BEA Tuxedo

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

BEA Tuxedo リリース 8.0J
8.0 版
2001 年 10 月

Copyright

Restricted Rights Legend

Trademarks or Service Marks

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

Document Edition	Date	Software Version
8.0J	2001 年 10 月	BEA Tuxedo リリース 8.0J

目次

このマニュアルについて

このマニュアルでは、BEA Tuxedo システムで使用するファイル形式、データ記述方法、管理情報ベース (MIB)、およびシステム・プロセスのリファレンス情報について説明します。リファレンス・ページでは、ファイル形式、データ記述、MIB、または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作成および管理する管理者
- 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、Web サイト上にあります。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 Documents 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 をお持ちではない場合は、Adobe Web サイト (http://www.adobe.co.jp/) から無償で入手できます。

関連情報

関連情報は各リファレンス・ページの「関連項目」に一覧表示されています。MIB については、クラスごとにではなく MIB 全体についての関連情報が示されます。

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 8.0 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、**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`*

規則	項目
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。
[ ]	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

BEA 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)	テキスト形式のドメイン・コンフィギュレーション・ファイル
GWTOPEND(5) の DMCONFIG	TOP END Domain Gateway 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
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)	イベント・ブローカの管理情報ベース
factory_finder.ini	ドメイン用の FactoryFinder 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

BEA 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
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 のゲートウェイ・サーバ
langinfo(5)	言語情報定数
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 コア・システムの管理情報ベース
TMFFNAME	FactoryFinder および NameManager サービスを実行するサーバ
TMIFRSVR	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
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)	実行時トレーシング機能
TMUSREVT(5)	ユーザ・イベント通知プロセス
tperrno(5)	BEA Tuxedo システムのエラー・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 システムにより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

BEA 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
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・リスナ・サーバ

テーブルとファイルの紹介

機能説明

このセクションでは、各種のテーブルとファイルの形式について説明します。

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、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 管理情報ベースは、MIB(5) のリファレンス・ページ、および以下に示す MIB ページに説明されています。

- ACL_MIB(5)
- APPQ_MIB(5)
- DM_MIB(5)
- EVENT_MIB(5)
- 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更新を行う前に、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なければなりません。管理要求をフォーマットしたり管理応答を解釈する場合は、ACL_MIB(5) を共通の MIB リファレンス・ページ MIB(5) と組み合わせて用います。このマニュアルに記載されたクラスや属性を利用して MIB(5) の手順に従ってフォーマットした要求は、アクティブなアプリケーション中に存在するさまざまな ATMI インターフェイスを利用して管理サービスを要求するために使用できます。ACL_MIB(5) のすべてのクラス定義に関連する追加情報については、12 ページ「ACL_MIB(5) に関する追加情報」を参照してください。

ACL_MIB(5) には、次のクラスがあります。

ACL_MIB **のクラス**

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、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た属性の詳細な説明

属性表
　　クラスの各属性に関する名前、タイプ、パーミッション、値、およびデフォルト値を示す表。属性表の形式を以下に示します。

属性の意味
　　各属性の意味を説明します。

制限事項
　　このクラスにアクセスしたり、このクラスを解釈する場合の制限事項。

属性表の形式

上記で説明したように、この MIB に含まれる各クラスは以下の 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をリストし、さらに管理者、オペレータ、一般ユーザが属性を使用してアプリケーションとインターフェイスをとる方法を示しています。属性表の各属性記述には 5 つの構成要素 (名前、タイプ、パーミッション、値、デフォルト値) があります。各項目については MIB(5) で説明します。

TA_FLAG 値

MIB(5) は共通の TA_FLAGS 属性を定義します。long 型で、共通およびコンポーネントの MIB 固有フラグ値の両方が含まれます。ここでは、ACL_MIB(5) 固有フラグ値は定義していません。

FML32 フィールド・テーブル

このマニュアル・ページに記述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、アプリケーションによって FLDTBLDIR 環境変数で指定されるコロンで区切ったリストに含まれ、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で区切ったリストに含まれている必要があります。

制限事項

この MIB のヘッダ・ファイルやフィールド・テーブルには、BEA Tuxedo リリース 6.0 以降のサイト (ネイティブおよびワークステーションの両方) で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

ACL_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 (\\n) は使用できません。

TA_GROUPID:0 <= *num* < 16,384

ユーザに対応するグループ識別子。0 はデフォルト・グループ "other" を示します。作成時に省略すると、次に利用可能な（一意な）識別子で、0 以上の識別子がデフォルト値となります。

TA_STATE:

GET:{VALid}

GET 操作は、選択した T_ACLGROUP オブジェクトについてコンフィギュレーション情報を検索します。下に示した状態は、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。このクラスで有効な状態は VALid 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 オブジェクトに対するコンフィギュレーション情報を更新します。次に、SET 要求で TA_STATE に設定できる値の意味を説明します。記述されていない状態を設定することはできません。

NEW	アプリケーションに対する T_ACLGROUP 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。INValid 状態ではこの組み合わせは使用できません。正常終了すると、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_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 グループだけに関連付けることができます。複数の役割を持つユーザや複数のグループに対応付けられるユーザの場合、複数のユーザ・エントリを定義しなければなりません。

T_ACLPERM クラスの定義

概要

T_ACLPERM クラスは BEA Tuxedo システム・エンティティにアクセス可能な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。エンティティ名は現在のサービス名、イベント名、およびアプリケーション・キュー名を示します。

属性表

ACL_MIB(5): T_ACLPERM 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_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。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 (\\n) 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*

対応エンティティへのアクセスを許可するグループ識別子 (番号) をカンマで区切ったリスト。*string* の長さは、マシンのディスク容量でのみ制限されます。

TA_STATE:

GET:{VALid}

GET 操作は、選択した T_ACLPERM オブジェクトについてコンフィギュレーション情報を検索します。下に示した状態は、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。このクラスでは、有効な状態は VALid だけです。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 オブジェクトに対するコンフィギュレーション情報を更新します。次に、SET 要求で TA_STATE に設定できる値の意味を説明します。記述されていない状態を設定することはできません。

NEW	アプリケーションに対する T_ACLPERM 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。INValid 状態ではこの組み合わせは使用できません。正常終了すると、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_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に対してではなく、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やドメイン、およびそれに対応す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結合するには、ユーザ固有パスワードを提供する必要があります。

属性表

ACL_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]	"*"

ACL_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_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 操作には 1 つ以上必要

属性の意味

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 (#)、カンマ (,)、コロン (:)、および改行文字 (\n) は使用できません。

TA_PRINCLTNAME:*string*
ユーザに対応するクライアント名。一般的に、対応ユーザの役割を記述し、ユーザ・エントリに関する付加的な情報となります。作成時に省略すると、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 (:) や改行文字 (\\n) は使用できません。

TA_PRINID:1 <= *num* < 131,072
一意なユーザ識別番号。作成時に省略すると、次に利用可能な (一意な) 識別子で、0 以上の識別子がデフォルト値となります。

TA_PRINGRP:0 <= *num* < 16,384
ユーザに対応するグループ識別子。0 はデフォルト・グループ "other" を示します。作成時に省略すると、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 オブジェクトに対するコンフィギュレーション情報を検索します。下に示した状態は、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。このクラスでは、有効な状態は VALid だけです。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 オブジェクトに対するコンフィギュレーション情報を更新します。次に、SET 要求で TA_STATE に設定できる値の意味を説明します。記述されていない状態を設定することはできません。

NEW	アプリケーションに対する T_ACLPRINCIPAL 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。INValid 状態ではこの組み合わせは使用できません。正常終了すると、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_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 つのユーザまたはドメインはただ 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や複数のグループに関連付けられるユーザの場合、複数のプリンシパル・エントリを定義しなければなりません。

ACL_MIB(5) に関する追加情報

診断

一般に、ACL_MIB(5) とのインターフェイスでは、2 つの一般的なタイプのエラーがユーザに返されます。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()、tpdequeue()) で、いずれかが各関数で定義されたエラーを返します。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() および 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、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) 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ます)。

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、いずれも負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 に戻され、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・ページに記載されています。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 クラス、T_ACLGROUP クラス、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、さらに本マニュアル・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・プラットフォームおよびワークステーション・プラットフォームで利用可能です。

使用例

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以下に示します。

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ためには、フィールド・テーブル *tpadm* が必要です。そのために、以下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

