バイナリを介してネットワーク・アプリケーションに参加する場合に必要です。このパラメータは、ソケット・ベースの BEA Tuxedo システム・バイナリでは不要です。

NLSADDR = *string_value*[0..256] (BEA Tuxedo 8.0 以前では最大 78 バイト)

このパラメータは、*LMID* でネットワークにサービスを提供する tlisten(1) プロセスが使用するネットワーク・アドレスを指定します。NLSADDR 用のネットワーク・アドレスは、上記 NADDR パラメータ用に指定されたのと同じ形式です。アドレスの形式が "0x*hex-digits*" または "\\x*hex-digits*" の場合、偶数の有効な 16 進数桁を含める必要があります。TCP/IP アドレスは "//*#.#.#.#:port*" という形式にできます。NLSADDR が警告を表示する MASTER LMID 以外のエントリにない場合、tmloadcf(1) はエラーを表示します。ただし、NLSADDR が MASTER LMID にない場合、tmadmin(1) はリモート・マシン上で管理モードでは実行できず、読み取り専用操作に限定されます。これはまた、バックアップ・サイトが障害後にマスタ・サイトを再起動できないことを意味します。

FADDR = *string_value*[0..256] (BEA Tuxedo 8.0 以前では最大 78 バイト)

ほかのマシンに接続する際にローカル・マシンが使用するネットワーク・アドレスを指定します。このパラメータは、FRANGE パラメータと共に、アウトバウンド接続を確立する前にプロセスがバインドを試みる TCP/IP ポートの範囲を決定します。このアドレスには、TCP/IP アドレスを指定する必要があります。TCP/IP アドレスのポート部分は、プロセスが TCP/IP ポートをバインドできる範囲のベース・アドレスを表します。FRANGE パラメータは、範囲の大きさを指定します。たとえば、このアドレスが //mymachine.bea.com:30000 で FRANGE が 200 の場合、この LMID からアウトバウンド接続の確立を試みるすべてのネイティブ・プロセスは、mymachine.bea.com 上のポートを 30000 から 30200 の間でバインドします。このパラメータを設定しない場合、

デフォルトは空文字列になり、オペレーティング・システムがローカル・ポートを任意に選択します。

FRANGE = *number*

アウトバウンド接続を確立する前にネイティブ・プロセスがバインドを試みる TCP/IP ポートの範囲を指定します。FADDR パラメータは、範囲のベース・アドレスを指定します。たとえば、FADDR パラメータが //mymachine.bea.com:30000 で、FRANGE が 200 に設定されている場合、この LMID からアウトバウンド接続の確立を試みるすべてのネイティブ・プロセスは、mymachine.bea.com 上のポートを 30000 から 30200 の間でバインドします。有効範囲は 1 から 65535 までです。デフォルトは 1 です。

MINENCRYPTBITS = {0 | 40 | 56 | 128}

このマシンへのネットワーク・リンクを確立するときに必要な暗号化の最低レベルを指定します。0 は暗号化を行わないことを示し、40、56、および 128 は暗号化キーの長さをビット単位で指定します。ここで指定する最小レベルの暗号化が満たされない場合、リンクの確立は失敗します。デフォルトは 0 です。

注記　リンク・レベルの暗号化の値 40 は、下位互換性を維持するために提供されています。

MAXENCRYPTBITS = {0 | 40 | 56 | 128}

ネットワーク・リンクを確立するときに使用できる暗号化の最高レベルを指定します。0 は暗号化を行わないことを示し、40、56、および 128 は暗号化キーの長さをビット単位で指定します。デフォルトは 128 です。

注記　リンク・レベルの暗号化の値 40 は、下位互換性を維持するために提供されています。

NETGROUP = *string_value*

string_value は、このネットワーク・エントリに関連付けるネットワーク・グループです。指定しない場合、デフォルトは DEFAULTNET です。NETGROUP パラメータを DEFAULTNET に設定しない場合は、ファイルの NETGROUPS セクションにあるグループ名として前に指定された値を指定する必要があります。NETGROUP パラメータが DEFAULTNET に設定されているネットワーク・エントリは、すべて TM_MIB の T_MACHINE クラスで表されますが、ほかの NETGROUP に関連付けられ

た NETWORK エントリは、TM_MIB の T_NETMAP クラスで表され、旧リリースとの相互運用が可能です。

SERVERS セクション

このセクションでは、システムで起動されるサーバの初期状態に関する情報を定義します。特定のリモート環境に対しては、サーバを、連続的に動作していて、プロセスへのサーバ・グループのサービス要求を待機しているプロセスであるとする考え方を適用できる場合とできない場合があります。多くの環境では、オペレーティング・システムあるいはリモート・ゲートウェイが、サービスの唯一のディスパッチャになります。どちらの場合でも、SERVICE テーブル・エントリ (次のセクションを参照) のみと、SERVER テーブル・エントリなしをリモート・プログラムのエントリ・ポイントに対して指定する必要があります。BEA Tuxedo システム・ゲートウェイ・サーバは、リモート・ドメイン・サービス要求を宣言し、キューに入れます。ホスト固有のリファレンス・ページでは、UBBCONFIG のサーバ・テーブル・エントリが特定の環境に適応しているかどうかを示し、適応している場合は対応するセマンティクスを明記する必要があります。SERVERS セクション内にある各行の形式は次のとおりです。

AOUT required_parameters [*optional_parameters*]

AOUT は、tmboot(1) によって実行されるファイル (*string_value*) を指定します。tmboot は、サーバが属するサーバ・グループ用に指定されたマシン上で *AOUT* を実行します。tmboot はそのターゲット・マシン上で *AOUT* ファイルを検索します。したがって、*AOUT* がそのマシンのファイル・システムに存在していなければなりません (もちろん、*AOUT* へのパスには、別のマシンのファイル・システムへの RFS 接続を含めることができます)。サーバの相対パス名が指定されると、まずディレクトリ APPDIR、次にディレクトリ TUXDIR/bin、/bin、*path* (*path* はマシン環境ファイル上の始めの PATH= 行) の順に *AOUT* が検索されます。APPDIR および TUXDIR の値は、TUXCONFIG ファイル内の適切なマシン・エントリから取得されます。詳細については、MACHINES セクションの ENVFILE を参照してください。

BEA Tuxedo 8.1 以降では、SERVERS セクションの *AOUT* の長さは最大 256 バイトです。BEA Tuxedo 8.0 以前では、SERVERS セクションの *AOUT* の長さは最大 78 バイトです。

以下のパラメータは必須です。

SRVGRP = *string_value*

サーバが動作するグループのグループ名を指定します。*string_value* は、GROUPS セクション内のサーバ・グループに関連する論理名で、30 文字以下でなければなりません。GROUPS セクション内のエントリと関連するということは、そのサーバに対して LMID が指定されているマシン上で *AOUT* が実行されるということを意味します。また、このパラメータは、サーバ・グループの GRPNO と、関連するリソース・マネージャがオープンするときに渡されるパラメータも指定します。すべてのサーバ・エントリには、サーバ・グループのパラメータが指定されていなければなりません。

SRVID = *number*

グループ内でサーバを一意に識別する整数を指定します。識別子は、1 以上 30,000 以下でなければなりません。このパラメータはすべてのサーバ・エントリに必要です。

オプション・パラメータは、ブート・オプションと実行時オプションという 2 つのカテゴリに分けられます。ブート・オプションは、サーバの実行時に tmboot(1)tmboot が使用します。いったん実行されると、サーバはコンフィギュレーション・ファイルからエントリを読み込み、ランタイム・オプションを決定します。一意のサーバ ID を使用して正しいエントリを見つけることができます。

オプションのブート・パラメータは以下のとおりです。

CLOPT = *string_value*

起動時に *AOUT* に渡される servopts(5) オプションを指定します。何も指定しない場合、デフォルトは -A です。*string_value* の長さは最大 256 バイトです。

SEQUENCE = *number*

このサーバを、他のサーバに関連していつ起動またはシャットダウンするかを指定します。SEQUENCE を指定しない場合、サーバは SERVERS セクションで指定された順序で起動します (シャットダウンはその逆の順序で行われます)。シーケンス番号が指定されているサーバと指定されていないサーバが混在する場合は、まず、シーケンス番号を持つサーバがすべて昇順に起動し、次に、シーケンス番号を持たないサーバがコンフィギュレーション・ファイル内の順序ですべて起動します。シーケンス番号は 1 から 9999 までの範囲で指定しなければなりません。

MIN = *number*

tmboot によって起動するサーバのオカレンスの最小数を指定します。RQADDR が指定されており、MIN が 1 より大きい場合、そのサーバは MSSQ セットを形成します。サーバ識別子は、SRVID + MAX - 1 までの SRVID となります。各サーバには、すべて同一のシーケンス番号とその他のサーバ・パラメータが付けられます。MIN には 0 ～ 1000 の範囲の値を指定できます。デフォルトは 1 です。

MAX = *number*

起動できるサーバのオカレンスの最大数を指定します。tmboot が実行されると、MIN で指定した数のサーバが起動します。次に、tmboot の -i オプションを使用して関連するサーバ識別子を指定し、その他のサーバ (MAX で指定した数まで) を起動します。MAX 値の範囲は MIN から 1000 までです。このパラメータの指定がない場合、デフォルトは MIN と同じになります。

オプションの実行時パラメータは以下のとおりです。

ENVFILE = *string_value*[0..256] (BEA Tuxedo 8.0 以前では最大 78 バイト)

初期化の間にサーバ環境のファイルにある値の追加を要求します。サーバが、別のマシンに移行可能なサーバ・グループに関連付けられている場合は、移行元マシンと移行先マシンの同じ場所に ENVFILE を格納する必要があります。

このファイルはサーバが起動した後に処理されます。したがって、サーバ実行に必要な実行可能ファイルまたは動的にロードするファイルを検索するためにパス名を設定することはできません。代わりにマシン ENVFILE を使用してください。このファイルを使用して環境を変更する方法については、MACHINES セクションの ENVFILE を参照してください。

CONV = {Y | N}

サーバが会話型サーバであるかどうかを指定します。接続は会話型サーバとの間でのみ確立でき、tpacall() または tpcall() を使用した rpc 要求は非会話型サーバに対してのみ行うことができます。デフォルトは N です。

RQADDR = *string_value*

AOUT の要求キューのシンボル名を指定します。シンボル名は 30 文字以内でなければなりません。このパラメータの指定がない場合は、

AOUT がアクセスするキューのために固有のキー (GRPNO.SRVID) を選択します。複数のサーバに対して同じ RQADDR と実行可能ファイル名を指定すると、複数サーバ / 単一キュー (MSSQ) セットを定義できます。2 つのサーバに、同じキュー名を持つ RQADDR が指定されている場合、それらは同じサーバ・グループになければなりません。

RQPERM = *number*

要求キューに対する数値パーミッションを指定します。*number* は通常の UNIX システムの方式 (たとえば 0600) で指定されます。RQPERM を指定しない場合、PERM が RESOURCES セクションに指定されていればその値が使用されます。それ以外の場合は、値 0666 が使用されます。値は 0001 以上 0777 以下の範囲で指定できます。

REPLYQ = {Y | N}

AOUT に対して応答キューを確立するかどうかを指定します。Y を指定した場合は、応答キューが *AOUT* と同じ LMID 上に作成されます。デフォルトは N です。MSSQ セットのサーバの場合、応答を受けようとするサーバの REPLYQ は Y に設定します。

注記　会話型サーバに対する REPLYQ の値は、UBBCONFIG の中で割り当てられている値に関係なく常に Y に設定されます。

RPPERM = *number*

応答キューに対する数値パーミッションを指定します。number は通常の UNIX システムの方式 (たとえば 0600) で指定されます。RPPERM を指定しない場合、デフォルトは 0666 です。要求と応答が両方とも同じキューから読み取られる場合、指定する必要があるのは RQPERM だけで、RPPERM は無視されます。値は 0001 以上 0777 以下の範囲で指定できます。

RCMD = *string_value*[0..256] (up to 78 bytes for BEA Tuxedo 8.0 or earlier)

AOUT が再起動できる場合、このパラメータは、*AOUT* が異常終了した場合に実行するコマンドを指定します。最初のスペースまたはタブまでの文字列は、絶対パス名または APPDIR を基準とする相対パス名で指定された実行可能な UNIX ファイル名でなければなりません (コマンドの先頭でシェル変数を設定しないでください)。このコマンドには、コマンド行引き数をオプションで指定することもできます。コマンド行には、GRPNO と SRVID という、サーバの再起動に関連する 2 つ

の引き数が追加されます。string_value はサーバの再起動と並行して実行されます。

MAXGEN = *number*

AOUT が再起動できる場合、このパラメータは、GRACE によって指定された時間内に最大 *number* - 1 回再起動できることを示します。指定できる値は 0 より大きく、256 より小さい数値です。この値を指定しないと、デフォルトの 1 (サーバは一度起動できるが、再起動はできない) が設定されます。

GRACE = *number*

AOUT が再起動できる場合、このパラメータは指定秒数内で最大 MAXGEN 回再起動できることを示します。この値は、0 以上 2,147,483,648 未満でなければなりません。0 の場合、*AOUT* は何回でも再起動できます。GRACE を指定しない場合は、デフォルトの 86,400 秒 (24 時間) が指定されます。

RESTART = {Y | N}

AOUT が再起動できるかどうかを指定します。デフォルトは N です。サーバを移行できる場合は、RESTART を Y に設定します。SIGTERM シグナルで終了したサーバは、再起動できないためリブートする必要があることに注意してください。

SYSTEM_ACCESS = *identifier*[,*identifier*]

アプリケーション・プロセス内で BEA Tuxedo システム・ライブラリが BEA Tuxedo システムの内部テーブルへのアクセス権を取得するために使用するデフォルトのモードを指定します。有効なアクセス・タイプは、FASTPATH または PROTECTED です。FASTPATH を指定した場合、ライブラリは共用メモリを介して迅速に内部テーブルへアクセスできます。PROTECTED を指定した場合、BEA Tuxedo システム・ライブラリが共用メモリを介して内部テーブルにアクセスできる間、BEA Tuxedo システム・ライブラリの外部からはそれらのテーブルの共用メモリにアクセスできません。NO_OVERRIDE を単独あるいは FASTPATH または PROTECTED と共に指定した場合、アプリケーション・プロセスによって選択モードを変更することができません。SYSTEM_ACCESS を指定しない場合、デフォルト・モードは RESOURCES セクションの SYSTEM_ACCESS キーワードの設定値によって決まります。

制限事項 : SYSTEM_ACCESS を PROTECTED に設定しても、マルチスレッド・サーバには効果がない場合があります。これは、あるスレッドが BEA Tuxedo コードを実行しているときに (つまりスレッドが掲示板にアタッチされているとき) 別のスレッドがユーザ・コードを実行できるからです。BEA Tuxedo では、このような状況を防止することはできません。