以下のヘッダ・ファイルがあります。

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

ユーザの追加

以下のコードはデフォルト・グループ "other" にユーザを追加します。

```
/* 入力バッファおよび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) の属性を設定 *
 Fchg32(ibuf, TA_OPERATION, 0, "SET", 0);
 Fchg32(ibuf, TA_CLASS, 0, "T_ACLPRINCIPAL", 0);

 /* ACL_MIB(5) 属性を設定 */
 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。

APPQ_MIB(5) は、管理要求をフォーマットし、管理応答を解釈するために共通の MIB のリファレンス・ページ MIB(5) と組み合わせて使用できます。MIB(5) で説明されているように、要求はこのリファレンス・ページで解説されているクラスと属性を使用してフォーマットされます。この要求は、アクティブなアプリケーションに存在する多数の ATMI インターフェイスのどれかを使用して管理サービスを要求するのに使用されます。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も管理できます。APPQ_MIB(5) のすべてのクラス定義に関連する追加情報については、44 ページ「APPQ_MIB(5) に関する追加情報」を参照してください。

APPQ_MIB(5) は以下のクラスから構成されます。

APPQ_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、アプリケーションが定義している永続的 (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) なキュー (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、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た属性の詳細な説明

属性表
: クラスの各属性に関する名前、タイプ、パーミッション、値、およびデフォルト値を示す表。属性表の形式を以下に示します。

属性の意味
: 各属性の意味を説明します。

制限事項
: このクラスにアクセスしたり、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 の一部を構成している各クラスは、4 つのパートで記録されます。1 つは属性表です。属性表はクラス内の属性をリストし、さらに管理者、オペレータ、一般ユーザが属性を使用してアプリケーションとインターフェイスをとる方法を示しています。

属性表の各属性記述には 5 つの構成要素 (名前、タイプ、パーミッション、値、デフォルト値) があります。各項目については MIB(5) で説明します。

TA_FLAG 値

MIB(5) は、共通およびコンポーネントの MIB 固有のフラグ値を持った long の共通 TA_FLAGS 属性を定義します。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 フラグと組み合わせて使用します。

QMIB_FORCECLOSE
: T_APPQSPACE オブジェクトの TA_STATE 属性を CLEaning に設定する際に、このフラグは、キューの状態がたとえ ACTive であっても状態の変更が成功することを示します。

QMIB_FORCEDELETE
: T_APPQSPACE オブジェクトの TA_STATE 属性を INValid に設定する際に、このフラグは、キュー・スペースが ACTive な場合でも、そのキュー・スペースのどれかにメッセージが存在している場合でも、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 に設定する際に、キューにメッセージがある場合でも、キュー・スペースにプロセスがアタッチされている場合でも、このフラグによってキューを削除することができます。

QMIB_FORCEPURGE
: T_APPQ オブジェクトの TA_STATE 属性を INValid に設定する際に、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。しかし、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現在かかわっていると、状態の変更は失敗し、エラーをユーザ・ログに書込みます。

FML32 フィールド・テーブル

このリファレンス・ページに記述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、またフィールド・テーブル名 tpadm は FIELDTBLS 環境変数で指定されるカンマで区切ったリストに、アプリケーションによって指定されなければなりません。

制限事項

この MIB は、リリース 6.0 以降の BEA Tuxedo を使用しているサイトでのネイティブおよび /WS だけに提供されます。

リリース 6.0 以前の BEA Tuxedo リリースを動作させているサイトがアプリケーションでアクティブな場合は、この MIB による管理アクセスは以下のように制限されます。

- SET 操作は不可能です。
- リリース 6.0 以前のサイトのローカル情報アクセスは利用できません。

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 つ以上のアプリケーション・キューがあります。

制限事項

すべてのキー・フィールドを未設定にすると、このクラスのすべてのインスタンスを検索できません。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・フィールドを指定しなければなりません。必要なキー・フィールドは、TA_APPQSPACENAME、TA_QMCONFIG、TA_LMID です。ただし、アプリケーションの環境が未設定 (TUXCONFIG 環境変数が未設定) の場合を除きます。環境が未設定の場合は、TA_LMID を省略しなければなりません。たとえば、tpcall() を用いて、要求に 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・スペースのすべての T_APPQ オブジェクトが検索されます。

属性表

APPQ_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

APPQ_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_LMID(k)(r)(*)[b]	string	ru-r--r--	*string*[1..30]	N/A
TA_STATE (Note 3)[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
TA_DEFDELIVERYPOLICY	string	rw-r--r--	{PERSIST | NONPERSIST}	PERSIST
TA_CMD	string	rw-r--r--	*shell-command-string*[0..78]	""
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*[0-78]	""
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

APPQ_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_CURNONPERSISTMSG	long	r--r--r--	0 <= *num*	N/A

(k) — GET キー・フィールド [e]
(r) — オブジェクト作成に必要なフィールド
(*) — 必要な 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属性です。
[b] アプリケーションのコンフィギュレーションが設定されていない (TUXCONFIG 環境変数が未設定) 場合以外、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 オブジェクトに対するすべての操作 (GET、SET の両方) は、対応するキュー・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 に設定されます。キュー・スペースが大きいと、操作に時間がかかります。
[d] アプリケーション・キューを作成した後は、TA_APPQORDER を変更できません。
[e] 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 操作の際に適切なキー・フィールドを設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・キュー名。

TA_APPQSPACENAME:*string*[1..15]
アプリケーション・キューがあるアプリケーションキュー・スペース名。

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、選択され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 要求に対して返される TA_STATE の意味を示します。

VALid	指定されたキューが存在します。この状態は INActive と同等で、パーミッションの確認に使用されます。

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、新しいキューを作成します。以下に、SET 要求で返される TA_STATE の意味を示します。リストにない状態は設定されない場合があります。

NEW	指定したキュー・スペースに新規にキューを作成します。正常に作成されると、キューは VALid 状態となります。
INValid	指定されたキューを削除します。削除するには、キューが VALid 状態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 フラグが設定されない限り、キューは削除されません。さらに、キューにメッセージが追加されていると、QMIB_FORCEPURGE を指定しないと、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。正常終了すると、オブジェクト状態は変更されません。

TA_APPQORDER:

キュー内のメッセージを処理する順序。正当な値は、PRIO、TIME、または EXPIR です。ソート基準の組み合わせは、最上位基準を最初に指定し、ほかの基準、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。キュー作成時に順序を指定しないと、デフォルト値は 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..78]

デイスク・ベースの一時的メッセージの上限値 TA_CMDHW に達すると、コマンドが自動的に実行されます。下限値 TA_CMDLW に達した後に、再度、上限値になると、コマンドが再実行されます。

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

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

TA_CMD 属性で指定されたコマンドの自動実行を制御する上限値および下限値。どちらも 0 以上の整数です。TA_CMDHW および TA_CMDLW の後には以下のキー文字が続き、キー文字は TA_CMDHW と TA_CMDLW に一致する必要があります。

b

上限値および下限値を、キューの永続的メッセージ (デイスク・ベースのメッセージ) で使用するバイト数で設定します。

B

上限値および下限値を、キュー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

上限値および下限値を、キューの永続的メッセージおよび一時的メッセージの数で設定します。

%

上限値および下限値を、キュー容量のパーセンテージで設定します。これには永続的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、キューが 50 メッセージまで減少した後に再度 100 メッセージまで増加しないと、再実行されません。

TA_CMDNONPERSIST:*shell-command-string*[0..78]

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、コマンドが自動的に実行されるように指定します。一時的メッセージの下限値 TA_CMDNONPERSISTLW に達した後に、再度上限値になると、コマンドが再実行されます。

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および下限値を指定します。どちらも 0 以上の整数で、以下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

上限値および下限値は、キューの一時的 (メモリ内の) メッセージで使用されるバイト数で表されます。

B

上限値および下限値は、キューの一時的 (メモリ内の) メッセージで使用されるブロック数で表されます。

%

上限値および下限値は、キューで使用される、キュー・スペース内で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されます。

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、キューの永続的メッセージおよび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できません。

TA_CURBLOCKS:0 <= *num*

現在、キューによって使用されているディスク・ページ数。

TA_CURMSG:0 <= *num*

現在、キューにある永続的メッセージ数。キューにあるメッセージの合計数を求めるには、この値に TA_CURMEMMSG を加えます。

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、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たり管理インターフェイスを介して削除されないと、そのメッセージと関連す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。メッセージがトランザクション内にあるときに期限切れになった場合、それによってトランザクションが異常終了することはありません。トランザクション内でキューへの登録中、またはキューからの取り出し中に期限切れになったメッセージは、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。メッセージが期限切れになったという通知はありません。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あっ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 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方針を指定します。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) 配信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) 配信方法で配信されます。キューのデフォルトの配信方針を変更しても、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。変更するキューが、現在キュー・スペースにあるメッセージに対する名前付きの応答キューである場合、キューのデフォルトの配信方針を変更しても、メッセージに対するサービスの応答品質は変わりません。

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、メッセージの登録操作は異常終了します。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・スペースに対して T_APPQSPACE クラスの TA_MEMNONPERSIST 属性が 0 の場合、一時的メッセージ用の領域は確保されません。そのような場合には、一時的メッセージを登録しようとしても常に異常終了します。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。デフォルトは 0 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)。デフォルトは 0 です。

TA_CURNONPERSISTBYTES:0 <= *num*

この属性は、キュー上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を指定します。

TA_CURNONPERSISTMSG:0 <= *num*

この属性は、現在キューにある一時的メッセージ数を指定します。キューの合計メッセージ数を求めるには、この値に TA_CURMSG を加えます。

T_APPQMSG クラスの定義

概要

T_APPQMSG クラスは、アプリケーション・キュー内に保存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() 呼び出しの結果作成されます。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、メッセージの特定属性は管理者が変更できます。たとえば、管理者はメッセージを同一キュー・スペース内のあるキューから別のキューに移動させたり、優先順位を変更することができます。

制限事項

すべてのキー・フィールドを未設定にすると、このクラスのすべてのインスタンスを検索できません。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・フィールドを指定しなければなりません。必要なキー・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未設定 (TUXCONFIG 環境変数が未設定) の場合を除きます。環境が未設定の場合は、TA_LMID を省略しなければなりません。たとえば、tpcall() を用いて、要求に TA_APPQSPACENAME、TA_QMCONFIG、および TA_LMID 属性を設定した場合、指定キュー・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

APPQ_MIB(5): APPQ_MIB(5): T_APPQMSG **クラス定義** : **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

APPQ_MIB(5): APPQ_MIB(5): T_APPQMSG **クラス定義** : **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
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]
(*) — 必要な 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属性です。

[b] アプリケーションのコンフィギュレーションが設定されていない (TUXCONFIG 環境変数が未設定) 場合を除いて、TA_LMID をキー・フィールドとして指定しなければなりません。
[c] T_APPQMSG オブジェクトに対する操作はすべて (GET、SET の両方とも) 対応するキュー・スペースを自動的にオープンします。キュー・スペースの状態が OPEn または ACTive 以外の場合、暗黙的に OPEn に設定します。キュー・スペースが大きい場合、この操作には大変時間がかかります。
[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 操作の際に適切なキー・フィールドを設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 操作に対して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]