MAXDISPATCHTHREADS = *number*

個々のサーバ・プロセスで生成可能な、同時にディスパッチされるスレッドの最大数を指定します。このパラメータは、サーバが buildserver -t コマンドを使用して構築された場合にのみ有効です。

MAXDISPATCHTHREADS > 1 の場合、別のディスパッチ・スレッドが使用されます。このディスパッチ・スレッドは、パラメータで指定した数には含まれません。MINDISPATCHTHREADS <= MAXDISPATCHTHREADS でなければなりません。このパラメータを指定しない場合、デフォルトは 1 です。

MINDISPATCHTHREADS = *number*

最初のサーバの起動時に開始されるサーバ・ディスパッチ・スレッドの数。このパラメータは、サーバが buildserver -t コマンドを使用して構築された場合にのみ有効です。

MAXDISPATCHTHREADS > 1 のときに使用される各ディスパッチ・スレッドは、MINDISPATCHTHREADS で指定した数には含まれません。MINDISPATCHTHREADS <= MAXDISPATCHTHREADS でなければなりません。このパラメータのデフォルト値は 0 です。

THREADSTACKSIZE = *number*

マルチスレッド・サーバの各ディスパッチ・スレッドに対して作成されるスタックのサイズを指定します。この値は 0 以上 2,147,483,647 以下でなければなりません。デフォルトは 0 です。このパラメータがサーバに対して有効なのは、MAXDISPATCHTHREADS に 1 より大きい値が指定されている場合のみです。

このパラメータが指定されていない場合、または 0 が指定されている場合、デフォルトのスレッド・サイズが使用されます。デフォルト・サイズは、オペレーティング・システムのデフォルト・サイズです。ただし、この値がマルチスレッド BEA Tuxedo アプリケーション用に十分であることが分かっている場合は、BEA Tuxedo のデフォルト・

サイズが使用されます。現在、BEA Tuxedo のデフォルト・スレッド・スタック・サイズは 1,024,000 です。

スレッド・スタック・サイズを上回った場合、サーバはコア・ダンプを行います。

SEC_PRINCIPAL_NAME = *string_value* [0..511]

BEA Tuxedo 7.1 以降のソフトウェアを実行するアプリケーションで認証に使用するためのセキュリティ・プリンシパル名の識別文字列を指定します。このパラメータには、最後のヌル文字を除いて 511 文字まで指定できます。このパラメータに指定するプリンシパル名は、このサーバで実行される 1 つまたは複数のシステム・プロセスの識別子として使用されます。

SEC_PRINCIPAL_NAME は、コンフィギュレーション階層の 4 つのレベル (RESOURCES セクション、MACHINES セクション、GROUPS セクション、および SERVERS セクション) のいずれでも指定できます。特定のコンフィギュレーション・レベルでのプリンシパル名は、それより下位レベルで変更可能です。どのレベルにも SEC_PRINCIPAL_NAME が指定されていない場合、アプリケーションのプリンシパル名のデフォルト値には、このアプリケーションの RESOURCES セクションに指定されている DOMAINID 文字列が設定されます。

SEC_PRINCIPAL_NAME のほかにも、SEC_PRINCIPAL_LOCATION と SEC_PRINCIPAL_PASSVAR というパラメータがあります。後の 2 つのパラメータは、アプリケーション起動時に、BEA Tuxedo 7.1 以降のアプリケーションで実行されるシステム・プロセス用の復号化キーを開く処理に関するものです。特定のレベルで SEC_PRINCIPAL_NAME だけが指定されている場合には、これ以外の 2 つのパラメータはそれぞれ、長さゼロの NULL 文字列に設定されます。

SEC_PRINCIPAL_LOCATION = *string_value* [0..511]

SEC_PRINCIPAL_NAME で指定されたプリンシパルの復号化 (秘密) キーを収めるファイルまたはデバイスの場所を指定します。このパラメータには、最後のヌル文字を除いて 511 文字まで指定できます。

SEC_PRINCIPAL_LOCATION は、コンフィギュレーション階層の 4 つのレベル (RESOURCES セクション、MACHINES セクション、GROUPS セクション、および SERVERS セクション) のいずれでも指定できます。このパラメータは、どのレベルで指定する場合でも

SEC_PRINCIPAL_NAME パラメータと対になっている必要があり、それ以外の場合には無視されます。(SEC_PRINCIPAL_PASSVAR はオプションです。これが指定されていない場合、システムによって長さゼロの NULL 文字列が設定されます。

SEC_PRINCIPAL_PASSVAR = *string_value* [0..511]

SEC_PRINCIPAL_NAME で指定されたプリンシパルのパスワードが格納される変数を指定します。このパラメータには、最後のヌル文字を除いて 511 文字まで指定できます。

SEC_PRINCIPAL_PASSVAR は、コンフィギュレーション階層の 4 つのレベル (RESOURCES セクション、MACHINES セクション、GROUPS セクション、および SERVERS セクション) のいずれでも指定できます。このパラメータは、どのレベルで指定する場合でも SEC_PRINCIPAL_NAME パラメータと対になっている必要があり、それ以外の場合には無視されます。(SEC_PRINCIPAL_LOCATION はオプションです。これが指定されていない場合、システムによって長さゼロの NULL 文字列が設定されます。

初期化処理中、管理者は SEC_PRINCIPAL_PASSVAR で設定された各復号化キーのパスワードを入力する必要があります。パスワードの入力は、tmloadcf(1) で求められます。管理者が入力したパスはシステム側で自動的に暗号化され、暗号化されたそれぞれのパスワードは対応するパスワード変数に割り当てられます。

SICACHEENTRIESMAX = *string_value*

文字列が数字だけで構成されている場合、その数字はこのサーバで保持できるサービス・キャッシュ・エントリの最大数を指定します。このパラメータには、0 以上 32,768 未満の値を指定します。それ以外の場合、文字列の値として DEFAULT が使用できます。この場合、キャッシュできるサービスの数は、このサーバに対応する MACHINE セクションのエントリによって指定されます。値を指定しない場合、文字列 DEFAULT が有効値として使用されます。0 を指定した場合、このマシン上のプロセスでサービスはキャッシュされません。このパラメータの最大値は 32,767 です。

CONCURR_STRATEGY=PER_REQUEST
CONCURR_STRATEGY = PER_OBJECT

CONCURR_STRATEGY は、マルチスレッド CORBA サーバ・アプリケーションによって使用されるスレッド・モデルを指定するために使用

します。CONCURR_STRATEGY パラメータには、次のいずれかの値を指定できます。

```
CONCURR_STRATEGY = PER_REQUEST
```

```
CONCURR_STRATEGY = PER_OBJECT
```

CONCURR_STRATEGY = PER_REQUEST を指定して要求単位のスレッド・モデルを使用する場合、CORBA サーバ・アプリケーションでの各呼び出しは、スレッド・プール内の任意のスレッドに割り当てられます。

CONCURR_STRATEGY = PER_OBJECT を指定してオブジェクト単位のスレッド・モデルを使用する場合、アクティブな各オブジェクトはいつでも単一のスレッドに関連付けられます。オブジェクトの各要求によって、ディスパッチ・スレッドとそのオブジェクトの間に関連付けが行われます。

SERVICES セクション

このセクションでは、アプリケーションが使用するサービスに関する情報を指定します。SERVICES セクション内の行の形式は次のとおりです。

SVCNM [*optional_parameters*]

SVCNM はサービスの (*string_value*) 名です。*SVCNM* は 15 文字以下でなければなりません。

必須パラメータはありません。オプション・パラメータを設定する必要ない場合は、サービスをリストする必要はありません。オプション・パラメータは以下のとおりです。

LOAD = *number*

SVCNM がシステムに *number* のロード・ファクタを負わせることを示します。*number* には、1 以上 32,767 以下の値を指定できます。指定しない場合、デフォルトは 50 です。数値が大きくなるほどロード・ファクタも大きくなります。

PRIO = *number*

SVCNM をキューから取り出す優先順位の数値を指定します。この値は 0 より大きく、100 以下でなければなりません (100 が最高の優先度)。デフォルト値は 50 です。
メッセージは、FIFO に基づき 10 回に 1 回取り出されるため、優先順位の低いメッセージがキューに無制限にとどまることはありません。優先度の低いインターフェイスやサービスでは、応答時間を問題にすべきではありません。

SRVGRP = *string_value*

指定されたすべてのパラメータをサーバ・グループ *string_value* 内の *SVCNM* に適用することを指定します。SRVGRP を使用すると、同じサービスが異なるサーバ・グループ内では異なるパラメータ設定を持つことが可能になります。このパラメータの長さは 30 文字以内でなければなりません。

BUFTYPE = "*type1*[:*subtype1*[,*subtype2* . . .]][;*type2*[:*subtype3*[, . . .]]] . . . "

このサービスで受け付けるデータ・バッファのタイプおよびサブタイプのリスト。このパラメータの長さは 256 文字までです。また、最大 32 のタイプ / サブタイプの組み合わせを指定できます。BEA Tuxedo

システムに用意されているデータ・バッファのタイプには、FML と FML32 (FML バッファ用)、XML (XML バッファ用)、VIEW、VIEW32、X_C_TYPE、または X_COMMON (FML VIEW 用)、STRING (NULL で終了する文字配列用)、および CARRAY または X_OCTET (送信時に符号化も復号化もされない文字配列用) があります。これらのタイプのうち、VIEW、VIEW32、X_C_TYPE、および X_COMMON にはサブタイプがあります。VIEW タイプは、サービスが期待する特定の VIEW の名前を指定します。アプリケーションのタイプとサブタイプも追加できます (tuxtypes(5) を参照)。サブタイプを持つ TYPE では、サブタイプに "*" を指定して、該当サービスが関連するタイプのすべてのサブタイプを受け付けるようにできます。

1 つのサービスが解釈できるバッファ・タイプは一定のものに限られています。すなわち、そのバッファ・タイプ・スイッチにあるものしか解釈できません (tuxtypes(5) を参照)。BUFTYPE パラメータが ALL に設定されている場合、そのサービスはそのバッファ・タイプ・スイッチにあるバッファ・タイプをすべて受け付けます。BUFTYPE パラメータを省略することと、このパラメータを ALL に設定することは同じです。サービス名は同じで異なる SRVGRP パラメータを持つ複数のエントリがある場合、BUFTYPE パラメータはそのエントリすべてにおいて同じでなければなりません。

タイプ名は 8 文字以下、サブタイプ名は 16 文字以下で指定することができます。タイプ名とサブタイプ名は、セミコロン、コロン、カンマ、アスタリスクを含んではいけません (タイプ値とサブタイプ値がどこで終わるのか分かりにくくなります)。

次に、有効な BUFTYPE 指定の例を示します。

```
BUFTYPE=FML は、サービスが FML バッファを取ることを示します。
BUFTYPE=VIEW:* は、サービスが FML VIEW のすべてのサブタイプを取ることを示します。
```

ROUTING = *string_value*

データ依存型ルーティングを行うときに、このサービスに使用されるルーティング基準名を指定します。*string_value* は ROUTING セクションに定義される *ROUTING_CRITERIA_NAME* で、このサービスに対するデータ依存型ルーティングのルーティング基準の名前です。このパラメータを指定しない場合、このサービスではデータ依存型ルーティ

ングが行われません。*string_value* は 15 文字以内でなければなりません。サービス名は同じで異なる SRVGRP パラメータを持つ複数のエントリがある場合、ROUTING パラメータはそのエントリすべてにおいて同じでなければなりません。

SVCTIMEOUT = *number*

特定のサービスの処理に与える時間を秒単位で指定します。この値は 0 以上でなければなりません。この値が 0 の場合、サービスはタイムアウトになりません。サービスがタイムアウトになると、サービス要求を処理しているサーバが SIGKILL シグナルで終了します。このシグナルは、サーバ内のすべてのスレッドに影響することに注意してください。このパラメータのデフォルト値は 0 です。

SIGNATURE_REQUIRED = {Y | N}

このサービスのすべてのインスタンスで、その入力メッセージ・バッファにデジタル署名が必要かどうかを指定します。指定しない場合、デフォルトは N です。このパラメータは、BEA Tuxedo 7.1 以降のソフトウェアが実行されているアプリケーションにのみ適用されます。

SIGNATURE_REQUIRED は、コンフィギュレーション階層の 4 つのレベル (RESOURCES セクション、MACHINES セクション、GROUPS セクション、および SERVICES セクション) のいずれでも指定できます。特定のレベルで SIGNATURE_REQUIRED を Y に設定すると、そのレベル以下で実行するすべてのプロセスで署名が必要となります。

ENCRYPTION_REQUIRED = {Y | N}

このサービスのすべてのインスタンスで、暗号化された入力メッセージ・バッファが必要かどうかを指定します。指定しない場合、デフォルトは N です。このパラメータは、BEA Tuxedo 7.1 以降のソフトウェアが実行されているアプリケーションにのみ適用されます。

ENCRYPTION_REQUIRED は、コンフィギュレーション階層の 4 つのレベル (RESOURCES セクション、MACHINES セクション、GROUPS セクション、および SERVICES セクション) のいずれでも指定できます。特定のレベルで ENCRYPTION_REQUIRED を Y に設定すると、そのレベル以下で実行するすべてのプロセスで暗号化が必要となります。

以下のパラメータは、DTP アプリケーション専用です。

AUTOTRAN = {Y | N}
: まだトランザクション・モードでない状態で要求のメッセージが取り出された場合、トランザクションが自動的に開始されるかどうかを指定します。デフォルトは N です。

TRANTIME = *number*
: 関連付けられているサービス用に自動的に開始されるトランザクションのデフォルトのタイムアウト値を秒単位で指定します。この値は、0 以上 2,147,483,648 未満でなければなりません。0 を指定すると、マシンの最大タイムアウト値が設定されます。

INTERFACES セクション

このセクションでは、アプリケーションで使用する CORBA インターフェイスに対するアプリケーション全体のデフォルト・パラメータを定義するための情報を指定します。ファクトリ・ベースのルーティング (特定のサーバ・グループに処理を分散する機能) を実行しない場合には、CORBA インターフェイスに必要なパラメータはありません。ファクトリ・ベースのルーティングを実行する場合は、以下のパラメータを指定する必要があります。

表 64 ファクトリ・ベース・ルーティングのパラメータ

セクション名	指定する値
INTERFACES	■ 使用するインタフェースの名前 ■ システム側で各インターフェイスに適用するルーティング基準の名前
ROUTING	ルーティング基準
GROUPS	サーバ・グループの名前