メッセージを保存するアプリケーション・キュー名。

TA_APPQSPACENAME:*string*[1..15]
メッセージのあるアプリケーション・キュー・スペース名。

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 要求に対して返される 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、パーミッションの確認に使用されます。

SET:{INValid}
SET 操作は、選択したメッセージの特性を更新します。以下に、SET 要求で返される TA_STATE の意味を示します。リストにない状態は設定されない場合があります。

INValid	メッセージがキュー・スペースから削除されます。この操作を行う場合、メッセージは VALid 状態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。正常終了すると、オブジェクトの状態は変更されません。

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、キューに残っていたり、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。メッセージがトランザクション内にあるときに期限切れになった場合、それによってトランザクションが異常終了することはありません。トランザクション内でキューへの登録中、またはキューからの取り出し中に期限切れになったメッセージは、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。メッセージが期限切れになったという通知はありません。

  期限切れ時間の値を変更する責任を持つキュー・マネージャではメッセージの期限切れを利用できる場合でも、この機能を利用できない BEA Tuxedo システムのバージョンの場合、その BEA Tuxedo システムでキューに登録されるメッセージに期限切れ時間を追加することはできません。期限切れ時間を追加しようとすると、失敗します。

  期限切れ時間が設定されない場合、GET 操作は空文字列を返します。TA_EXPIRETIME には以下のいずれかの形式が使用されます。

  +*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 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たり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。絶対期限切れ時間は、キュー・マネージャ・プロセスが存在するマシンの時計によって決まります。

NONE
メッセージが決して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 オブジェクトの発生を検索する最高 / 最低優先順位。これらの属性は GET 操作のキー・フィールドとしてしか使用できません。

TA_MSGEXPIRESTARTTIME:
TA_MSGEXPIREENDTIME:
T_APPQMSG オブジェクトの発生を検索する期限切れ開始 / 終了時間。範囲には各値を含みます。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。これらの属性は GET 操作のキー・フィールドとしてしか使用できません。

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

TA_MSGSTARTTIME:
TA_MSGENDTIME:
T_APPQMSG オブジェクトの発生を検索する開始 / 終了時間。範囲には各値を含みます。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。これらの属性は GET 操作のキー・フィールドとしてしか使用できません。

TA_NEWAPPQNAME:*string*[1..15]
選択したメッセージの移動先のキュー名。このキューは同一キュー・スペースに存在するキューでなければなりません。この操作を行うには、メッセージの状態は 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方針によって影響されません。期限切れ時間のあるメッセージを移動する場合、その期限切れ時間が相対期限切れ時間として指定されていても、新しいキューでは絶対期限切れ時間として扱われます。

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:
メッセージの応答に対するサービス配信の品質。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、また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方針を使用するときは DEFAULT に設定されます。

デフォルトの配信方針は、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まります。つまり、元のメッセージがキューに登録されてからそのメッセージに対する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方針が変更された場合は、応答が最後に登録された時点で有効な方針が使用されます。

TA_TIME:
メッセージを処理する時間。以下のいずれかの形式が使用されます。

+seconds
メッセージを *seconds* 秒後に処理することを指定します。0 を指定すると、メッセージはすぐに処理されます。

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 ～ 37 までの年は 2000 ～ 20037、70 ～ 99 は 1970 ～ 1999 として処理されます。38 ～ 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の詳細は、TM_MIB(5) の T_DEVICE クラスを参照してください。各キュー・スペースには通常 1 つ以上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保存されます。

キュー・スペースは、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)、デバイスが存在する論理マシン (TA_LMID 属性) の複数の属性により一意に識別されます。

キュー・スペースは通常、環境設定済みのアプリケーションでは 1 つのサーバグループにしか関連付けられません。キュー・スペース名およびデバイス名は、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、このクラスのすべてのインスタンスを検索できません。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、すべてのキー・フィールドを指定しなければなりません。環境設定がされていない (TUXCONFIG 環境変数が未設定) 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・フィールドは省略しなければなりません。

/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 オブジェクト、T_APPQMSG オブジェクト、T_APPQTRANS オブジェクトにも該当します。

属性表

APPQ_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
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

APPQ_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
(*) — 必要な 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属性です。
[b] アプリケーションの環境設定がされていない (TUXCONFIG 環境変数が未設定) 場合を除いて、TA_LMID をキー・フィールドとして指定しなければなりません。

[c] T_APPQ オブジェクト、T_APPQMSG オブジェクト、および T_APPQTRANS オブジェクトに対する操作はすべて (GET、SET の両方とも) 対応するキュー・スペースを自動的にオープンします。キュー・スペースの状態が OPEn または ACTive 以外の場合、暗黙的に OPEn に設定します。キュー・スペースが大きい場合、この操作には大変時間がかかります。
[d] キュー・スペース作成後、TA_MAXQUEUES は変更不可能となります。

属性の意味

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

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 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。以下に、GET 要求に対して返される 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・スペースに対するディスク領域を初期化中です。この状態は ACTive と同等で、パーミッション検査に使用されます。
OPEn	キュー・スペースに対する共有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、現在共有メモリには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、パーミッションの確認に使用されます。

ACTive	キュー・スペースに対する共有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。共有メモリには現在最低1つのプロセスがアタッチされています。プロセスはキュー・スペースに対応するキュー・サーバ(TMS_QM、TMQUEUE、および、おそらく TMQFORWARD)、qmadmin(1) などの管理プロセス、あるいは他のアプリケーションに関連付けられたプロセスとなります。

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、または新規に作成します。以下に、SET 要求で返される TA_STATE の意味を示します。リストにない状態は設定できません。

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省略時値は 16。

TA_CURACTIONS:0 <= *num*
: この属性は、キュー・スペースで使用している現在のアクション数を指定します。この値は、キュー・スペースが OPEn または ACTive の場合、あるいはキュー・スペースが新規に作成された場合のみ決定されます。どの条件も当てはまらない場合は、-1 の値が返されます。

TA_CURCURSORS:0 <= *num*
: この属性は、キュー・スペースで使用している現在のカーソル数を指定します。この値は、キュー・スペースが OPEn または ACTive の場合、あるいはキュー・スペースが新規に作成された場合のみ決定されます。どの条件も当てはまらない場合は、-1 の値が返されます。

TA_CUREXTENT:0 <= *num* <= 100
: キュー・スペースが使用す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・スペースで使用する現在のハンドル数を指定します。この値は、キュー・スペースが OPEn または ACTive の場合、あるいはキュー・スペースが新規に作成された場合のみ決定されます。どの条件も当てはまらない場合は、-1 の値が返されます。

NEW	新規にキュー・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有メモリおよび他の IPC リソースを割り当て、初期化します。キュー・スペースが INActive 状態の場合のみ可能です。
CLEaning	キュー・スペースに対する共有メモリおよび他の IPC リソースを削除します。キュー・スペースが OPEn または ACTive 状態の場合のみ可能です。状態が ACTive の場合は、QMIB_FORCECLOSE フラグが指定されていなければなりません。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、ある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 フラグを渡さない限り、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。正常終了の場合は、オブジェクト状態は変更されません。

TA_CURMEMFILTERS:0 <= *num*

どの条件も当てはまらない場合は、-1 の値が返されます。この値は、キュー・スペースが OPEn または ACTive の場合、あるいはキュー・スペースが新規に作成された場合のみ決定されます。どの条件も当てはまらない場合は、-1 の値が返されます。

TA_CURMEMNONPERSIST:0 <= *num*

キュー・スペース内の一時的メッセージが使用する現在のメモリ容量 (バイト)。この値は、キュー・スペースが OPEn または ACTive の場合、あるいはキュー・スペースが新規に作成された場合のみ決定されます。どの条件も当てはまらない場合は、-1 の値が返されます。

TA_CURMEMOVERFLOW:0 <= *num*
: この属性は、キュー・スペース内のオーバフロー・メモリで使用する現在のバイト数を指定します。この値は、キュー・スペースが OPEn または ACTive の場合、あるいはキュー・スペースが新規に作成された場合のみ決定されます。どの条件も当てはまらない場合は、-1 の値が返されます。

TA_CURMSG:0 <= *num*
: キュー・スペース内の現在のメッセージ数。この値は、キュー・スペースが OPEn または ACTive の場合、あるいはキュー・スペースが新規に作成された場合のみ決定されます。どの条件も当てはまらない場合は、-1 の値が返されます。

TA_CUROWNERS:0 <= *num*
: この属性は、キュー・スペースで使用する現在の所有者数を指定します。この値は、キュー・スペースが OPEn または ACTive の場合、あるいはキュー・スペースが新規に作成された場合のみ決定されます。どの条件も当てはまらない場合は、-1 の値が返されます。

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数。

TA_CURQUEUES:0 <= *num*
: キュー・スペースに現在存在しているキュー数。この値は、キュー・スペースが OPEn または ACTive の場合、あるいはキュー・スペースが新規に作成された場合のみ決定されます。どの条件も当てはまらない場合は、-1 の値が返されます。

TA_CURTMPQUEUES:0 <= *num*
: この属性は、キュー・スペースで使用する現在の一時キューの数を指定します。この値は、キュー・スペースが OPEn または ACTive の場合、あるいはキュー・スペースが新規に作成された場合のみ決定されます。どの条件も当てはまらない場合は、-1 の値が返されます。

TA_CURTRANS:0 <= *num*
: キュー・スペースを使用する現在の未終了トランザクション数。

TA_ERRORQNAME:*string*[0..15]
: キュー・スペースに対応するエラー・キュー名。エラー・キューがない場合、GET 要求は空き文字列を返します。

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・タイプに応じて (通常ファイルまたは raw スライスなど)、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、同時に到達した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た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た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した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、一時的メッセージが使用した最大メモリ容量 (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・メモリで使用された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た後、ある時点でキュー・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・スペースで同時に到達した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のキュー・スペース内の最大プロセス数。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た後、ある時点でキュー・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、キュー・スペースで同時にオープンされた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た後、ある時点でキュー・スペースを使用する最大未処理トランザクション数。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 環境変数で示す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・スペースの共有メモリにアクセスする際に使用する 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。ブロッキング操作が発生したときに追加アクションが利用可能な場合、ブロッキング操作はその条件を満たすまで無視されます。ブロッキング操作が無視されると、別の操作要求を処理できるようになります。ブロッキング操作が完了すると、その操作に関連付けられていたアクションをその後の操作で利用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できるプロセス数と等しいアクションを確保しているため、各キュー・マネージャ・プロセスは最低 1 つのブロッキング・アクションを持つことができます。管理者は、システムが確保するブロッキング・アクション数を超えて、追加のブロッキング・アクションを処理するようにシステムを設定できます。ブロッキング操作が要求された場合にすぐにその条件を満たすことができず、ほかに利用できるアクションもない場合は、操作は異常終了します。

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。カーソルはキューを操作するのに使用します。カーソルが削除されると、そのカーソルのリソースはその後のカーソル生成操作で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、同時に割り当てられる最大カーソル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しようとしてカーソルのリソースを利用できない場合、操作は異常終了します。BEA Tuxedo アプリケーションではこの値を調節する必要はありません。この値を調節しても、共用メモリを必要以上に浪費するだけで、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、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・ハンドルがクローズされると、そのハンドルはその後のオブジェクトのオープン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。ユーザがキューイング・サービス・オブジェクトをオープンしようとした場合にハンドルを利用できないと、操作は異常終了します。この値を調節しても、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。ユーザに対してオープンしているハンドル数に関係なく、各ユーザに 1 つの所有者レコードがあります。オープンしているハンドルがないユーザの場合、その所有者レコードはその後のユーザに利用できます。システムはアクション数と等しい所有者を確保し、異なる所有者がそれぞれのアクションを開始できるようにします。管理者は、システムが確保したキューイング・サービス・リソースを同時に使用できる所有者数を超えて、追加の所有者に対応するようにシステムを設定できます。オープンしているハンドルがない場合に、ユーザがハンドルをオープンしようとして利用できる所有者がないと、操作は異常終了します。この値を調節しても、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、新規にエクステントが割り当てられます (TA_CUREXTENT 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によって使用され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、動的自己設定型アプリケーションで使用されます。一時キューに登録されるメッセージは、永続的ではありません。一時キューのすべてのハンドルがクローズされると、一時キューのリソースはその後の一時キュー作成に利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、管理者は、同時にアクティブにな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オープンしようとした場合に一時キューのリソースを利用できないと、操作は異常終了します。この値を調節しても、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。メモリのサイズはバイトで指定します。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。フィルタが削除されると、そのフィルタによって使用されていたメモリはその後のコンパイル済みフィルタに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、同時にコンパイルされ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しようとした場合にコンパイルされるフィルタに十分なメモリが割り当てられていないと、操作は異常終了します。この値を調節しても、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、共用メモリで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、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・サイズが 1024 バイト (1KB) と仮定すると、値 2000b が要求された場合メモリの割り当ては 1 ページ (1024 バイト) になり、値 2048b が要求された場合メモリの割り当ては 2 ページ (2048 バイト) になります。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、値 50B を要求した場合、50 ページのメモリが割り当てられます。

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は、完全に失われます。

キュー・スペースに対して TA_MEMNONPERSIST が 0 の場合、一時的メッセージのための領域は確保されていません。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、常に異常終了します。たとえば、メッセージに対してサービス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、メッセージの登録操作は異常終了します。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、共用メモリで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がクローズまたは削除されると、メモリは解放されてその後のオーバフロー状況に利用できるようになります。追加のメモリ領域を利用すると、設定より多くのオブジェクトを作成できますが、すべてのオブジェクトに常に追加メモリを利用できるとは限りません。現在、オーバフロー・メモリを使用するのは、アクション、ハンドル、カーソル、所有者、一時キュー、タイマ、およびフィルタだけです。

TA_MEMSYSTEMRESERVED:0 <= *num*

この属性は、キユーイング・サービス・システムが使用するために、共用メモリで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、すべてのキユーイング・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・スペースに対して初期化された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・キューに対応す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、このクラスのすべてのインスタンスを検索できません。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、すべてのキー・フィールドを指定しなければなりません。たとえば、TA_XID を除くすべてのキー・フィールドを tpcall() を使用して要求に設定した場合、指定キュー・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対応しないので注意して下さい。トランザクションは実際に他のアプリケーションに属さない、あるは影響を持たないため、トランザクションにコミットしたり、トランザクションをアボートする場合は注意してください。TA_XID 属性の値はアプリケーション間で一意とは限りません。

属性表

APPQ_MIB(5): APPQ_MIB(5): T_APPQTRANS **クラス定義** : **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}" SET:"{HAB | HCO}"	N/A N/A

(k) — GET キー・フィールド [c]
(*) — 必要な 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属性です。
[b] T_APPQTRANS オブジェクト—GET および SET—対応するキュー・スペースを自動的にオープンします。キュー・スペースの状態が OPEn または ACTive 以外の場合、暗黙的に OPEn に設定します。キュー・スペースが大きい場合、この操作には大変時間がかかります。

[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 操作の際に適切なキー・フィールドを設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対応するアプリケーション・キュー・スペース名。

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の情報を検索します。以下に、GET 要求に対して返される TA_STATE の意味を示します。状態はすべて ACTive と同等で、パーミッション検査に使用されます。

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ように識別されています。
ABorteD	トランザクションはロールバックされるように識別されており、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 要求で返される TA_STATE の意味を示します。リストにない状態は設定できません。

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。正常終了の場合は、オブジェクトは HABort 状態になります。
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。正常終了の場合は、オブジェクトは HCOmmit 状態になります。

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 FLM32 関数および ATMI 関数、そして本マニュアル・ページに記述するヘッダ・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・プラットフォームおよびワークステーション・プラットフォームで利用可能です。

相互運用性

この MIB は、リリース 6.0 以降の BEA Tuxedo を使用しているサイトでのネイティブおよびワークステーションだけに提供されます。

リリース 6.0 以前の BEA Tuxedo システムのリリースを動作させているサイトがアプリケーションでアクティブな場合は、本 MIB による管理アクセスは以下のように制限されます。

- SET 操作は不可能です。
- リリース 6.0 以前のサイトのローカル情報アクセスは利用できません。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) が相互運用を行う場合、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に利用可能で、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、以下のように、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後、各コードの後には、以下のような、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 が環境中で利用可能でなければなりません。そのために、以下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

以下のヘッダ・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 を使用するときは、ライブラリを手動でリンクしなければなりません。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、同じバッファを再度初期化する */
Finit32(rqbuf, (FLDLEN) Fsizeof32(rqbuf));
```

```
/* キュー・スペースを作成する要求を作成 */
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

以下のコードは、上の例で作成したキュー・スペースに新規に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)。

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想定して要求を作成 */
rval = tpcall(".TMIB", rqbuf, 0, rpbuf, rplen, flags);

/* 各 TMS_QM グループに対して、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・スペース名を決める 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 および 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する ) */
```

```
  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 (つまり、TA_APPQSPACENAME、TA_QMCONFIG、および TA_LMID) の優先順位がわかっていないと検索できません。

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"

機能説明

AUTHSVR は、BEA Tuxedo が提供する、認証サービスを備えたサーバです。クライアントがアプリケーションに結合する際には、高セキュリティ・アプリケーションでこのサービスを使用することにより、各ユーザごとに認証することができます。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されたパスワードと比較することにより、要求が妥当かをチェックします。要求が妥当であると認められた場合は正常に終了し、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。

tpreturn(3c) の rcode パラメータがアプリケーション・キーの設定に使用されます。このパラメータは、妥当性検査が正常に終了するかパーミッションが与えられない時点で、tpinit(3c) を呼び出したコードに (tpurcode で) 返されます。

AUTHSVR に関連する追加情報については、53 ページ「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認証サービス (AUTHSVC) を指定します。

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

AUTHSVC パラメータを指定しないと、認証サービスはデフォルトで AUTHSVC になります。

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*" オプションを使用して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、パスワード・ファイルで一致するデータが 1 つ見つかれば、それ以降のデータに対してはチェックされません。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することができます。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 や 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。そうでないと、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 (システムの管理者) と tpsysop (システム・オペレータ) は、AUTHSVR(5) によって特別な方法で処理されます。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 が返すアプリケーション・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 ${TUXDIR}/bin/AUTHSVR に入れた状態で出荷され、上記で説明したセマンティクスを持っています。ソース・コードのサンプルは、${TUXDIR}/lib/AUTHSVR.c に入っています。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 および 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 (この値でなければなりません)。

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は、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 パラメータ設定は、SECURITY が ACL か MANDATORY_ACL に設定されている場合に AUTHSVR によって宣言される認証サービス (..AUTHSVC) を指定します。

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

AUTHSVC パラメータを指定しないと、認証サービスはデフォルトで ..AUTHSVC になります。

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 とし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 つのインスタンスを実行し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。ユーザは 1 つのエントリしかユーザ・ファイルに持てず、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()、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利用して管理することができます。

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 (システムの管理者) と tpsysop (システム・オペレータ) は、AUTHSVR(5) によって特別な方法で処理されます。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 が返すアプリケーション・キーは、下位の 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リザーブされています)。tpsysadm および 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 (この値でなければなりません)。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 に入っている標準仕様の AUTHSVR を使用する必要があります。

AUTHSVR に関する追加情報

使用方法

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、なるべく ${APPDIR} にインストールするようにしてください。

移植性

AUTHSVR は、BEA Tuxedo が提供するサーバとして非 /WS 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・モジュールを完成し、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、以下の作業を行うことが必要です。

- ソース・ファイルをコンパイルする
- 実行可能ファイルを必要なライブラリを使用してリンクする

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() がそれぞれ用意されています。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。

リンクするには buildclient または buildserver を実行する必要がありますが、システムではより柔軟なコンパイル方法が提供されています。必要に応じて、自分で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指定します。

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 system 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し、エクスポートします。

TUXDIR
: BEA Tuxedo system 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・ログのファイル名の前に付ける接頭辞。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— コロンで区切られた FIELDTBLS ファイルを検索する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)
■ 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(1)、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	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) マニュアル・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> ( アプリケーションが必要とする場合 )
#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は、代わりに次のコマンドを使用してください。