ファクトリ・ベース・ルーティングおよび関連パラメータの詳細については、664 ページの「ROUTING セクション」を参照してください。

指定するパラメータがない場合は、CORBA インターフェイスを記述する必要はありません。

指定できるオプション・パラメータは以下のとおりです。

`AUTOTRAN = {Y | N}`

操作が呼び出されるたびにトランザクションを自動的に開始し、呼び出しから復帰したときに自動的に終了するかどうかを指定します。`AUTOTRAN` パラメータは、トランザクション・ポリシーが `optional` のインターフェイスにのみ適用されます。それ以外の場合、このパラメータは無視されます。デフォルトは `N` です。

トランザクション・ポリシーは、インプリメンテーション・コンフィギュレーション・ファイルで指定されます。このトランザクション・ポリシーが、実行時に、関連する `T_IFQUEUE MIB` オブジェクトのトランザクション・ポリシーの属性になります。

AUTOTRAN 値を設定する前に、システム管理者はプログラマがインターフェイスに割り当てたトランザクション・ポリシーを知っておく必要があります。そうでないと、実行時に AUTOTRAN が期待どおりに機能しない可能性があります。

AUTOTRAN を Y に設定した場合、TRANTIME パラメータも設定する必要があります。

FACTORYROUTING = *criteria_name*
このインターフェイスのオブジェクト・リファレンスを作成するときにルーティング基準を使用する場合に必要です。ルーティング基準は、UBBCONFIG ファイルの ROUTING セクションで指定します。

LOAD = *number*
CORBA インターフェイスによってシステムが受けると考えられる相対的な負荷を表す 1 ～ 100 の範囲内の任意の数字。この値は、このアプリケーションで使用されるほかの CORBA インターフェイスに割り当てられた LOAD 値との関係で相対的に決まります。デフォルトは 50 です。CORBA 環境では、LOAD の値を使用して、要求をキューに登録するのに最適なマシンが選択されます。ルーティング先のサーバの負荷は、要求された CORBA インターフェイスのロード・ファクタ分 (LOAD) だけ増加します。

PRIO = *number*
CORBA インターフェイスのすべてのメソッドすべてに対して、キューから取り出されるときの優先順位を指定します。この値には、1 ～ 100 の値を指定します。100 は、優先順位が最も高いことを示します。デフォルト値は 50 です。

SRVGRP = *server-group-name*
INTERFACES セクションのこの部分で定義するすべてのパラメータが、指定されたサーバ・グループ内のインターフェイスに適用されることを示します。この方法を使用すると、特定の CORBA インターフェイスに対して、サーバ・グループごとに異なるパラメータ値を定義できます。

TRANTIME = *number*
処理されるトランザクションのタイムアウト値 (秒数)。AUTOTRAN が Y に設定されている場合、TRANTIME パラメータを設定する必要があります。この値は 0 ～ 2147483647 (2^{31} - 1)、つまり約 68 年でなければな

りません。0 は、トランザクションにタイムアウトが設定されていないことを示します。デフォルトは 30 秒です。

`TIMEOUT = *number*`

この CORBA インターフェイスのメソッドの処理にかかる時間を秒単位で指定します。この値は 0 以上でなければなりません。0 を指定した場合は、インターフェイスではタイムアウトが発生しません。タイムアウト・メソッドにより、インターフェイスに対してメソッドを処理するサーバが `SIGKILL` イベントと共に終了します。実行に最も時間がかかるメソッドに合わせてタイムアウト値を指定することをお勧めします。

ROUTING セクション

このセクションでは、FML バッファ、XML バッファ、および VIEW を使用するサービス要求のデータ依存型ルーティングに関する情報を指定します。ここで指定するルーティング基準は、デフォルトのルーティング関数 _froute、_xroute、および _vroute が使用される場合にのみ使用されます (tuxtypes(5) を参照)。ROUTING セクション内の行の形式は次のとおりです。

ROUTING_CRITERIA_NAME required_parameters

ROUTING_CRITERIA_NAME は、SERVICES セクションの特定のサービス・エントリの ROUTING パラメータに割り当てられる (*string_value*) 名です。*ROUTING_CRITERIA_NAME* は 15 文字以下でなければなりません。

以下のパラメータは必須です。

FIELD = *string_value*

ルーティング・フィールドの名前を指定します。指定する文字数は 30 文字以内でなければなりません。このフィールドは、FML または FML32 バッファ、XML の要素または属性、FML フィールド・テーブルで指定される VIEW フィールド名 (2 つの環境変数 —FLDTBLDIR および FIELDTBLS または FLDTBLDIR32 および FIELDTBLS32 を使用)、または FML テーブル (2 つの環境変数 —VIEWDIR および VIEWFILES または VIEWDIR32 および VIEWFILES32 を使用) であるとそれぞれ見なされます。この情報は、メッセージの送信時に、データ依存型ルーティングに関連するフィールド値を取得するために使用されます。FML または FML32 バッファ内のフィールドがルーティングに使用される場合は、フィールドの値は 8,191 以下でなければなりません。

XML 文書を要素の内容または属性に基づいてルーティングするには、次の構文で FIELD パラメータを定義する必要があります。

FIELD="*root_element*[/*child_element*][/*child_element*][/. . .][/@*attribute_name*]"

FIELD の値には、ルーティングの要素または要素の属性名を指定します。要素は、XML 文書またはデータグラムの要素のタイプ (要素名) または要素の属性名と見なされます。この情報は、ドキュメントまたはデータグラム送信時に、データ依存型ルーティングで要素の内容または属性を識別するために使用されます。要素名と属性名を組み合わせて、最大 30 文字まで指定できます。インデックスはサポートされ

ないので、BEA Tuxedo システムは、データ依存型ルーティングで XML バッファを処理する際に、指定された要素タイプの最初のオカレンスだけを認識します。

XML は、属性名に使用できる文字セットを厳密に定義しています。属性名は、単一の文字、アンダースコア (_)、またはコロン (:) を含む文字列で、その後に 1 つ以上の名前文字が続きます。要素名と属性名はいずれも、大文字小文字が区別されます。

XML の詳細については、World Wide Web Consortium の Web サイト http://www.w3c.org/XML を参照してください。

FIELDTYPE = *type*

FIELD パラメータに指定されたルーティング・フィールドのタイプを指定します。このパラメータは、XML バッファをルーティングする場合にのみ使用されます。値 *type* は、CHAR、SHORT、LONG、FLOAT、DOUBLE、STRING のいずれかに設定できます。ルーティング・フィールドのデフォルトのタイプは STRING です。

RANGES = *string_value*

ルーティング・フィールドの範囲と関連するサーバ・グループを指定します。*string* は二重引用符で囲む必要があります。*string* で使用できる文字数は最大 2,048 文字です。ただし、Domains の場合は *string* は最大 4,096 文字です。この文字列は、範囲 /group_name の組み合わせのリストをカンマで区切った形式をとります。たとえば、RANGES="0-2:DBG1,3-5:DBG2,6-9:"DBG3" のように指定します。

範囲は、単一の値 (符号付き数値または一重引用符で囲んだ文字列) または "lower - upper" のいずれかの形式で表します。lower と upper はいずれも符号付き数値または一重引用符で囲んだ文字列です。"lower" は "upper." 以下でなければなりません。文字列値に一重引用符を埋め込むには (例 : O'Brien)、一重引用符の前にバックスラッシュ (\) を 2 つ入れます (例 : O\\'Brien)。MIN を使用すると、マシンの関連する FIELD のデータ型の最小値を指定できます。MAX を使用すると、マシンの関連する FIELD のデータ型の最大値を指定できます。したがって、"MIN - -5" は -5 以下のすべての数値を指し、"6 - MAX" は、6 以上のすべての数値を指すことになります。範囲内のメタキャラクタ "*" (ワイルドカード) は、既にエントリとして指定した範囲では使用されなかった任意の値を示します。各エントリでは、1 つのワイルド

カードによる範囲指定だけが可能です。1 つのエントリで使用できるワイルドカード範囲は 1 つだけで、最後になければなりません (後続の範囲は無視される)。

ルーティング・フィールドには、FML でサポートされている任意のデータ型を指定できます。数値ルーティング・フィールドには数値で範囲を指定し、文字列ルーティング・フィールドには文字列で範囲を指定する必要があります。

文字列で範囲を設定する場合は、文字列、carray、および文字フィールド型の値を一重引用符で囲みます。先頭に符号を付けることはできません。short 型および long 型の整数値は数字の文字列であり、オプションで先頭に正の符号または負の符号を付けることができます。C コンパイラまたは `atof(3)` で使用できる浮動小数点数は、まず任意の符号、次に数字列 (小数点が入ってもよい)、任意の `e` または `E`、任意の符号またはスペース、最後に整数という形式を取ります。

グループ名は、フィールドが範囲と一致する場合の要求のルーティング先となるグループを示します。メタ文字 "*" (ワイルドカード) は、フィールド値が範囲と一致しない場合には要求がデフォルトのグループに送られること、またはフィールド値が範囲と一致していても、範囲エントリと関連するグループ内に実行可能なサーバが存在しない場合には、ワイルドカード "*" で指定された範囲エントリのデフォルトのグループにサービス要求が転送されることを示します。

範囲とグループの組み合わせの中では、範囲とグループ名は ":" で区切ります。

XML の要素の内容および属性の値は UTF-8 で符号化される必要があり、`FIELDTYPE` パラメータで指定されたデータ型に変換可能な場合にはルーティングに使用できます。

ルーティングに使用する場合、この要素の内容に文字リファレンス、エンティティ・リファレンス、および CDATA セクションを含めることはできません。

UTF-8 で符号化された XML 属性値は、この属性が属する要素が定義されている場合にルーティングに使用できます。

BUFTYPE = "*type1*[:*subtype1*[,*subtype2*...]][;*type2*[:*subtype3*[,...]]]..."

このルーティング・エントリで有効なデータ・バッファのタイプとサブタイプのリストを指定します。このパラメータの長さは 256 文字までです。また、最大 32 のタイプ / サブタイプの組み合わせを指定できます。タイプは、FML、FML32、XML、VIEW、VIEW32、X_C_TYPE、X_COMMON のいずれかでなければなりません。FML、FML32、XML にはサブタイプを指定できません。VIEW、VIEW32、X_C_TYPE、および X_COMMON にはサブタイプが必要です。サブ・タイプの名前には、セミコロン (;)、コロン (:)、カンマ (,)、アスタリスク (*) は使用できません。タイプとサブタイプのペアのうち、重複するものは同じルーティング基準名として指定できません。タイプとサブタイプのペアが一意の場合、複数のルーティング・エントリは同じ基準名を持つことができます。このパラメータは必須です。単一のルーティング・エントリに複数のバッファ・タイプが指定される場合、各バッファ・タイプに対するルーティング・フィールドのデータ型は同じでなければなりません。

次に、ルーティング・エントリの例を示します。

```
BRNCH FIELD=B_FLD RANGES="0-2:DBG1,3-5:DBG2,6-9:DBG3"
BUFTYPE="FML"
```

この例では、フィールド B_FLD のバッファの値 0-2 をサーバ・グループ DBG1 に、値 3-5 をサーバ・グループ DBG2 に、値 6-9 をサーバ・グループ DBG3 に送信しています。その他の値は使用できません。

フィールド値が設定されない場合 (FML バッファに対して) またはフィールド値が特定の範囲に一致せず、かつワイルドカードの範囲が指定されていない場合は、アプリケーションにエラーが戻されます。

次に、XML 要素の CODE に基づくルーティング・エントリの例を示します。

```
PRODUCT FIELD="ORDER/CODE" RANGES="'AAA' - 'FFF':DBG1,
'GGG-ZZZ':DBG2" BUFTYPE="XML"
```

ここでは、CODE はルート要素 ORDER の子要素です。

ORDERNO 属性に基づくルーティング・エントリは、次の例のようになります。

```
ORDER FIELD="ORDER/HEADER/@ORDERNO" FIELDTYPE=long
RANGES="0-9999:DBG1,10000-MAX:DBG3"  BUFTYPE="XML"
```

ここでは、ORDERNO は、ルート要素 ORDER の XML 子要素である HEADER の属性です。

UBBCONFIG(5) に関する追加情報

ファイル MASTER マシン上で TUXCONFIG コンフィギュレーション・ファイルを検索するには、環境変数 TUXCONFIG および TUXOFFSET を使用します。