```
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。ここに示した通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場合は、次のようにコマンドを実行します。

```
viewc view_file
```

view_file は、VIEW 用のソース記述を格納している 1 つ以上のファイルです。

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 を使用することができます。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 を使用することができます。

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
: コロンで区切った、FIELDTBLS ファイルを検索する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
: コロンで区切った、FIELDTBLS ファイルを検索する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 ファイルの 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 のインスタンスは 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" に設定します)。

SERVERS セクションでは、DMADM サーバに対するサーバ・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 環境変数を、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、リリース 5.0 以降の BEA Tuxedo システム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 ファイル内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提供するには、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・プロセスの詳細および 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)、GWTOPEND(5) の DMCONFIG、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 ドメイン機能を利用して、通信およびサービスの共有が可能な 2 つ以上のドメイン (またはアプリケーション) のセットです。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、各ドメインのドメイン・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 ファイルと呼ばれます (環境変数によって、実際に使用されたファイル名は保持されます)。

DMCONFIG ファイルは以下を定義します。

- ローカル・ドメインが通信できるリモート・ドメイン
- リモート・ドメインがアクセス可能なローカル資源 (サービスやキューなど)
- ローカル・ドメインがアクセス可能なリモート資源
- どのゲートウェイを通じて、どのローカル資源およびリモート資源にアクセス可能か

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。dmadmin(1) コマンドは、BDMCONFIG (またはそのコピー) を使用して、実行時アプリケーションを監視します。

ドメイン機能を使用するマルチ・ドメイン・コンフィギュレーションで、各ドメインに 1 つの BDMCONFIG ファイルが必要です。

DMCONFIG ファイルと BDMCONFIG ファイルの関係は、BEA Tuxedo アプリケーショ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。

DMCONFIG ファイル全体に関連する追加情報については、81 ページ「DMCONFIG(5) に関する追加情報」を参照してください。

定義

BEA Tuxedo システムにおけるドメイン・アプリケーションは、環境の定義として単一の TUXCONFIG ファイル内で記述されます。BEA Tuxedo システムのアプリケーションは、ドメイン・ゲートウェイ・グループを介して、別の BEA Tuxedo システムのアプリケーションや別の TP アプリケーションと通信できます。「BEA Tuxedo システム・ドメイン」の用語では、アプリケーションと TP ドメインは同義です。