使用例

```
# 次は、2 種類のマシン・タイプがある
# 2 サイトを含むコンフィギュレーション・ファイルの例です。データ依存型ルーティング
# が使用されます。
*RESOURCES
IPCKEY      80952 # 既知のアドレスのキー
DOMAINID    My_Domain_Name
UID         4196  #  IPC 構造体のユーザ ID
GID         601   # IPC 構造体のグループ ID
PERM        0660  # IPC アクセスのパーミッション
MAXSERVERS  20    # 同時に処理できるサーバの数を 20 に設定
MAXSERVICES 40    # 提供するサービス数を 40 に設定
MAXGTT     20   # 同時に処理できるグローバル･トランザクション数を 20 に設定
MASTER      SITE1
SCANUNIT    10
SANITYSCAN  12
BBLQUERY    180
BLOCKTIME   30
NOTIFY      DIPIN
OPTIONS     LAN,MIGRATE
SECURITY    USER_AUTH
AUTHSVC     AUTHSVC



     MP  # 掲示板に基づくマルチプロセッサ
LDBAL     Y  # ロード・バランシングを実行
#
*MACHINES
mach1 LMID=SITE1 TUXDIR="/usr4/tuxbin"
       MAXACCESSERS=25
       APPDIR="/usr2/apps/bank"
       ENVFILE="/usr2/apps/bank/ENVFILE"
       TLOGDEVICE="/usr2/apps/bank/TLOG" TLOGNAME=TLOG
       TUXCONFIG="/usr2/apps/bank/tuxconfig" TYPE="3B2"
       ULOGPFX="/usr2/apps/bank/ULOG"
       SPINCOUNT=5
mach386    LMID=SITE2 TUXDIR="/usr5/tuxbin"
       MAXACCESSERS=100
       MAXWSCLIENTS=50
       APPDIR="/usr4/apps/bank"
```

```
        ENVFILE="/usr4/apps/bank/ENVFILE"
        TLOGDEVICE="/usr4/apps/bank/TLOG" TLOGNAME=TLOG
        TUXCONFIG="/usr4/apps/bank/tuxconfig" TYPE="386"
        ULOGPFX="/usr4/apps/bank/ULOG"
#
*GROUPS

DEFAULT:   TMSNAME=TMS_SQL  TMSCOUNT=2
# Windows の場合、:bankdb: を ;bankdb; にする
BANKB1    LMID=SITE1  GRPNO=1
  OPENINFO="TUXEDO/SQL:/usr2/apps/bank/bankdl1:bankdb:readwrite"
# Windows の場合、:bankdb: を ;bankdb; にする
BANKB2    LMID=SITE2  GRPNO=2
  OPENINFO="TUXEDO/SQL:/usr4/apps/bank/bankdl2:bankdb:readwrite"
DEFAULT:
AUTHGRP    LMID=SITE1  GRPNO=3
#
*NETWORK
SITE1     NADDR="mach1.80952" BRIDGE="/dev/starlan"
          NLSADDR="mach1.serve"
#
SITE2     NADDR="mach386.80952" BRIDGE="/dev/starlan"
          NLSADDR="mach386.serve"
*SERVERS
#
DEFAULT: RESTART=Y MAXGEN=5 REPLYQ=Y CLOPT="-A"

TLR       SRVGRP=BANKB1  SRVID=1  RQADDR=tlr1
          CLOPT="-A -- -T 100"
TLR       SRVGRP=BANKB1  SRVID=2  RQADDR=tlr1
          CLOPT="-A -- -T 200"
TLR       SRVGRP=BANKB2  SRVID=3  RQADDR=tlr2
          CLOPT="-A -- -T 600"
TLR       SRVGRP=BANKB2  SRVID=4  RQADDR=tlr2
          CLOPT="-A -- -T 700"
XFER      SRVGRP=BANKB1  SRVID=5
XFER      SRVGRP=BANKB2  SRVID=6
ACCT      SRVGRP=BANKB1  SRVID=7
ACCT      SRVGRP=BANKB2  SRVID=8
BAL       SRVGRP=BANKB1  SRVID=9
BAL       SRVGRP=BANKB2  SRVID=10
BTADD     SRVGRP=BANKB1  SRVID=11
BTADD      SRVGRP=BANKB2  SRVID=12
AUTHSVR   SRVGRP=AUTHGRP SRVID=20 #
*SERVICES
DEFAULT:    LOAD=50     AUTOTRAN=N
WITHDRAWAL  PRIO=50     ROUTING=ACCOUNT_ID
DEPOSIT     PRIO=50     ROUTING=ACCOUNT_ID
TRANSFER    PRIO=50     ROUTING=ACCOUNT_ID
```

```
INQUIRY     PRIO=50     ROUTING=ACCOUNT_ID
CLOSE_ACCT  PRIO=40     ROUTING=ACCOUNT_ID
OPEN_ACCT   PRIO=40     ROUTING=BRANCH_ID


BR_ADD      PRIO=20     ROUTING=BRANCH_ID
TLR_ADD     PRIO=20     ROUTING=BRANCH_ID
ABAL        PRIO=30     ROUTING=b_id
TBAL        PRIO=30     ROUTING=b_id
ABAL_BID    PRIO=30     ROUTING=b_id
TBAL_BID    PRIO=30     ROUTING=b_id SVCTIMEOUT=300
#
#
*ROUTING
ACCOUNT_ID  FIELD=ACCOUNT_ID  BUFTYPE="FML"
 RANGES="MIN - 9999:*,10000-59999:BANKB1,60000-109999:BANKB2,*:*"
BRANCH_ID   FIELD=BRANCH_ID   BUFTYPE="FML"
 RANGES="MIN - 0:*,1-5:BANKB1,6-10:BANKB2,*:*"
b_id     FIELD=b_id  BUFTYPE="VIEW:aud"
 RANGES="MIN - 0:*,1-5:BANKB1,6-10:BANKB2,*:*"
```

相互運用性

相互運用するアプリケーションでは、マスタ側で最新のリリースが動作していなければなりません。異なるリリースの BEA Tuxedo システム間で相互運用する場合、PMID (マシン ADDRESS)、LMID、TLOGNAME、グループ名、RQADDR、サービス名、および ROUTING (ルーティング基準名) の各パラメータ値は、有効な C 言語識別子 (UBBCONFIG キーワードではない) でなければなりません。

ネットワーク・アドレス

BRIDGE を実行しているローカル・マシンが TCP/IP アドレス指定機能を使用していて、そのアドレスは 155.2.193.18 であり、名前は backus.company.com であるとします。また、BRIDGE が要求を受け取るポート番号は 2334 だとします。このポート番号 2334 は、bankapp-naddr という名前のネットワーク・サービス・データベースに追加されているとします。この場合、アドレスは次のように表現されます。

```
//155.2.193.18:bankapp-naddr//155.2.193.18:2334
//backus.company.com:bankapp-naddr
//backus.company.com:2334
0x0002091E9B02C112
```

最後の表現は 16 進形式です。0002 は TCP/IP アドレスの先頭部分、091E は 16 進数に変換されたポート番号 2334 です。その後、IP アドレス 155.2.193.1 の各要素は 16 進数に変換されています。つまり、155 は 9B、2 は 02 というようになります。

関連項目 buildserver(1)、tmadmin(1)、tmboot(1)、tmloadcf(1)、tmshutdown(1)、tmunloadcf(1)、buffer(3c)、tpinit(3c)、servopts(5)

『BEA Tuxedo アプリケーションの設定』

『BEA Tuxedo アプリケーション実行時の管理』

『C 言語を使用した BEA Tuxedo アプリケーションのプログラミング』

viewfile(5)

名前 viewfile—VIEW 記述用のソース・ファイル

機能説明 VIEW ファイルは、1 つまたは複数の C データ構造体 (VIEW) の記述用ソース・ファイルです。viewc() コマンドへの入力として使用される場合、VIEW ファイルはバイナリ・ファイル (ファイル名 *view_filename*.V) とヘッダ・ファイル (*view_filename*.h) の基礎となります (viewc、viewc32(1) を参照)。

BEA Tuxedo システムでは、バイナリ .V ファイルは次の 2 とおりの方法で使用されます。

- Fvftos() と Fvstof() を使用するプログラムでは、.V ファイルは実行時に解釈され、FML バッファと C 構造体との間のマッピングが行われます。
- VIEW および VIEW32 型付きバッファを割り当てるプログラムでは、.V ファイルは tpalloc() の *subtype* 引数で与えられる名前を持つ構造体の検索に使用されます。

構造体のメンバがその論理名によって参照できるように、VIEW を使用するすべてのプログラムに .h ファイルをインクルードする必要があります。

VIEW 記述 ソース viewfile の各 VIEW 記述は、次の 3 つの部分から構成されます。

- キーワード "VIEW" で始まり、その後に VIEW 記述の名前が続く行。この名前は最大 33 文字で、有効な C 識別子でなければなりません。つまり、先頭がアンダースコアまたはアルファベットで、英数字またはアンダースコアしか使用できません。tpalloc(3c) で使用する場合、名前に使用できる文字数は 16 文字以内です。
- メンバ記述のリスト。各行には 7 つのフィールドがあります。
- キーワード "END" で始まる行。

各 VIEW 記述の先頭行は、キーワード "VIEW" で始まり、後ろに VIEW 記述の名前が続かなければなりません。メンバ記述 (またはマッピング・エントリ) は、C 構造体のメンバに関する情報が含まれている行です。。キーワード "END" で始まる行は、VIEW 記述の最後の行です。また、シャープ (#) で始まる行はコメントとして扱われ、無視されます。

したがって、ソース VIEW 記述は、一般に次のような構造になっています。

```
VIEW vname
# type  cname  fbname  count  flag  size  null
# ----  -----  ------  -----  ----  ----  ----
  -------------- メンバの記述 ------------------

.
.
END
```

記述の変数フィールドには以下のような意味があります。

vname
: VIEW 記述の名前。これは C 構造体の名前としても使用されるため、有効な C 識別子を指定します。

type
: メンバの型。int、short、long、char、float、double、string、carray、または dec_t のいずれかを指定します。'-' を指定すると、VIEW が FML バッファにマップされる場合、メンバの型にはデフォルトで *fbname* 型が設定されます。

cname
: 構造体メンバの識別子。これは C 構造体メンバの名前であるため、有効な C 識別子名を指定します。VIEW が FML バッファにマップされない場合、これは有効な *fbname* ではありません。

fbname
: フィールド化バッファのフィールドの名前。この名前には、フィールド・テーブル・ファイルまたはフィールド・ヘッダ・ファイルのいずれかにある名前を指定します。FML バッファにマップしない VIEW の場合、このフィールドは無視されますが、ダッシュ (-) などのプレースホルダ値を含んでいる必要があります。

count

割り当てる要素の数、つまりこのメンバに格納されるオカレンスの最大数を示します。65,535 以下の値を指定します。

flag

カンマで区切ったオプションのリスト。ダッシュ (-) は、オプションが設定されていないことを示します。*flag* オプションについては後述します。FML バッファにマップしない VIEW の場合、このフィールドには C または L オプション (あるいはその両方) を指定でき、またはプレース・ホルダ値のダッシュ ( ) を指定する必要があります。

size

型が string または carray の場合、メンバのサイズを示します。65,535 以下の値を指定します。32 ビット FML の場合、最大サイズは 2 の 32 乗です。dec_t 型の場合、*size* は 2 つの数値をカンマで区切ります。1 つ目の数値は 10 進数値のバイト数 (0 より大きく 10 未満) で、2 つ目の数値は小数点の右側の桁数です (0 より大きく、バイト数の 2 倍から 1 を引いた値未満)。その他のフィールド・タイプの場合、ダッシュ (-) を指定すると VIEW コンパイラがサイズを計算します。

null

ユーザ指定の NULL 値。ダッシュ (-) を指定すると、該当するフィールドのデフォルトのヌル値が設定されます。ヌル値については後述します。

フラグ・オプション

以下に、VIEW 記述内のメンバ記述の flag 要素として指定できるオプションのリストを示します。L および C オプションは、FML ベースではない VIEW 用にも構造体メンバを追加生成します。

C

このオプションは、メンバ記述で記述されている構造体メンバに加えて、関連するカウント・メンバ (ACM) と呼ばれる付加的な構造体メンバが生成されることを示します (FML ベースではない VIEW も同様)。データをフィールド化バッファから構造体に転送する場合は、その構造体内の各 ACM を、関連付けられている構造体メンバに転送されるオカレンスの数に設定します。ACM の値 0 は、対応する構造体メンバにフィールドが転送されなかったことを示します。正の値は、構造体メンバの配列に実際に転送されたフィールドの数を示します。負の値は、構造体メンバの配列に転送できる以上のフィールドがバッ

ファ内にあったことを示します (ACM の絶対値は、構造体に転送されなかったフィールドの数と同等です)。構造体メンバの配列からフィールド化バッファへのデータ転送時には、ACM を使用して転送すべき配列要素の数を指定します。たとえば、あるメンバの ACM が N に設定されている場合は、最初の N のヌルではないフィールドがフィールド化バッファに転送されます。N が配列のサイズより大きい場合、N はデフォルトで配列のサイズに設定されます。どちらの場合も、転送後、ACM はフィールド化バッファに転送される配列のメンバの実際の数に設定されます。ACM の型は short (VIEW32 の場合は 32 ビット long) として宣言され、その名前は "C_*cname*" という形式で生成されます。ここで *cname* は、ACM が宣言されている cname エントリです。たとえば、parts という名前のメンバの ACM は、次のように宣言されます。

```
short C_parts;
```

生成された ACM 名が、先頭に 接頭辞 "C_" が付いている構造体メンバと矛盾する場合もあります。そのような矛盾は VIEW コンパイラにより報告され、これを VIEW コンパイラは致命的なエラーとみなします。たとえば、構造体のメンバの名前が "C_parts" である場合、これは、メンバ "parts" のために生成された ACM の名前と矛盾します。また、-r コマンド行オプションを指定すると、VIEW コンパイラは ACM メンバと ALM (後述の L オプション参照) メンバの構造化レコード定義を生成します。

F

構造体からフィールド化バッファへの一方向のマッピングを指定します。このオプションによるメンバのマッピングは、構造体からフィールド化バッファへのデータ転送時のみ有効です。

L

このオプションは、carray 型または string 型のメンバ記述に対してのみ使用され、これらの可変長フィールドに対して転送されたバイト数を示します。string フィールドまたは carray フィールドが常に固定長データ項目として使用される場合、このオプションを使用するメリットはありません。L オプションを指定すると、carray または string 型の構造体メンバに関連する長さメンバ (ALM) が生成されます (FML ベースではない VIEW も同様)。フィールド化バッファから構造体にデータを転送する場合、ALM は対応する転送されたフィールドの長さに設

定されます。フィールド化バッファのフィールドの長さがマップされた構造体メンバで割り当てられた領域を超える場合、割り当てられたバイト数しか転送されません。対応する ALM は、フィールド化バッファ項目のサイズに設定されます。このため、ALM が構造体メンバの配列の大きさを超えた場合、フィールド化バッファ情報は転送時に切り捨てられます。構造体メンバからフィールド化バッファのフィールドにデータを転送する場合、そのフィールドが carry 型であれば、ALM を使用してフィールド化バッファに転送するバイト数を示します。string 型のフィールドの場合、ALM は転送時には無視されますが、後で転送されたバイト数に設定されます。carry フィールドの長さはゼロであることもあるため、ALM が 0 である場合、関連する構造体メンバの値がヌル値でなければ、フィールド化バッファに長さゼロのフィールドが転送されることを示します。ALM は unsigned short (VIEW32 の場合は 32 ビット unsigned long) として定義され、"L_*cname*" という名前が付けられます。ここで *cname* は ALM が宣言される構造体の名前です。宣言する ALM に対応するメンバのオカレンス数が 1 である場合、またはデフォルト値が 1 に設定されている場合、ALM は次のように宣言します。

```
unsigned short L_cname;
```

一方、オカレンス数が 1 より大きい場合 (N)、ALM は次のように宣言されます。

```
unsigned short L_cname[N];
```

これは ALM 配列として参照されます。このような場合は、ALM 配列内の各要素は構造体メンバ (またはフィールド) の対応するオカレンスを参照します。生成された ALM 名と先頭に "L_" 接頭辞が付いている構造体メンバが矛盾する場合もあります。そのような矛盾は VIEW コンパイラにより報告され、これを VIEW コンパイラは致命的なエラーとみなします。たとえば、構造体のメンバの名前が "L_parts" である場合、これは、メンバ "parts" のために生成された ALM の名前と矛盾します。また、-r コマンド行オプションを指定すると、VIEW コンパイラは ACM および ALM (前述の C オプションを参照) メンバの構造化レコード定義を生成します。

N

ゼロ方向マッピングを指定します。つまり C 構造体にフィールド化バッファはマップされません。このオプションでは、FML ベースでない VIEW では無視されます。これは、C 構造体に充填文字を割り当てるときに使用することができます。

P

このオプションは、string および carry 型の構造体メンバのヌル値として解釈される値を決定するために使用します。このオプションは、FML ベースでない VIEW では無視されます。このオプションを指定しない場合、構造体メンバの値がユーザ指定のヌル値と等しいと (後続のヌル文字は考慮しない)、構造体メンバがヌルになります。一方、このオプションを指定した場合は、ユーザ指定のヌル値と構造体メンバの値が等しく、さらに最後の文字が完全な長さまで及んでいると (後続のヌル文字は考慮しない)、構造体メンバがヌルになります。値がヌルであるメンバは、データが C 構造体からフィールド化バッファに転送されても宛先バッファに転送されません。たとえば、構造体メンバ TEST の型が carry[25] で、ユーザ指定のヌル値 "abcde" が指定されているとします。P オプションが設定されていない場合、最初の 5 文字が a、b、c、d、e であれば TEST はヌルと見なされます。P オプションが設定されている場合、最初の 4 文字 a、b、c、d で、この文字配列の残りの文字の中に 'e' (21 個の e) があれば TEST はヌルとなります。