ゲートウェイ・グループは、特定のタイプの TP ドメインと共に通信サービスを提供するドメイン・ゲートウェイ・プロセスの集合です。

ドメイン・ゲートウェイは、別の TP ドメインとの間で要求と応答を中継する BEA Tuxedo システムのドメイン・プロセスです。

ローカル・ドメインは、ほかのドメインからアクセス可能なアプリケーションの一部 (サービスのセットまたはサブセット) です。ローカル・ドメインは、常にドメイン・ゲートウェイ・グループとして表されるため、これらの 2 つの用語は同義語として使用されます。

リモート・ドメインは、ゲートウェイ・グループを介してアクセスされるリモート・アプリケーションです。リモート・アプリケーションは、ほかの BEA Tuxedo システムのドメイン・アプリケーションまたはほかの TP システムで動作するアプリケーションになります。

リモート・サービスは、アプリケーションがゲートウェイ・グループを介してアクセスするリモート・ドメインのサービスです。

ローカル・サービスは、リモート・ドメインがゲートウェイ・グループを介してアクセスするローカル・ドメインのサービスです。

コンフィギュレーション・ファイルのフォーマット

ドメイン・コンフィギュレーション・ファイルの形式は、次のとおりです。

このファイルには 8 つの指定セクションがあります。使用可能なセクションは、DM_LOCAL_DOMAINS、DM_REMOTE_DOMAINS、DM_LOCAL_SERVICES、DM_REMOTE_SERVICES、DM_RESOURCES、DM_ROUTING、DM_ACCESS_CONTROL、および DM_domtype です。*domtype* は OSITP、OSITPX、SNAX、TDOMAIN、または TOPEND のいずれかになります。DM_LOCAL_DOMAINS セクションは、DM_REMOTE_DOMAINS セクションの前になければなりません。

注記 このリファレンス・ページでは、TDOMAIN ドメインの設定方法のみを説明します。OSITP、OSITPX または SNAX ドメインの設定方法については、BEA eLink マニュアルを参照してください。TOP END ドメイン・ゲートウェイの設定方法については、GWTOPEND(5) の DMCONFIG および『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。

パラメータは通常、*KEYWORD* = *value* という形式で指定します。等号 (=) のいずれの側にでも、空白類 (空白またはタブ文字) を使用できます。この形式で *KEYWORD* が *value* に設定されます。有効なキーワードは、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。

デフォルト :[*KEYWORD1* = *value1* [*KEYWORD2* = *value2* [...]]]

この行で設定した値は、ほかの 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 (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 ドメイン機能に既知の文字列値) の場合、一般的に標準 C 規則が使用されます。標準 C *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。識別子に使用できる最大文字数は 30 文字です (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

リリース 7.1 で、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、クラスおよび属性の新しい用語を使用して、ローカル・ドメインとリモート・ドメイン間の相互作用を説明しています。新しい用語は以前のドメイン用語より正確になっていますが、ドメイン関連のマニュアルとエラー・メッセージで新しい用語が使用されているのはこのリリースだけです。新しい用語は、DM_MIB クラス、リファレンス・ページ、エラー・メッセージ、DMCONFIG ファイル構文、および各種 DMCONFIG エラー・メッセージで使用されています。

下位互換性のため、リリース 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。このリリースでは、DMCONFIG は両方の用語を使用できます。詳細については、DM_MIB(5) リファレンス・ページの「Domains 関連の新しい用語」を参照してください。

DM_LOCAL_DOMAINS セクション

このセクションでは、ローカル・ドメインと関連するゲートウェイ・グループを識別します。このセクションには、各ゲートウェイ・グループ (ローカル・ドメイン) 用のエントリが必要です。各エントリでは、グループで実行中のドメイン・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。

エントリの形式は次のとおりです。

LDOM *required_parameters* [*optional_parameters*]

LDOM は、ローカル・ドメインの名前として使用する *identifier* です。*LDOM* は、特定のコンフィギュレーション内で一意な値でなければなりません。DM_LOCAL_SERVICES セクションで説明するとおり、*LDOM* 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識別子です。

以下は、必須パラメータです。

GWGRP = *identifier*
: このローカル・ドメインを表すゲートウェイ・サーバ・グループの名前 (TUXCONFIG ファイルで指定される名前) を指定します。*DOMAINID* (下記参照) とゲートウェイ・サーバ・グループ名は、1 対 1 の関係です。

TYPE = *identifier*
: このパラメータを使用して、複数のローカル・ドメインをグループ化します。TYPE には、TDOMAIN、SNAX、OSITP、または TOPEND のいずれかを指定できます。TDOMAIN は、このローカル・ドメインがほかの BEA Tuxedo システム・ドメインとのみ通信できることを示します。SNAX は、このローカル・ドメインが SNA プロトコルを介してほかの TP ドメインと通信できることを示します。OSITP または OSITPX は、このローカル・ドメインが OSI TP プロトコルを介してほかの TP ドメインと通信できることを示します。TOPEND は、このローカル・ドメインが BEA TOP END システムとのみ通信できることを示します。ドメインのタイプは、$TUXDIR/udataobj/DMTYPE ファイルで指定する必要があります。

  注記 OSITPX は、BEA OSI TP 4.0 またはそれ以降のバージョンのゲートウェイであり、BEA Tuxedo 8.0 以降に接続する場合に必ず使用します。

DOMAINID = *string*
: このパラメータを使用して、ローカル・ドメインを識別します。DOMAINID は、ローカル・ドメインとリモート・ドメインの両方に渡り、一意である必要があります。*string* の値には、一連の文字 ("BA.CENTRAL01" など) か、"0x" で始まる 16 進数 ("0x0002FF98C0000B9D6" など) を指定できます。DOMAINID は、30 以下のオクテット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)。

オプション・パラメータは、ドメイン・ゲートウェイの操作で使用する資源と最大値 / 最小値を指定します。

AUDITLOG = *string*
: このローカル・ドメインに対する動作記録ログ・ファイルの名前を指定します。動作記録ログ機能は、dmadmin(1) コマンドによって起動し、このローカル・ドメインで行われるすべての動作を記録します。動作記録ログ機能がオンになっている場合に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 キーワードで指定されるディレクトリに、DMmmddyy.LOG ファイル (mm = 月、dd = 日、yy = 年) が作成されます。

BLOCKTIME = *numeric*
: 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 ファイルで指定した SCANUNIT パラメータの乗数を設定します。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 ファイルで指定した BLOCKTIME パラメータの値がデフォルトとして使用されます。タイムアウトの発生は、関連する要求が失敗したことを示します。トランザクション内で要求が発行されると、TUXCONFIG ファイルでトランザクションに対して指定したタイムアウトが必ず使用されます。

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、リモート・ドメインへの接続を確立する際に、このローカル・ドメインを確認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を実行している TDOMAIN ドメイン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・ドメインの DOMAINID 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・ドメインの CONNECTION_PRINCIPAL_NAME パラメータに値を割り当てる場合、このローカル・ドメインの DOMAINID パラメータに割り当てられる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。

  エラー :Unable to acquire credentials.

CONNECTION_POLICY = *string*
: ローカル・ドメイン・ゲートウェイがリモート・ドメインに対して接続を試行するときの条件を指定します。指定可能な値は、ON_DEMAND、ON_STARTUP、INCOMING_ONLY のいずれかです。

接続方針が 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、または管理コマンド "connect" が実行されたとき、のいずれかの場合に接続が試行されます。CONNECTION_POLICY のデフォルト設定は ON_DEMAND です。接続方針が ON_DEMAND の場合は、以前のリリース (CONNECTION_POLICY 機能が提供されていない状態) と同等の機能が働きます。つまり、接続方針が ON_DEMAND として設定されていると、再接続は行われません。

接続方針が 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みます。CONNECTION_POLICY が ON_STARTUP に設定されていると、リモート・ドメインへの接続が確立された場合にのみリモート・サービス (ローカル・ドメイン・ゲートウェイによって宣言されたサービス) は宣言されます。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、60 秒おきに、失敗した接続が再試行されるよう設定されています。再接続の間隔は変更することもできます (MAXRETRY および RETRY_INTERVAL を参照)。

接続方針が 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。そのため、初期のリモート・サービス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、ローカル・ドメイン・ゲートウェイが接続を受信したときに宣言されます。INCOMING_ONLY が接続方針として設定されていると、再接続は行われません (ドメイン・タイプが OSI または SNA の場合、CONNECTION_POLICY パラメータは適用されません)。