S

フィールド化バッファから構造体への一方向のマッピングを指定します。このオプションは、FML ベースでない VIEW では無視されます。このオプションを指定したメンバのマッピングは、フィールド化バッファから構造体へのデータ転送時のみ有効です。

ヌル値 ヌル値は、C 構造体のメンバが空であることを示すために使用されます。デフォルトのヌル値が指定されていますが、独自のヌル値を定義することもできます。

デフォルトのヌル値は、数値型では 0 (dec_t の場合は 0.0)、char 型では "\"、string および carry 型では "" です。

ヌル値を指定するためにエスケープ規則定数を使用することもできます。VIEW コンパイラが認識するエスケープ定数は、*ddd* (*d* は 8 進数)、0、n、t、v、b、r、f、、'、および " です。

string、carray、および char 型のヌル値は、二重引用符または一重引用符で囲まれている場合があります。VIEW コンパイラは、ユーザ定義のヌル値の内部のエスケープされていない引用符は受け付けません。

要素は、値がその要素の NULL 値と同じであれば NULL になりますが、以下の例外があります。

- 構造体メンバに対して P オプションが設定されており、構造体メンバが string 型か carray 型である場合、特別な制約が加わります。P オプション・フラグの詳細については、上記の説明を参照してください。
- メンバが string 型である場合。その値は必ずヌル値と同じ文字列です。
- メンバが carry 型で、ヌル値の長さが N である場合、文字配列の最初の N 文字は必ずヌル値と同じです。

VIEW 記述のヌル・フィールドにキーワード "NONE" を指定することもできます。これは、そのメンバに対するヌル値がないことを意味します。

文字列および文字配列のメンバのデフォルトの最大サイズは 2,660 文字です。

文字列メンバの場合、通常 "0" で終了するため、ユーザ定義のヌル値の最後の文字として "0" は必要ありません。

環境変数

`VIEWFILES`

アプリケーションのオブジェクト VIEW ファイルのカンマ区切りのリストを指定します。絶対パス名で指定したファイルはそのまま使用され、相対パス名で指定したファイルは `VIEWDIR` 変数 (下記参照) で指定したディレクトリのリストから検索されます。

`VIEWDIR`

オブジェクト・ファイルが存在する可能性のあるディレクトリのコロン区切りのリストを指定します。`VIEWDIR` を設定しない場合、この値はカレント・ディレクトリとなります。

`VIEW32` では、環境変数 `VIEWFILES32` と `VIEWDIR32` が使用されます。

使用例

```
# FML ベースの VIEW ファイルの開始
VIEW custdb
$/* コメント行 */
#
#type    cname   fbname count flag  size  null
```

```
#
carray  bug   BUG_CURS 4    -   12   "no bugs"
long   custid CUSTID  2    -   -   -1
short   super  SUPER_NUM 1    -   -   999
long   youid  ID    1    -   -   -1
float   tape  TAPE_SENT 1    -   -   -.001
char   ch   CHR    1    -   -   "0"
string  action ACTION  4    -   20   "no action"
END
# 非依存 VIEW ファイルの開始
VIEW viewx
$ /* viewx 情報のための VIEW 構造体 */
#
#type    cname   fbname count flag  size  null
#
int     in    -   1   -   -   -
short    sh    -   2   -   -   -
long     lo    -   3   -   -   -
char     ch    -   1   -   -   -
float    fl    -   1   -   -   -
double    db    -   1   -   -   -
string    st    -   1   -   15   -
carray    ca    -   1   -   15   -
END
```

関連項目 viewc、viewc32(1), tpalloc(3c), Fvftos、Fvftos32(3fml), Fvstof、Fvstof32(3fml)

『FML を使用した BEA Tuxedo アプリケーションのプログラミング』

WS_MIB(5)

名前 WS_MIB— ワークステーションの管理情報ベース

形式

```
#include <fml32.h>
#include <tpadm.h>
```

機能説明 BEA Tuxedo システムの MIB は、ワークステーション・グループ (1 つの WSL とそれに関連した WSH プロセス) を管理するためのクラスのセットを定義します。

管理要求のフォーマットと管理応答の解釈を行うには、WS_MIB(5) を共通 MIB リファレンス・ページ MIB(5) と一緒に使用します。このリファレンス・ページで説明するクラスや属性を使用し、MIB(5) の説明に従ってフォーマットした要求を使用すると、アクティブなアプリケーションの既存の ATMI インターフェイスの 1 つを通じて管理サービスを要求できます。WS_MIB(5) のすべてのクラス定義の追加情報については、697 ページの「WS_MIB(5) に関する追加情報」を参照してください。

WS_MIB(5) は、次のクラスで構成されています。

表 65WS_MIB **のクラス**

クラス名	属性
T_WSH	ワークステーション・ハンドラ
T_WSL	ワークステーション・リスナ

各クラスの説明セクションには、次の 4 つのサブセクションがあります。

概要
: このクラスに関連付けられている属性の概要

属性表
: クラスの各属性に関する名前、タイプ、パーミション、値、およびデフォルト値を示す表。属性表の形式については以下に示してあります。

属性の意味
: 各属性の意味の説明

制限事項
: このクラスにアクセスし、このクラスを解釈する場合の制限事項

属性表の形式

前述のように、この MIB に含まれる各クラスは、4 つの部分に分けて以下に定義されています。その 1 つが属性表です。属性表はクラス内の属性のリファレンス・ガイドであり、管理者、オペレータ、一般ユーザがそれらの属性を使用してアプリケーションと対話するための方法を説明しています。属性表の各属性の説明には、5 つの構成要素 (名前、タイプ、パーミッション、値、デフォルト) があります。各要素については、MIB(5) を参照してください。

TA_FLAGS 値

MIB(5) は、共通 TA_FLAGS 属性を定義します。この属性は long 値フィールドで、共通 MIB フラグ値とコンポーネント MIB 固有フラグ値の両方を持ちます。現時点では、WS_MIB(5) 固有のフラグ値は定義されていません。

FML32 フィールド・テーブル

このリファレンス・ページで説明する属性のフィールド・テーブルは、システムにインストールした BEA Tuxedo システム・ソフトウェアのルート・ディレクトリからの相対パスで指定される udataobj/tpadm ファイルにあります。${TUXDIR}/udataobj ディレクトリは、FLDTBLDIR 環境変数で指定されるコロン区切りのリストにアプリケーションによって追加される必要があり、フィールド・テーブル名 tpadm() は、FIELDTBLS 環境変数で指定されるカンマ区切りのリストに追加される必要があります。

制限事項

この MIB のヘッダ・ファイルとフィールド・テーブルには、BEA Tuxedo システム 6.0 以降が実行されているサイト (ネイティブとワークステーションの両方) からのみアクセスできます。

T_WSH クラスの定義

概要 T_WSH クラスは、WSH クライアント・プロセスの実行時の属性を表します。この属性値は、特定の WSH クライアント・プロセスに固有のワークステーション統計情報の特性を示します。このクラスは、共通キー・フィールドの TA_SRVGRP と TA_SRVID によって T_WSL にリンクされます。さらに、共通キー・フィールドの TA_WSHCLIENTID によって T_CLIENT クラス (TM_MIB(5) を参照) にもリンクされます。

属性表

表 66WS_MIB(5): T_WSH クラス定義の属性表

属性 [1]	タイプ	パーミッション	値	デフォルト値
TA_CLIENTID(*)	string	R--R--R--	*string*[1..78]	N/A
TA_WSHCLIENTID(*)	string	R--R--R--	*string*[1..78]	N/A
TA_SRVGRP(*)	string	R--R--R--	*string*[1..30]	N/A
TA_SRVID(*)	long	R--R--R--	1 <= *num* < 30,001	N/A
TA_GRPNO(*)	long	R--R--R--	1 <= *num* < 30,000	N/A
TA_STATE(k)	string	R-XR-XR--	TM_MIB(5) の T_CLIENT クラスを参照してください。	
TA_LMID(*)	string	R--R--R--	*LMID*	N/A
TA_PID(*)	long 型	R--R--R--	1 <= *num*	N/A
TA_NADDR	string	R--R--R--	*string*[1..256] [2]	N/A
TA_HWCLIENTS	long	R--R--R--	1 <= *num* < 32,767	N/A
TA_MULTIPLEX	long	R--R--R--	1 <= *num* < 32,767	N/A
TA_CURCLIENTS	long	R--R--R--	1 <= *num* < 32,767	N/A
TA_TIMELEFT	long	R--R--R--	0 <= *num*	N/A

表 66WS_MIB(5): T_WSH **クラス定義の属性表 (続き)**

属性 [1]	タイプ	パーミッション	値	デフォルト値
TA_ACTIVE	string	R--R--R--	"{Y | N}"	N/A
TA_TOTACTTIME	long	R--R--R--	0 <= *num*	N/A
TA_TOTIDLTIME	long	R--R--R--	0 <= *num*	N/A
TA_CURWORK	long	R--R--R--	0 <= *num*	N/A
TA_FLOWCNT	long	R--R--R--	0 <= *num*	N/A
TA_NUMBLOCKQ	long	R--R--R--	0 <= *num*	N/A
TA_RCVDBYT	long	R--R--R--	0 <= *num*	N/A
TA_RCVDNUM	long	R--R--R--	0 <= *num*	N/A
TA_SENTBYT	long	R--R--R--	0 <= *num*	N/A
TA_SENTNUM	long	R--R--R--	0 <= *num*	N/A

(k)—GET キー・フィールド
(*)—GET/SET キー、SET 操作用に 1 つ以上必要

[1] T_WSH クラスの属性はすべてローカル属性です。
[2] BEA Tuxedo 8.0 以前のリリースの場合、この属性の文字列の長さは最大 78 バイトです。

属性の意味

TA_CLIENTID: *string*[1..78]
この WHS のクライアント識別子。等号比較の場合を除き、エンド・ユーザはこのフィールドのデータを直接解釈することはできません。

TA_WSHCLIENTID: *string*[1..78]
Client identifier for this WSH. 等号比較の場合を除き、エンド・ユーザはこのフィールドのデータを直接解釈することはできません。このフィールドを使用して、WSH をそれに対応するワークステーション・クライアントの T_CLIENT オブジェクトにリンクできます。この

フィールド値は、このクラスの TA_CLIENTID 属性の値と常に同じです。

TA_SRVGRP: *string*[1..30]
関連付けられた WSL のサーバ・グループの論理名。

TA_SRVID: 1 <= *num* < 30,001
関連付けられた WSL の一意の (サーバ・グループ内の) サーバ識別番号。

TA_STATE:
アプリケーション内の WSH クライアントの状態。TM_MIB(5) の T_CLIENT クラスに対して定義された任意の状態が返されるか、またはこのリファレンス・ページに示すとおり設定します。SUSpended 状態への状態変更は、SUSpended 状態の WSH の ACTive へのリセットと同様に、この WSH に関連付けられたすべてのクライアントにとって過渡的なものです。さらに、SUSpended 状態の WSH クライアントには、WSL によって接続してくるクライアントがこれ以上割り当てられることはありません。T_CLIENT クラスにアクセスするときに WSH クライアントを DEAD に設定することができない場合があります。ただし、DEAD への状態変化は T_WSH クラスを介して可能となり、この場合、WSH によって処理されるすべての接続が切断されます。

TA_LMID:*LMID*
WSH が稼働している現在の論理マシン。

TA_PID: 1 = *num*
WSH クライアントのネイティブ・オペレーティングシステム・プロセス識別子。各クライアントが異なるマシンに存在し、プロセス識別子が重複するような場合、この属性は一意になりません。

TA_NADDR: *string*[1..256] (BEA Tuxedo 8.0 以前の場合は最大 78 バイト)
ワークステーション・ハンドラのネットワーク・アドレス。16 進のアドレスは先頭に 0x が付いた ASCII 形式に変換されます。TCP/IP アドレスは、"//*#.#.#.#:port*" という形式で報告されます。

TA_HWCLIENTS: 1 <= *num* <32,767
この WSH を介してアプリケーションにアクセスするクライアントの最大数。

TA_MULTIPLEX: 1 <= *num* <32,767
この WSH を介してアプリケーションにアクセスする可能性のあるクライアントの最大数。

TA_CURCLIENTS: 1 <= *num* <32,767
この WSH を介してアプリケーションにアクセスするクライアントの現在数。

TA_TIMELEFT: 0 <= *num*
この属性に 0 以外の値が設定されている場合、指定された時間 (秒数) で初期化プロセスを完了する新たに接続するワークステーション・クライアントが WSH に割り当てられています。

TA_ACTIVE:{Y | N}
値 Y は、WSH がそれに関連付けられているワークステーション・クライアントの 1 つの代わりに作業を実行していることを示しています。値 N は、WSH がそれに関連付けられたワークステーション・クライアントの 1 つの代わりに実行する処理を待っていることを示しています。

TA_TOTACTTIME: 0 <= *num*
WSH が処理の開始からアクティブ状態にあった時間 (秒数)。

TA_TOTIDLTIME: 0 <= *num*
WSH が処理の開始からアイドル状態にあった時間 (秒数)。

TA_CURWORK: 0 <= *num*
WSL による最後の WSH 割り当て後に、この WSH によって処理された作業量。この値は、WSL が一連の WSH プロセスの間で新しい接続のロード・バランシングを行うために使用します。

TA_FLOWCNT: 0 <= *num*
この WSH でフロー制御が発生した回数。WSH のライフタイム中に元に戻る可能性があるので、この属性は最近の過去値のみに基づいて考慮する必要があります。

TA_NUMBLOCKQ: 0 <= *num*
キュー・ブロック状態のために、この WSH をローカル UNIX システム・メッセージ・キューに入れることができなかった回数。WSH の

ライフタイム中に元に戻る可能性があるので、この属性は最近の過去値のみに基づいて考慮する必要があります。

TA_RCVDBYT: 0 <= *num*
: この WSH が、その現在と過去のワークステーション・クライアントのすべてから受信したバイト数。WSH のライフタイム中に元に戻る可能性があるので、この属性は最近の過去値のみに基づいて考慮する必要があります。

TA_RCVDNUM: 0 <= *num*
: この WSH が、その現在と過去のワークステーション・クライアントのすべてから受信した BEA Tuxedo システム・メッセージ数。WSH のライフタイム中に元に戻る可能性があるので、この属性は最近の過去値のみに基づいて考慮する必要があります。

TA_SENTBYT: 0 <= *num*
: この WSH が、その現在と過去のすべてのワークステーション・クライアントに送信したバイト数。WSH のライフタイム中に元に戻る可能性があるので、この属性は最近の過去値のみに基づいて考慮する必要があります。

TA_SENTNUM: 0 <= *num*
: この WSH が、その現在と過去のすべてのワークステーション・クライアントに送信した BEA Tuxedo システム・メッセージ数。WSH のライフタイム中に元に戻る可能性があるので、この属性は最近の過去値のみに基づいて考慮する必要があります。

制限事項

このクラスは T_CLIENT クラスの特殊な場合を表すクラスなので、対応する T_CLIENT オブジェクトでも重複して定義されている特定の属性を表します。T_CLIENT クラスに含まれる属性のうちここに記載されていないものにはこのクラスを介してアクセスする必要があり、T_WSH クラスからアクセスすることはできません。

WSH サーバの属性は、実行時環境でのみ意味を持ちます。したがって、起動されていない環境で tpadmcall(3c) 関数を使用しても、これらの属性を変更することはできません。

T_WSL クラスの定義

概要 T_WSL クラスは、ワークステーション・グループを管理するためにコンフィギュレーションされた WSL サーバ・プロセスのコンフィギュレーションおよび実行時の属性を表します。これらの属性値により、アプリケーション内の T_SERVER オブジェクトのワークステーション固有のコンフィギュレーション属性が識別され、その特性が示されます。このクラスは、共通キー・フィールドの TA_SRVGRP と TA_SRVID によって T_WSH にリンクされます。

属性表

表 67WS_MIB(5): T_WSL クラス定義の属性表

属性	タイプ	パーミッション	値	デフォルト値
TA_SRVGRP(r)(*)	string	ru-r--r--	*string*[1. .30]	N/A
TA_SRVID(r)(*)	long	ru-r--r--	1 <= *num* < 30,001	N/A
TA_GRPNO(k)	long	r--r--r--	1 <= *num* < 30,001	N/A
TA_STATE(k)	string	rwxr-xr--	TM_MIB(5) の T_SERVER クラスを参照してください。	
TA_LMID(k)	string	R--R--R--	*LMID*	N/A
TA_PID(k)	long	R--R--R--	1 <= *num*	N/A
TA_DEVICE	string	rw-r--r--	*string*[0..78]	N/A
TA_NADDR(r)	string	rw-r--r--	*string*[1..256] [3]	N/A
TA_EXT_NADDR	string	rw-r--r--	*string*[0..78]	""
TA_WSHNAME	string	rw-r--r--	*string*[1..78]	"WSH"
TA_MINHANDLERS	long	rwxr-xr--	0 <= *num* < 256	0
TA_MAXHANDLERS	long	rw-r--r--	0 <= *num* < 32,767	[1] を参照してください。
TA_MULTIPLEX	long	rw-r--r--	1 <= *num* < 32,767	10

表 67WS_MIB(5): T_WSL **クラス定義の属性表 (続き)**

属性	タイプ	パーミッション	値	デフォルト値
TA_MINENCRYPTBITS	string	rwxrwx---	"{0 | 40 | 56 | 128}" [2]	"0"
TA_MAXENCRYPTBITS	string	rwxrwx---	"{0 | 40 | 56 | 128}" [2]	"128"
TA_MINWSHPORT	long	rwxr-xr--	0 <= *num* < 65,535	2048
TA_MAXWSHPORT	long	rw-r--r--	0 <= *num* < 65,535	65,535
TA_MAXIDLETIME	long	rwxr-xr--	0 <=*num* < 35,204,650	35,204,649
TA_MAXINITTIME	long	rwxr-xr--	1 <= *num* < 32,767	60
TA_CMPLIMIT	string	rwxr-xr--	*threshold*	MAXLONG
TA_CLOPT	string	rwxr--r--	*string*[0..128]	"-A"
TA_ENVFILE	string	rwxr--r--	*string*[0..256] [3]	""
TA_GRACE	long	rwxr--r--	0 <= *num*	0
TA_KEEPALIVE	string	rwxr-xr--	"{client | handler | both | none}"	"none"
TA_MAXGEN	long	rwxr--r--	0 <= *num* < 256	1
TA_NETTIMEOUT	long	rwxr-xr--	0<=*num*<= MAXLONG	0
TA_RCMD	string	rwxr--r--	*string*[0..256] [3]	""
TA_RESTART	string	rwxr--r--	"{Y | N}"	"Y"
TA_SEQUENCE(k)	long	rwxr--r--	1 <= *num* < 10,000	>= 10,000
T_WSL クラス定義のローカル属性表				
TA_CURHANDLERS	long	R--R--R--	0 <= *num*	N/A
TA_HWHANDLERS	long	R--R--R--	0 <= *num*	N/A

表 67WS_MIB(5): T_WSL **クラス定義の属性表 (続き)**

属性	タイプ	パーミッション	値	デフォルト値
TA_WSPROTO	long	R--R--R--	0 <= *num*	N/A
TA_SUSPENDED	string	R-XR-XR--	"{NEW | ALL | NONE}"	N/A
TA_VIEWREFRESH	string	--X--X---	"Y"	N/A

(k)—GET キー・フィールド
(r)— オブジェクトの作成に必要なフィールド (SET TA_STATE NEW)
(*)—GET/SET キー、SET 操作用に 1 つ以上必要

[1] オブジェクト作成時、この属性に値を指定しないと、0 が割り当てられます。この属性に 0 を指定すると、アクティブ化時に、TA_MAXHANDLERS の現在の設定値と TA_MAXWSCLIENTS の T_MACHINE クラス設定値から有効値が決定されます。MIB_LOCAL フラグを設定して GET 操作を実行すると、起動時のデフォルト設定値でオブジェクトの有効値が返されるということに注意してください。
[2] リンク・レベルの暗号化の値の 40 ビットは、下位互換性を維持するために提供されています。
[3] BEA Tuxedo 8.0 以前のリリースの場合、この属性の文字列の長さは最大 78 バイトです。

属性の意味

TA_SRVGRP: *string*[1..30]
サーバ・グループの論理名。サーバ・グループ名には、アスタリスク (*)、カンマ (,)、コロン (:) を使用することはできません。

TA_SRVID: 1 <= *num* < 30,001
サーバ・グループ内で一意なサーバ識別番号。

TA_GRPNO: 1 <= *num* < 30,001
このサーバ・グループに関連付けられたグループ番号。

TA_STATE:
アプリケーション内の WSH サーバの状態。TM_MIB(5) の T_SERVER クラスに対して定義された任意の状態が返されるか、またはこのリファレンス・ページに示すとおりに設定します。

TA_LMID:*LMID*
サーバが稼働している現在の論理マシン。

TA_PID: 1 = *num*
WSH サーバのネイティブ・オペレーティングシステム・プロセス識別子。各サーバが別のマシンに存在し、プロセス ID が重複するような場合、この属性は一意になりません。

TA_DEVICE: *string*[0..78]
WSL プロセスがネットワークにアクセスするために使用するデバイス名。この属性はオプションです。

TA_NADDR: *string*[1..256] (BEA Tuxedo 8.0 以前の場合は最大 78 バイト)
WSL プロセスが接続指示受け付けアドレスとして使用する完全なネットワーク・アドレスを指定します。WSL の接続指示受け付けアドレスは、アプリケーションに参加している他のワークステーション・クライアント・プロセスがこの WSL プロセスと通信するための手段として使用されます。*string* の形式が "0x*hex-digits*" または "\\x*hex-digits*" の場合、偶数の有効な 16 進数桁を含める必要があります。これらの形式は、TCP/IP アドレスを含む文字配列に内部変換されます。*string* の値は次のいずれかの形式で指定します。

//hostname:*port_number*

//#.#.#.#:*port_number*

最初の形式では、gethostbyname(3c) を介してアクセスされたローカル設定の名前解決機能を使ってアドレスが結合されるときに、*hostname* は TCP/IP ホスト・アドレスに解決されます。*#.#.#.#* はドットで区切った 10 進数の形式で、各 *#* は 0 から 255 までの 10 進数の値を表します。*Port_number* は 0 から 65535 までの 10 進数です。

注記　一部のポート番号は、お使いのシステムで使用されるトランスポート・プロトコル (TCP/IP など) のために予約されている場合があります。予約されているポート番号を確認するには、トランスポート・プロトコルのマニュアルを調べてください。

TA_EXT_NADDR: *string*[0..78]
WSH プロセスの既知のアドレス・テンプレートとして使用される完全ネットワーク・アドレスを指定します。このアドレスは、ワークステーション・クライアントが WSH プロセスに接続するのに使用する

既知のネットワーク・アドレスを生成するために、WHS ネットワーク・アドレスと組み合わされます。このアドレスの形式は、TA_NADDR と同じです。ただし、組み合わされたネットワーク・アドレスの位置が WSH ネットワーク・アドレスからコピーされることを示すために、ポート番号が同じ長さの文字 M で置き換えられます。たとえば、アドレス・テンプレートが 0x0002MMMMdddddddd で WSH ネットワーク・アドレスが 0x00021111ffffffff のときは、既知のネットワーク・アドレスは 0x00021111dddddddd です。アドレス・テンプレートが "//" で始まる場合、ネットワーク・アドレス・タイプは IP 対応であり、WSH ネットワーク・アドレスの TCP/IP ポート番号はアドレス・テンプレートにコピーされて、組み合わされたネットワーク・アドレスが生成されます。この機能は、ワークステーション・クライアントがネットワーク・アドレス変換を実行するルータを通じて WSH に接続したいときに役立ちます。既存の T_WSL オブジェクトの SET オペレーションの空の TA_EXT_NADDR 文字列は、TA_CLOPT 属性から -H エントリを削除します。

TA_WSHNAME: *string*[1..78]
: ワークステーション・リスナに対してワークステーション・ハンドラ・サービスを供給する実行可能ファイルの名前。デフォルトは、システム提供のワークステーション・ハンドラに対応する WSH です。ワークステーション・ハンドラは、buildwsh() コマンドを使用してカスタマイズできます。詳細については、カスタマイズの項と buildwsh(1) リファレンス・ページを参照してください。

TA_MINHANDLERS: 0 <= *num* < 256
: 任意の時間にこの WSL と共に使用できるハンドラの最小数。WSL は起動するとすぐにこれと同数の WSH を起動し、管理者が WSL をシャットダウンするまで WSH の数をこの最小数以上に維持します。稼働している WSL に対してこの属性を変更すると、新たなハンドラが起動されます。

TA_MAXHANDLERS: 0 <= *num* < 32,767
: 任意の時間にこの WSL と共に使用できるハンドラの最大数。システムにアクセスしようとしているワークステーション・クライアントの要求を満たすために、必要に応じてハンドラが起動されます。この属性は、ハンドラの最小数の値以上でなければなりません。

TA_MULTIPLEX: 1 <= *num* < 32,767
: 1 つのハンドラ・プロセスが同時にサポートするクライアントの最大数。

TA_MINENCRYPTBITS: {0 | 40 | 56 | 128}
: BEA Tuxedo システムに接続する際に必要な暗号化の最小レベルを指定します。0 は暗号化を行わないことを示し、40、56、および 128 は暗号化キーの長さをビット単位で指定します。ここで指定する最小レベルの暗号化が満たされない場合、リンクの確立は失敗します。デフォルト値は 0 です。

注記　リンク・レベルの暗号化の値 40 は、下位互換性を維持するために提供されています。

TA_MAXENCRYPTBITS: {0 | 40 | 56 | 128}
: BEA Tuxedo システムに接続する際に調整できる暗号化の最大レベルを指定します。0 は暗号化を行わないことを示し、40、56、および 128 は暗号化キーの長さをビット単位で指定します。デフォルトは 128 です。

注記　リンク・レベルの暗号化の値 40 は、下位互換性を維持するために提供されています。

TA_MINWSHPORT: 0 <= *num* < 65,535
: このリスナによって WSH プロセスに割り当てられる使用可能なポート番号の範囲の最小値です。

TA_MAXWSHPORT: 0 <= *num* < 65,535
: このリスナによって WSH プロセスに割り当てられる使用可能なポート番号の範囲の最大値です。

TA_MAXIDLETIME: 0 <= *num* < 35,204,650
: ワークステーション・クライアントがハンドラによってアプリケーションから切り離されるまでの最長アイドル時間 (分)。35,204,650 を指定した場合、クライアントはタイムアウトにならずに無制限にアイドル状態でいることができます。0 を指定した場合、クライアントは 1 秒を超えて非アクティブであれば終了します。

TA_MAXINITTIME: 1 <= *num* < 32,767
: ワークステーション・クライアントが WSH を介して初期化プロセスを完了するまでの最短時間 (秒)。この時間を過ぎると、ワークステーション・クライアントは WSL によってタイムアウトになります。

TA_CMPLIMIT: *threshold*
: ワークステーション・クライアントとの間のトラフィックを圧縮するしきい値メッセージ・サイズ。しきい値は、負以外の数値または文字列 MAXLONG です。MAXLONG は、マシンの最大長の設定に動的に変換されます。制限事項 : この属性値は、BEA Tuxedo Workstation リリース 6.1 以前のシステムを実行しているワークステーション・クライアントでは使用されません。

TA_CLOPT: *string*[0..128]
: 起動時に WSL サーバに渡されるコマンドライン・オプション。詳細については、servopts(5) リファレンス・ページを参照してください。制限事項 : 実行時にこの属性を変更しても、稼働中の WSL サーバに影響はありません。サーバ固有のオプション (ダブルダッシュ "--" の後のオプション) は設定できず、また返されません。

TA_ENVFILE: *string*[0..256] (BEA Tuxedo 8.0 以前の場合は最大 78 バイト)
: WSL サーバ固有の環境ファイル。このファイルを使用して環境を変更する方法については、T_MACHINE:TA_ENVFILE を参照してください。制限事項 : 実行時にこの属性を変更しても、稼働中の WSL サーバに影響はありません。

TA_GRACE: 0 <= *num*
: T_WSL:TA_MAXGEN 制限が適用される秒数。この属性は、再起動可能な WSL サーバに対してのみ意味を持ちます。つまり、T_WSL:TA_RESTART 属性が "Y" に設定されている場合にのみ意味があります。再起動するサーバが TA_MAXGEN 制限を超えても、TA_GRACE 時間を過ぎていれば、システムは現在の世代 (T_SERVER:TA_GENERATION) を 1 にリセットし、初期ブート時間 (T_SERVER:TA_TIMESTART) を現在の時刻にリセットします。この属性の値が 0 のときには、WSL サーバを必ず再起動する必要があります。

TA_KEEPALIVE: "{client | handler | both | none}"
クライアント、ハンドラ、またはその両方に対してネットワーク keep-alive 機能をオンにできます。"none" を指定すると、クライアントとハンドラの両方に対してこの機能をオフにできます。