DMTLOGDEV = *string*

このマシンのドメイン・トランザクション・ログ (DMTLOG) を含む BEA Tuxedo ファイル・システムを指定します。DMTLOG 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ます。このパラメータが指定されないと、ドメイン・ゲートウェイ・グループはトランザクション・モードで要求を処理できません。同じマシン上で動作しているローカル・ドメイン間で DMTLOGDEV 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・ドメインは、DMTLOGNAME キーワード (下記参照) 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 = *identifier*

このドメインに対するドメイン・トランザクション・ログの名前を指定します。複数のローカル・ドメイン間で DMTLOGDEV 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。このパラメータが指定されない場合、デフォルトの文字列である "DMTLOG" が使用されます。名前は 30 文字以内で指定する必要があります。

DMTLOGSIZE = *numeric*

このマシンに対するドメイン・トランザクション・ログのサイズをページ数で指定します。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い値を指定する必要があります。指定されない場合、デフォルトの 100 ページが設定されます。

MAXRDOM = *numeric*

1 つのゲートウェイに対する最大接続数を指定します。このパラメータは、ドメイン・タイプが OSITP または SNA の場合にのみ適用されます。

MAXRDTRAN = *numeric*

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。指定されない場合、デフォルトの 16 が設定されます。

MAXRETRY = {*numeric* | MAXLONG}

ドメイン・ゲートウェイがリモート・ドメインに対して接続を試みる回数を指定します。最小値は 0 であり、最大値は MAXLONG です。MAXLONG を指定すると、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。接続方針が ON_STARTUP の場合、MAXRETRY はデフォルトで MAXLONG に設定されます。MAXRETRY=0 を指定すると、自動再接続処理は行われません。その他の接続方針では、自動再接続処理機能は無効です。MAXRETRY パラメータは、接続方針が ON_STARTUP の場合にのみ有効です。ドメイン・タイプが OSI または SNA の場合、MAXRETRY パラメータは適用できません。

MAXTRAN = *numeric*

このローカル・ドメイン上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。0 以上で、TUXCONFIG ファイルで指定した MAXGTT パラメータ以下の値を指定する必要があります。このパラメータが指定されない場合、デフォルトの MAXGTT が設定されます。

MTYPE = *value*

このパラメータを使用して、ドメインをグループ化し、ドメイン間のメッセージの符号化と復号化を省略できるようにします。MTYPE が指定されていないと、デフォルトにより符号化や復号化が実行されます。MTYPE フィールドに設定した値がドメイン・コンフィギュレーション・ファイルの DM_LOCAL_DOMAINS と DM_REMOTE_DOMAINS の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 の *value* には、15 文字まで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。

RETRY_INTERVAL = *numeric*

リモート・ドメインへの自動接続が試行される間隔を秒単位で指定します。最小値は 0 であり、最大値は 2147483647 です。RETRY_INTERVAL のデフォルト値は 60 です。MAXRETRY が 0 に設定されている場合、RETRY_INTERVAL は指定できません。

接続方針が ON_STARTUP に設定されている場合のみ、RETRY_INTERVAL パラメータは有効です。その他の接続方針では、自動再接続処理機能は無効です。ドメイン・タイプが OSI または SNA の場合、RETRY_INTERVAL パラメータは適用できません。

SECURITY = *value*

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。SECURITY パラメータで指定できる値は、NONE、APP_PW、DM_PW の 3 つです。NONE は、セキュリティ機能を使用しないことを示します。これがデフォルトの設定です。APP_PW は、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・セキュリティを使用することを示します。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 は、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・セキュリティを使用することを示します。ドメイン・パスワードは、dmadmin(1) コマンドを使用して定義しておく必要があります。

SECURITY オプションは、OSITP ドメイン・タイプには適用されません。BEA Tuxedo 8.0 以上で使用される OSI TP 4.0 以上のゲートウェイのドメイン・タイプ OSITPX には、NONE または DM_PW を使用できます。

DM_REMOTE_DOMAINS セクション

このセクションは、既知のリモート・ドメインとその特性を識別します。

エントリの形式は次のとおりです。

RDOM *required_parameters* [*optional_parameters*]

RDOM は、このコンフィギュレーションで既に使用されている各リモート・ドメインを識別する *identifier* 値です。*RDOM* は、コンフィギュレーション内で一意な値である必要があります。

以下は、必須パラメータです。

TYPE = *identifier*

このパラメータを使用して、リモート・ドメインをグループ化します。TYPE には、TDOMAIN、SNAX、OSITP、OSITPX または TOPEND のいずれかを指定できます。TDOMAIN は、このリモート・ドメインがほかの BEA Tuxedo システム・ドメインとのみ通信できることを示します。SNAX は、このドメインが SNA プロトコルを介してほかの TP ドメインと通信することを示します。OSITP または OSITPX は、このリモート・ドメインが OSITP プロトコルを介してほかの TP ドメインと通信することを示します。TOPEND は、このリモート・ドメインが BEA TOP END システムであり、TOPEND タイプのローカル・ドメインとのみ通信できることを示します。

注記 OSITPX は、BEA OSI TP 4.0 またはそれ以降のバージョンのゲートウェイであり、BEA Tuxedo 8.0 以降に接続する場合に必ず使用します。

DOMAINID = *string*

このパラメータを使用して、リモート・ドメインを識別します。DOMAINID は、30 以下のオクテット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)。DOMAINID は、各リモート・ドメインに対して一意な識別子でなければなりません。*string* の値は、一連の文字か、または "0x" で始まる 16 進数で指定できます。

オプション・パラメータは、ドメイン・ゲートウェイの操作で使用する資源と最大値 / 最小値を指定します。

ACL_POLICY = {LOCAL | GLOBAL}

このリモート・ドメインに対する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を実行している TDOMAIN ドメインにのみ適用されます。

LOCAL は、ローカル・ドメインがこのリモート・ドメインから受け取ったサービス要求のアイデンティティを、このリモート・ドメインの LOCAL_PRINCIPAL_NAME パラメータで指定されるプリンシパル名に変更することを意味します。GLOBAL は、ローカル・ドメインがインバウンドのサービス要求でリモート・ドメインから受け取ったクリデンシャルを使用することを意味します。リモート・ドメインからクリデンシャルを受け取っていない場合は、クリデンシャルがないままサービス要求がサービスに転送されますが、通常これは失敗します。このパラメータを指定しないと、デフォルトの LOCAL が使用されます。

このパラメータは、リモート・ドメインからのクリデンシャルをローカル・ドメインで受け付けるかどうかを制御します。このパラメータに関連するパラメータとして CREDENTIAL_POLICY があり、これはローカル・ドメインがリモート・ドメインにクリデンシャルを送信するかどうかを制御します。

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・ドメインへの接続を確立する際に、このリモート・ドメインを確認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を実行している TDOMAIN ドメイン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により、このリモート・ドメインの DOMAINID 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・ドメインの CONNECTION_PRINCIPAL_NAME パラメータに値を割り当てるときは、このリモート・ドメインの DOMAINID パラメータに割り当てられる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: Unable to initialize administration key for domain *domain_name*.

CREDENTIAL_POLICY={LOCAL|GLOBAL}

このリモート・ドメイン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を実行する TDOMAIN ドメインにのみ適用されます。クリデンシャル方針が LOCAL の場合、ドメインは、呼び出しに対して要求を発信したユーザのクリデンシャルをリモート・ドメインにアタッチしません。クリデンシャル方針が GLOBAL の場合は、ドメインは、呼び出しに対して要求を発信したユーザのクリデンシャルをリモート・ドメインにアタッチします。このパラメータを指定しないと、デフォルトの LOCAL が使用されます。

このパラメータは、ユーザのクリデンシャルをリモート・ドメインに送信するかどうかを制御します。このパラメータに関連するパラメータとして ACL_POLICY があり、これは送信されたクリデンシャルをドメインで受け付けるかどうかを制御します。

LOCAL_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、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た ID です。このパラメータは、BEA Tuxedo 7.1 以降のソフトウェアを実行している TDOMAIN ドメインにのみ適用されます。

LOCAL_PRINCIPAL_NAME パラメータは、このリモート・ドメインの ACL_POLICY パラメータが LOCAL に設定されている (またはデフォルトにより LOCAL が使用される) 場合のみ有効です。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により、このリモート・ドメインの DOMAINID 文字列になります。

MTYPE = *value*

このパラメータを使用して、ドメインをグループ化し、ドメイン間のメッセージの符号化と復号化を省略できるようにします。MTYPE が指定されていないと、デフォルトにより符号化や復号化が実行されます。MTYPE フィールドに設定した値がドメイン・コンフィギュレーション・ファイルの DM_LOCAL_DOMAINS と DM_REMOTE_DOMAINS の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設定できます。この値は比較のためだけに使用します。