この属性値の変更は、新しい接続に対してのみ適用されます。

TA_MAXGEN: 1 <= *num* < 256
指定された猶予期間 (T_WSL:TA_GRACE) に、再起動可能な WSL サーバ (T_WSL:TA_RESTART == "Y") に対して許可される世代数。WSL サーバの最初の起動が 1 世代とカウントされ、各再起動も 1 とカウントされます。最大世代数を超えた後の処理については、前述の TA_GRACE に関する説明を参照してください。

TA_NETTIMEOUT: 0 <= *num* <= MAXLONG
TA_NETTIMEOUT の値は、ワークステーション・クライアントが WSL/WSH からの応答を受信するために待機状態でいられる最短時間 (秒数) です。デフォルト値は 0、つまりタイムアウトが発生しないことを示します。

この属性値の変更は、新しい接続に対してのみ適用されます。

TA_RCMD: *string*[0..256] (BEA Tuxedo 8.0 以前の場合は最大 78 バイト)
システムによるアプリケーション・サーバの再起動と同時に実行するアプリケーション指定のコマンド。このコマンドは、ネイティブ・オペレーティング・システム内の実行可能ファイルでなければなりません。

TA_RESTART: "{Y | N}"
再起動可能 ("Y") または再起動不可能 ("N") な WSL サーバ。このサーバ・グループに対してサーバ移行が指定された場合には (T_RESOURCE:TA_OPTIONS/MIGRATE T_GROUP:TA_LMID w/ 代替サイト)、この属性を "Y" に設定する必要があります。

TA_SEQUENCE: 1 <= *num* < 10,000
このサーバを他のサーバに基づいて起動 (tmboot(1)) またはシャットダウン (tmshutdown(1)) する必要があるかを指定します。
TA_SEQUENCE 属性を指定せずに、または無効な値を指定して追加された T_WSL オブジェクトは、他の自動的に選択されたデフォルト値より大きな 10,000 以上の値を持ちます。サーバは、シーケンス番号の昇順

に tmboot() で起動され、降順に tmshutdown() でシャットダウンされます。この属性を実行時に変更した場合、その効果は tmboot() と tmshutdown() にのみ適用され、稼働中のサーバが以後の tmshutdown() の呼び出しによってシャットダウンされる順序に適用されます。

TA_CURHANDLERS: 0 <= *num*
: この WSL に関連付けられている現在アクティブなハンドラの数。

TA_HWHANDLERS: 0 <= *num*
: 任意の時点でこの WSL に関連付けられている現在アクティブなハンドラの最大数。

TA_WSPROTO: 0 <= *num*
: このワークステーション・グループに対する BEA Tuxedo Workstation プロトコル・バージョン番号。このグループに接続している /WS クライアントは、その /WS クライアントに関連付けられた異なるプロトコル・バージョン番号を持っている可能性があることに注意してください。

TA_SUSPENDED: "{NEW | ALL | NONE}"
: 値 "NEW" は、新たに接続してくるクライアントがこの WSL オブジェクトを介して接続しない可能性があることを示しています。値 "ALL" は、新しいクライアントの接続の禁止に加えて、この WSL を介して既にアプリケーションに接続されているワークステーション・クライアントが SUSPENDED の状態にある (TM_MIB(5) を参照) ことを示しています。値 "NONE" は、有効な中断特性がないことを示しています。

TA_VIEWREFRESH:Y
: Y の値を設定すると、ワークステーション・グループ内のすべてのアクティブ WSH がその VIEW バッファ・タイプ・キャッシュをリフレッシュします。

制限事項 このクラスは T_SERVER クラスの特殊な場合を表すクラスなので、対応する T_SERVER オブジェクトでも重複して定義されている特定の属性を表します。T_SERVER クラスに含まれる属性のうちここに記載されていないものにはこのクラスを介してアクセスする必要があり、T_WSL クラスからアクセスすることはできません。

WS_MIB(5) に関する追加情報

診断　WS_MIB(5) への接続時には、2 つの一般的なタイプのエラーがユーザに返される場合があります。1 つは、管理要求に対する応答を検索する 3 つの ATMI 関数 (tpcall()、tpgetrply()、および tpdequeue()) が返すエラーです。これらのエラーは、該当するリファレンス・ページの記述に従って解釈されます。

ただし、要求がその内容に対応できるシステム・サービスに正常にルーティングされても、システム・サービス側でその要求を処理できないと判断されると、アプリケーション・レベルのサービス障害としてエラーが返されます。このような場合、tpcall() と tpgetrply() は、tperrno を TPESVCFAIL に設定してエラーを返し、以下のようにエラーの詳細を示す TA_ERROR、TA_STATUS、および TA_BADFLD フィールドと一緒に、元の要求を含む応答メッセージを返します。TMQFORWARD(5) サーバ経由でシステムに転送された要求に対してサービス障害が発生すると、元の要求で識別される異常終了キューに障害応答メッセージが追加されます (TMQFORWARD に対して -d オプションが指定されたと見なされる)。

管理要求の処理中にサービス・エラーが発生すると、TA_STATUS という FM32 フィールドにエラーの内容を説明したテキストが設定され、TA_ERROR という FM32 フィールドにはエラーの原因 (下記参照) を示す値が設定されます。以下のエラー・コードは、いずれもマイナスであることが保証されています。

[other]
すべてのコンポーネント MIB に共通のその他のエラー・リターン・コードは、MIB(5) リファレンス・ページに指定されています。これらのエラー・コードは、ここに定義する WS_MIB(5) 固有のエラー・コードと相互に排他関係にあることが保証されています。

以下の診断コードは TA_ERROR で戻されるもので、管理要求が正常に完了したことを示します。これらのコードはマイナスでないことが保証されています。

[other]
すべてのコンポーネント MIB に共通のその他のリターン・コードは、MIB(5) リファレンス・ページに指定されています。これらのコード

は、ここに定義する WS_MIB(5) 固有のリターン・コードと相互に排他関係にあることが保証されています。

相互運用性 このリファレンス・ページで定義されているヘッダ・ファイルおよびフィールド・テーブルは、BEA Tuxedo リリース 5.0 以降で利用できます。これらのヘッダやテーブルで定義するフィールドはリリースが変わっても変更されません。ただし、以前のリリースで定義されていない新しいフィールドが追加される場合があります。AdminAPI には、要求を作成するために必要なヘッダ・ファイルとフィールド・テーブルがあれば、どのサイトからでもアクセスできます。T_WSL および T_WSH クラスは、BEA Tuxedo システム・リリース 6.0 で新しく追加されたクラスです。そのため、AdminAPI を介して旧リリースのサイト上で WSL プロセスと WSH プロセスをローカル管理することはできません。ただし、このリファレンス・ページに定義された管理機能の多くは、リリース 6.0 のサイトと相互運用していれば、リリース 6.0 より前のサイトでも使用できます。リリースが異なるサイト (共にリリース 6.0 以降) を相互運用する場合、当該リリースの MIB マニュアル・ページに定義されるように、旧サイト上の情報はアクセスおよび更新可能で、以降のリリースで利用可能な情報のサブセットとなります。

移植性 BEA Tuxedo システムの MIB を使用した管理作業をサポートするために必要な既存の FML32 および ATMI 関数、さらにこのリファレンス・ページに定義するヘッダ・ファイルとフィールド・テーブルは、すべてのサポート対象ネイティブ・プラットフォームとワークステーション・プラットフォームで使用可能です。

使用例 以下の例は、TM_MIB(5) と WS_MIB(5) を組み合わせてワークステーション・グループを順番に非アクティブ化するコードです。

フィールド・テーブル 属性フィールド識別子にアクセスするには、フィールド・テーブル tpadm が必要です。そのためには、次のようにシェルで入力します。

```
$ FIELDTBLS=tpadm
$ FLDTBLDIR=${TUXDIR}/udataobj
$ export FIELDTBLS FLDTBLDIR
```

ヘッダ・ファイル 次のヘッダ・ファイルがインクルードされます。

```
#include <atmi.h>
#include <fml32.h>
#include <tpadm.h>
```

ワークステーション・グループの中断

以下のコードは、ワークステーション・グループの状態を SUSpended に設定します。このように設定すると、ワークステーション・グループはワークステーション・クライアントからの新しい接続を受け入れることができなくなり、現在グループの一部であるすべてのワークステーション・クライアントが一時停止します。このコードと後続のコードは、対象としているワークステーション・グループを識別するためのローカル変数 ta_srvgrp と ta_srvid が既に設定されているという前提で書かれています。

```
/* 入力バッファと出力バッファの割り当て */ ibuf = tpalloc("FML32", NULL,
1000);
 obuf = tpalloc("FML32", NULL, 1000);
 /* 要求タイプを定義する MIB(5) 属性を設定 */
 Fchg32(ibuf, TA_OPERATION, 0, "SET", 0);
 Fchg32(ibuf, TA_CLASS, 0, "T_WSL", 0);
 /* WS_MIB(5) 属性を設定 */
 Fchg32(ibuf, TA_SRVGRP, 0, ta_srvgrp, 0);
 Fchg32(ibuf, TA_SRVID, 0, (char *)ta_srvid, 0);
 Fchg32(ibuf, TA_SUSPENDED, 0, "ALL", 0);
 /* 要求を作成 */
 if (tpcall(".TMIB", (char *)ibuf, 0, (char **)obuf, olen, 0) 0) {
 fprintf(stderr, "tpcall failed: %s\en", tpstrerror(tperrno));
 if (tperrno == TPESVCFAIL) {
 Fget32(obuf, TA_ERROR, 0,(char *)ta_error, NULL);
 ta_status = Ffind32(obuf, TA_STATUS, 0, NULL);
 fprintf(stderr, "Failure: %ld, %s\en",
 ta_error, ta_status);
 }
 /* 追加のエラー処理 */
 }
 /* 今後使用するために論理マシン識別子をコピー */
 strcpy(ta_lmid, Ffind32(obuf, TA_LMID, 0, NULL));
```

WSH オブジェクトのリストの取得

既存の入力バッファを使って、クラスと操作を変更して新しい要求を作成します。特定の T_WSL オブジェクトのキー・フィールド、ta_srvgrp と ta_srvid に関連付けられているすべての T_WSH オブジェクトを検索します。検索を効率化するため、TA_FILTER 属性を設定します。

```
/* 要求タイプを定義する MIB(5) 属性を設定 */ Fchg32(ibuf, TA_CLASS, 0,
"T_WSH", 0);
 Fchg32(ibuf, TA_OPERATION, 0, "GET", 0);
 longval = TA_WSHCLIENTID;
 Fchg32(ibuf, TA_FILTER, 0, (char *)longval, 0);
 /* WS_MIB(5) 属性を設定 */
 Fchg32(ibuf, TA_LMID, 0, ta_lmid, 0);
 /* 別の出力バッファを割り当てて TA_WSHCLIENTID 値を保存 */
```

```
wshcltids = tpalloc("FML32", NULL, 1000);
/* 要求を作成 */
tpcall(".TMIB", (char *)ibuf, 0, (char **)wshcltids, olen, 0);
/* 取得数を確認 */
Fget32(wshcltids, TA_OCCURS, 0,(char *)wshcltcnt, NULL);
```

T_CLIENT オブジェクトの取得

検索した TA_WSHCLIENTID 値を使用して、このワークステーション・グループ内のワークステーション・クライアントの関連付けられた TA_CLIENTID 値のリストを検索します。

```
/* 要求バッファを初期化 */ Finit32(ibuf, Fsizeof32(ibuf));
/* 要求タイプを定義する MIB(5) 属性を設定 */
Fchg32(ibuf, TA_OPERATION, 0, "GET", 0);
Fchg32(ibuf, TA_CLASS, 0, "T_CLIENT", 0);
longval = TA_CLIENTID;
Fchg32(ibuf, TA_FILTER, 0, (char *)longval, 0);
longval = TA_WSHCLIENTID;
Fchg32(ibuf, TA_FILTER, 1, (char *)longval, 0);
/* WS_MIB(5) 属性を設定 */
Fchg32(ibuf, TA_LMID, 0, ta_lmid, 0);
Fchg32(ibuf, TA_WSC, 0, "Y", 0);
if (wshcltcnt == 1) {
/* 1 だけなので、それをキー・フィールドとして使用 */
Fchg32(ibuf, TA_WSHCLIENTID, 0,
Ffind32(wshcltids, TA_WSHCLIENTID, 0, NULL));
}
/* 出力バッファを割り当てて TA_CLIENTID/TA_WSHCLIENTID 値を保存 */
cltids = tpalloc("FML32", NULL, 1000);
/* 要求を作成 */
tpcall(".TMIB", (char *)ibuf, 0, (char **)cltids, olen, 0);
/* 取得数を確認 */
Fget32(cltids, TA_OCCURS, 0,(char *)cltcnt, NULL);
/* 必要な場合、関連付けられていないクライアントを削除 */
if (wshcltcnt > 1) {
for (i=(cltcnt-1); i >= 0 ;i--) {
p = Ffind32(cltids, TA_WSHCLIENTID, i, NULL);
for (j=0; j wshcltcnt ;j++) {
q = Ffind32(wshcltids, TA_WSHCLIENTID, j, NULL);
if (strcmp(p, q) == 0) {
break; /* このクライアントは現在のグループに所属 */
}
}
if (j >= wshcltcnt) {
/* クライアントが見つからないのでリストから削除 */
Fdel32(cltids, TA_CLIENTID, i);
Fdel32(cltids, TA_WSHCLIENTID, i);
cltcnt--;
```

```
 }
 }
 }
```

T_CLIENT オブジェクトへの通知

検索した TA_CLIENTID 値を使用して、このワークステーション・グループ内のワークステーション・クライアントにログオフを通知します。

```
notstr = tpalloc("STRING", NULL, 100);
(void)strcpy(notstr, "Please logoff now!");

/* 関係するクライアントをループ処理して一時停止 */
for (i=0; i cltcnt ;i++) {
  p = Ffind32(cltids, TA_CLIENTID, i, NULL);

  /* クライアントにログオフを通知 */
  tpconvert(p, (char *)ci, TPCONVCLTID);
  tpnotify(ci, notptr, 0, 0);
}
```

T_CLIENT オブジェクトの非活性化

検索した TA_CLIENTID 値を使用して、このワークステーション・グループ内の残りのワークステーション・クライアントを非活性化します。既に非活性化されているクライアントは SET でエラーを返しますが、これは無視します。

```
/* 要求バッファを初期化 */
Finit32(ibuf, Fsizeof32(ibuf));
 /* 要求タイプを定義する MIB(5) 属性を設定 */
Fchg32(ibuf, TA_OPERATION, 0, "SET", 0);
Fchg32(ibuf, TA_CLASS, 0, "T_CLIENT", 0);
Fchg32(ibuf, TA_STATE, 0, "DEAd", 0);

/* 関連するクライアントをループ処理して非活性化 */
for (i=0; i cltcnt ;i++) {
  p = Ffind32(cltids, TA_CLIENTID, i, NULL);
  Fchg32(ibuf, TA_CLIENTID, 0, p);

 /* 要求を作成 */
  tpcall(".TMIB", (char *)ibuf, 0, (char **)obuf, olen, 0);
}
```

T_WSL オブジェクトの非活性化

T_WSL オブジェクトを非活性化します。これにより、関連付けられたアクティブな T_WSH オブジェクトが自動的に非アクティブ化されます。

```
/* 要求タイプを定義する MIB(5) 属性を設定 */
Fchg32(ibuf, TA_OPERATION, 0, "SET", 0);
Fchg32(ibuf, TA_CLASS, 0, "T_WSL", 0);
Fchg32(ibuf, TA_STATE, 0, "INActive", 0);
```

```
 /* WS_MIB(5) 属性を設定 */
Fchg32(ibuf, TA_SRVGRP, 0, ta_srvgrp, 0);
Fchg32(ibuf, TA_SRVID, 0, (char *)ta_srvid, 0);

 /* 要求を作成 */
tpcall(".TMIB", (char *)ibuf, 0, (char **)obuf, olen, 0);
}
```

ファイル ${TUXDIR}/include/tpadm.h, ${TUXDIR}/udataobj/tpadm

関連項目 tpacall(3c)、tpalloc(3c)、tpcall(3c)、tpdequeue(3c)、tpenqueue(3c)、tpgetrply(3c)、tprealloc(3c)、FML 関数の紹介、Fadd、Fadd32(3fml)、Fchg、Fchg32(3fml)、Ffind、Ffind32(3fml)、MIB(5)、TM_MIB(5)

『BEA Tuxedo アプリケーションの設定』

『BEA Tuxedo アプリケーション実行時の管理』

『C 言語を使用した BEA Tuxedo アプリケーションのプログラミング』

『FML を使用した BEA Tuxedo アプリケーションのプログラミング』

WSL(5)

名前 WSL— ワークステーション・リスナ・サーバ

形式

```
WSL SRVGRP="identifier"
  SRVID="number"
  CLOPT="[-A] [servopts options] -- -n netaddr [-d device]
      [-w WSHname] [-t timeout-factor] [-T Client-timeout]
      [-m minh] [-M maxh] [-x mpx-factor]
      [-p minwshport] [-P maxwshport] [-I init-timeout]
      [-c compression-threshold] [-k compression-threshold]
      [-K {client|handler|both|none}]
     [-z bits] [-Z bits] [-H external-netaddr][-N network-timeout]"
```

機能説明 ワークステーション・リスナは、ワークステーション・クライアントからネイティブ・サービスへのアクセスを可能にする BEA Tuxedo システム提供のサーバです。アプリケーション管理者は、SERVERS セクションでワークステーション・リスナ・サーバをアプリケーション・サーバとして指定することにより、ワークステーションからアプリケーションへのアクセスを許可します。ワークステーション・リスナとワークステーション・ハンドラの処理を指定するには、関連するコマンド行オプションを使用します。

位置指定、サーバ・グループ、サーバ ID、その他の汎用サーバ関連パラメータは、サーバ用に定義されているコンフィギュレーション・ファイル機構を使用してワークステーション・リスナに関連付けられます。ワークステーション・リスナ専用のコマンド行オプションを使用して、カスタマイズすることもできます。

アプリケーションの一部として WSL が起動するたびに、単一の既知のネットワーク・アドレスから、ワークステーション上で作業するユーザのための代替クライアントとして機能する一群のワークステーション・ハンドラ (WSH) へアクセスすることを認めることにより、多数のワークステーション・クライアントがアプリケーションにアクセスできるようになります。WSH は、アプリケーション・ワークステーションからのワークロードに対応するために、必要に応じて WSL によって動的に起動および停止されます。アプリケーション管理者に対する利点としては、少数のネイティブ・サイト・プロセス (WSH) によって、多数のクライアントをサポートすることができるため、ネイティブ・サイトでのプロセス数を減らすことができるとい

う点、また、ネイティブ・サイトが、ワークステーション・サイトにある掲示板上の情報を維持するためのオーバーヘッドを負う必要がなくなるという点が挙げられます。

以下の WSL 固有のコマンド行オプションを使用できます。これらのコマンド行オプションは、CLOPT パラメータの二重ダッシュ (--) の後ろに指定できます。

-n *netaddr*

WSL プロセスが接続指示受け付けアドレスとして使用する完全なネットワーク・アドレスを指定します。これは必須パラメータです。

WSL の接続指示受け付けアドレスは、アプリケーションに参加している他のワークステーション・クライアント・プロセスがこの WSL プロセスと通信するための手段として使用されます。*netaddr* (1 ～ 78 文字) の形式が "0x*hex-digits*" または "\\x*hex-digits*" の場合、有効な偶数桁の 16 進数を含める必要があります。これらの形式は、TCP/IP アドレスを含む文字配列に内部変換されます。このアドレスは、次の 2 つの形式のいずれかで指定することもできます。

//hostname:port_number
//#.#.#.#:port_number

#.#.#.# はドットで区切った 10 進数の形式で、各 *#* は 0 から 255 までの 10 進数の値を表します。*port_number* は 0 から 65535 までの 10 進数です。

注記　一部のポート番号は、お使いのシステムで使用されるトランスポート・プロトコル (TCP/IP など) のために予約されている場合があります。予約されているポート番号を確認するには、トランスポート・プロトコルのマニュアルを調べてください。

[-d *device*]

ワークステーション・リスナおよびワークステーション・ハンドラによるネットワークへのアクセスの際に使用されるデバイス・ファイル名です。リリース 6.4 以降では、このパラメータは省略できます。デフォルトはありません。

[-w *WSHname*]

ワークステーション・リスナに対してワークステーション・ハンドラ・サービスを供給する実行可能ファイルの名前。デフォルトは、システム提供のワークステーション・ハンドラに対応する WSH です。

ワークステーション・ハンドラは、buildwsh() コマンドを使用してカスタマイズできます。詳細については、buildwsh(1) リファレンス・ページを参照してください。

[-t *timeout-factor*]

このオプションは -I オプションに置き換えられ、BEA Tuxedo リリース 6.0 での上位互換性のためにサポートされます。今後のリリースでは削除されることもあります。WSL によってタイムアウトとされる前に、WSH を通じて初期化処理を完了するためにワークステーション・クライアントに対して認められるべき秒単位の時間を、SCANUNIT の値と乗算することによって指定するためのパラメータです。このパラメータのデフォルトは、セキュリティなしのアプリケーションでは 3 秒、セキュリティありのアプリケーションでは 6 秒です。有効範囲は 1 ～ 255 です。

[-T *client-timeout*]

Client-timeout は、クライアントがアイドル状態を保持できる時間を分単位で表したものです。クライアントがこの時間内に要求を行わなかった場合、WSH はクライアント接続を切断します。このオプションは、安定性の低いクライアントのプラットフォームでの使用に適しています (ユーザが tpterm() の呼び出しを行わずにコンピュータの電源を切る可能性がある場合など)。また、クライアントが任意通知型メッセージの通知を受信しても再試行しない場合にも効果があります。-T を指定しない場合、タイムアウトはありません。

[-m *minh*]

任意の時間にこの WSL と共に使用できるハンドラの最小数。WSL は起動するとすぐにこれと同数の WSH を起動し、管理者が WSL をシャットダウンするまで WSH の数をこの最小数以上に維持します。このパラメータのデフォルト値は 0 で、有効な範囲は 0 ～ 255 です。

[-M *maxh*]

任意の時間にこの WSL と共に使用できるハンドラの最大数。システムにアクセスしようとしているワークステーション・クライアントの要求を満たすために、必要に応じてハンドラが起動されます。このパラメータのデフォルト値は、論理マシン上の MAXWSCLIENTS の設定値を WSL に対する多重化係数 (下記 -x オプションを参照) で除してから切り上げた値と等しくなります。このパラメータの有効範囲は 1 ～ 4,096 です。この値は *minh* 以上でなければなりません。

[-x *mpx-factor*]
各ワークステーション・ハンドラの範囲内で必要となる多重化の程度を制御するために使用するパラメータです。このパラメータの値は、各ワークステーション・ハンドラによって同時にサポート可能なワークステーション・クライアントの数を示しています。ワークステーション・リスナにより、新たなワークステーション・クライアントの処理が必要なときに新たなハンドラが起動します。この値は 1 ～ 4,096 でなければなりません。デフォルト値は 10 です。

[-p *minwshport*]
[-P *maxwshport*]

この一対のコマンド行オプションを使用すると、このリスナ・サーバに関連付けられている WSH が使用できるポート番号の範囲を指定できます。ポート番号は 0 ～ 65535 の範囲で指定します。デフォルト値は、*minwshport* が 2048、*maxwshport* が 65535 です。

注記　一部のポート番号は、お使いのシステムで使用されるトランスポート・プロトコル (TCP/IP など) のために予約されている場合があります。予約されているポート番号を確認するには、トランスポート・プロトコルのマニュアルを調べてください。

[-I *init-timeout*]
このオプションは -t オプションに取って代わるもので、クライアントの初期化タイムアウト間隔を設定するために使用します。ワークステーション・クライアントが WSH を介して初期化プロセスを完了するまでの時間 (秒) を指定します。この時間を過ぎると、ワークステーション・クライアントは WSL によってタイムアウトになります。このパラメータのデフォルト値は 60 で、有効な範囲は 1 ～ 32,767 です。

[-c *compression-threshold*]
このオプションは、ワークステーション・クライアントとハンドラが使用する圧縮しきい値を決定します。ワークステーション・クライアントとハンドラの間で送信されるバッファは、指定値よりも大きいときにはすべて圧縮されます。このパラメータのデフォルト値は 2,147,483,647 であり、許容範囲が 0 から 2,147,483,647 なので圧縮されないということを意味します。

[-k *compression-threshold*]
これは、USL France または ITI のクライアントの BEA Tuxedo リリース 6.2 より前のバージョン向けの特別な圧縮オプションです。この条件が適用される場合は、-c オプションで圧縮しきい値を制御する WSL/WSH のペアと、-k オプションで制御する WSL/WSH のペアが複数存在できます。-k の機能は -c と同じです。

[-K {client | handler | both | none}]
-K オプションは、client、handler、または both に対してネットワーク keep-alive 機能をオンにできます。none を指定すると、クライアントとハンドラの両方に対してこの機能をオフにできます。

[-z [0 | 40 | 56 | 128]]
このオプションは、ワークステーション・クライアントとワークステーション・ハンドラ間でネットワーク・リンクを確立する際に必要な暗号化の最小レベルを指定します。0 は暗号化が行われないことを示し、40、56、および 128 は暗号化キーの長さをビット単位で指定します。ここで指定する最小レベルの暗号化が満たされない場合、リンクの確立は失敗します。デフォルトは 0 です。このオプションは、国際版または米国 / カナダ版の BEA Tuxedo セキュリティ・アド・オン・パッケージがインストールされている場合にのみ使用できます。

注記　リンク・レベルの暗号化の値 40 は、下位互換性を維持するために提供されています。

[-Z [0 | 40 | 56 | 128]]
このオプションは、ワークステーション・クライアントとワークステーション・ハンドラ間でネットワーク・リンクを確立する際に必要な暗号化の最大レベルを指定します。0 は暗号化が行われないことを示し、40、56、および 128 は暗号化キーの長さをビット単位で指定します。デフォルトは 128 です。このオプションは、国際版または米国 / カナダ版の BEA Tuxedo セキュリティ・アド・オン・パッケージがインストールされている場合にのみ使用できます。

注記　リンク・レベルの暗号化の値 40 は、下位互換性を維持するために提供されています。

[-H *external-netaddr*]
WSH プロセスの既知のアドレス・テンプレートとして使用される完全ネットワーク・アドレスを指定します。このアドレスは、ワークス

テーション・クライアントが WSH プロセスに接続するのに使用する既知のネットワーク・アドレスを生成するために、WHS ネットワーク・アドレスと組み合わされます。このアドレスの形式は、-n オプションと同じです。ただし、組み合わされたネットワーク・アドレスの位置が WSH ネットワーク・アドレスからコピーされることを示すために、ポート番号が同じ長さの文字 M で置き換えられます。たとえば、アドレス・テンプレートが 0x0002MMMMddddddddd で WSH ネットワーク・アドレスが 0x00021111ffffffff のときは、既知のネットワーク・アドレスは 0x00021111dddddddd です。アドレス・テンプレートが "//" で始まる場合、ネットワーク・アドレス・タイプは IP 対応であり、WSH ネットワーク・アドレスの TCP/IP ポート番号はアドレス・テンプレートにコピーされて、組み合わされたネットワーク・アドレスが生成されます。この機能は、ワークステーション・クライアントがネットワーク・アドレス変換を実行するルータを通じて WSH に接続したいときに役立ちます。

[-N *network-timeout*]

このオプションでは、ワークステーション・クライアントがネットワークからデータを受信する際に Tuxedo 操作を待機状態にできる期間を秒単位で指定できます。この期間を超えると、操作は失敗してクライアントはアプリケーションから切断されます。値 0 (デフォルト) はタイムアウトが発生しないことを示します。注意：この設定値が低すぎると、切断が発生しすぎる可能性があります。

MAXWSCLIENTS の値がゼロの場合など、WSL がワークステーション・クライアントをサポートするのを妨げるコンフィギュレーションでは、起動時に WSL が異常終了します。

移植性 WSL は、サポートされているすべてのサーバ・プラットフォームで BEA Tuxedo システム提供のサーバとしてサポートされます。

相互運用性 WSL は相互運用するアプリケーションで実行できますが、BEA Tuxedo リリース 4.2 以降のノードで実行する必要があります。

使用例

```
*SERVERS
WSL SRVGRP="WSLGRP" SRVID=1000 RESTART=Y GRACE=0
  CLOPT="-A -- -n 0x0002ffffaaaaaaaa -d /dev/tcp"
WSL SRVGRP="WSLGRP" SRVID=1001 RESTART=Y GRACE=0
  CLOPT="-A -- -n 0x0002aaaaffffffff -d /dev/tcp -H 0x0002MMMMddddddddd"
```

```
WSL SRVGRP="WSLGRP" SRVID=1002 RESTART=Y GRACE=0
  CLOPT="-A -- -n //hostname:aaaa -d /dev/tcp -H //external_hostname:MMMM"
```

関連項目　buildwsh(1)、servopts(5)、UBBCONFIG(5)

『BEA Tuxedo アプリケーションの設定』

『BEA Tuxedo アプリケーション実行時の管理』

『C 言語を使用した BEA Tuxedo アプリケーションのプログラミング』