リモート・ドメインに関連付けられたエントリは、複数回指定できます。最初の指定 (リモート・ドメインへの接続が最初に試行されるときの接続先) は、一次アドレスとみなされます。一次エントリの NWADDR を使用してネットワーク接続が確立できない場合、2 番目のエントリに関連付けられた NWADDR が使用されます。

PRIORITY_TYPE = {LOCAL_RELATIVE | LOCAL_ABSOLUTE | GLOBAL}

INPRIORITY = number

このリモート・ドメインのメッセージの優先順位に関する処理を指定します。このパラメータは、BEA Tuxedo 8.0 以降のソフトウェアを実行しているドメインに適用されます。LOCAL_RELATIVE および LOCAL_ABSOLUTE パラメータは、すべてのドメイン・タイプに適用できます。ただし、GLOBAL パラメータは TDOMAIN のドメイン・タイプにしか適用されません。

LOCAL_RELATIVE パラメータは、tpsprio 呼び出しなどによってリモート・ドメインからの要求の優先順位がローカル・ドメインに伝播されないことを意味します。INPRIORITY の値を設定すると、リモート・ドメイン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。INPRIORITY の値は -99 ～ +99 です (-99 と +99 を含む)。INPRIORITY が設定されていない場合は、0 がデフォルト値になります。INPRIORITY の設定値により、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。最も高い優先順位は 100 です。リモート・ドメインに対する要求では、要求の優先順位も一緒にリモート・ドメインに送信されます。

LOCAL_ABSOLUTE パラメータは、リモート・ドメインからの要求の優先順位がローカル・ドメインに伝播されないことを意味します。ただし、受信した要求の優先順位は、INPRIORITY の絶対値に設定されます。INPRIORITY が設定されていない場合は、デフォルト値として 50 が設定されます。INPRIORITY の値の範囲は 1 ～ 100 (1 と 100 を含む) です。100 は最も高い優先順位です。リモート・ドメインへの要求の場合は、要求の優先順位が要求とともにリモート・ドメインに送信されます。

GLOBAL パラメータは、リモート・ドメインからの要求の優先順位がローカル・ドメインに伝播されることを意味します。特定のサービスに対するデフォルト値はローカルでは無視されます。INPRIORITY を設定すると、リモート・ドメインから受信する要求の優先順位は INPRIORITY の値に加算されます。受信した要求には、優先順位の合計の絶対値が設定されます。INPRIORITY の値は -99 ～ +99 です (-99 と +99 を含む)。INPRIORITY が設定されていない場合は、0 がデフォルト値になります。INPRIORITY の設定値により、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。最も高い優先順位は 100 です。INPRIORITY を 0 に設定した場合は、リモート・ドメインから優先順位が直接受け取られます。リモート・ドメインに対する要求では、要求の優先順位も一緒にリモート・ドメインに送信されます。

特に指定がない場合は、デフォルト設定として LOCAL_RELATIVE が使用され、INPRIORITY は 0 に設定されます。

DM_TDOMAIN セクション

このセクションは、TDOMAIN タイプのドメインに必要なアドレス指定情報を定義します。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 (ゲートウェイ・グループ) で受け付けられる場合、このセクションには、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、定義されたローカル・ドメインがアクセスできるリモート・ドメインごとに、1 つのエントリがなければなりません。

エントリの形式は次のとおりです。

DOM required_parameters [*optional_parameters*]

DOM は、DM_LOCAL_DOMAINS セクションまたは DM_REMOTE_DOMAINS セクションで、ローカル・ドメイン (*LDOM*) またはリモート・ドメイン (*RDOM*) を識別する値です。*DOM* 識別子は、DM_LOCAL_DOMAINS セクションで定義した *LDOM*、または DM_REMOTE_DOMAINS セクションで定義した *RDOM* と一致していなければなりません。

以下は、必須パラメータです。

NWADDR = *string*

このパラメータを使用して、ローカル・ドメインまたはリモート・ドメインに関連するネットワーク・アドレスを指定します。ローカル・ドメインの場合は、NWADDR を使用して、ほかの BEA Tuxedo システム・ドメインからの接続を受け付けます。リモート・ドメインの場合は、NWADDR を使用して接続を開始します。プロセスが接続指示を受け付けるアドレスとして使用するネットワーク・アドレスを指定します。ドメイン・ゲートウェイ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ゲートウェイ・プロセスの通信手段となります。

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

"*//host.name*:*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ネーム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。文字列 *#.#.#.#* 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 から 65535 の間の 10 進数または名前です。

注記 一部のポート番号は、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ことがあります。お使いのシステムでどの番号が予約されているか (もしあれば) を調べるには、トランスポート・プロトコルのマニュアルを参照してください。

このパラメータによりローカルまたはリモート・ドメインが使用するネットワーク・アドレスを指定して、ほかの BEA Tuxedo システム・ドメインからの接続を受け付けます。文字列の形式が "*0xhex-digits*" の場合、偶数の有効 16 進数桁を含める必要があります。

以下は、オプション・パラメータです。

NWDEVICE = *string*

このパラメータは、ローカル・ドメインまたはリモート・ドメインの接続指示受け付けアドレスにバインディングするときに使用するデバイス・ファイル名を指定します。NWDEVICE パラメータは不要です。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*

このパラメータは、データをリモート・ドメインに送信する際に使用する圧縮しきい値を指定します。このサイズを超えるアプリケーション・バッファは圧縮されます。このパラメータのデフォルト値は 2,147,483,647 です。

MINENCRYPTBITS = {0 | 40 | 56 | 128}

このドメイン用にリンクを確立する際に必要な最小暗号化レベルを指定します。0 は暗号化を行わないことを示します。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、下位互換性のために用意されています。

MAXENCRYPTBITS = {0 | 40 | 56 | 128}

このドメイン用にリンクを確立する際に使用できる最大暗号化レベルを指定します。0 は暗号化を行わないことを示します。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。デフォルト値は 128 です。

注記 リンク・レベルの暗号化の値 40 は、下位互換性のために用意されています。

リモート・ドメインに関連付けられたエントリは、複数回指定できます。最初の指定 (リモート・ドメインへの接続が最初に試行されるときの接続先) は、一次アドレスとみなされます。一次エントリの NWADDR を使用してネットワーク接続が確立できない場合、二次エントリに関連付けられた NWADDR が使用されます。

この TDOMAIN がローカル・ドメインの場合 (つまり、*DOM* が以前に指定した *LDOM* と一致している場合)、NWADDR は着信に対する接続指示の受け付けに使用するネットワーク・アドレスになります。二次エントリは、ローカル・ドメイン・エントリには使用できません。

この TDOMAIN エントリが二次リモート・ドメインを指す場合 (つまり、*DOM* が以前に指定した *RDOM* と一致している場合)、エントリはゲートウェイを指し、ネットワーク接続が一次エントリの NWADDR を使って確立しない時だけ使用します。二次ゲートウェイは、一次とは異なる BEA Tuxedo ドメインに属さなければなりません。しかし、二次ゲートウェイの DOMAINID は、一次ゲートウェイの DM_LOCAL_DOMAINS セクションで指定したものと同じでなければなりません。この仕組みはよく「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、一次ゲートウェイが使用できる時には、「ミラー」ゲートウェイの使用はお勧めできません。

DM_ACCESS_CONTROL セクション

このセクションは、ローカル・ドメインが使用するアクセス制御リストを指定します。このセクションの行は、次の形式をとります。

ACL_NAME required_parameters

ACL_NAME はアクセス制御リストを指定するための *identifier* 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 は、カンマで区切った 1 つ以上のリモート・ドメイン名 (RDOM) で構成されます。ワイルドカード文字 (*) を使用すると、DM_REMOTE_DOMAINS セクションで定義したすべてのリモート・ドメインがローカル・ドメインにアクセスできることを指定できます。

DM_LOCAL_SERVICES 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サービス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す。このセクションを指定しない場合、DM_LOCAL_DOMAINS セクションで定義したすべてのローカル・ドメインは、BEA Tuxedo システムのドメイン・アプリケーションが提供するすべてのサービスに対する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。

このセクションの行の形式は、次のとおりです。

service [*optional_parameters*]

service は、エクスポートされたサービスのローカル名を指定する識別子の値であり、15 文字以内で指定する必要があります。この名前は、ローカルの BEA Tuxedo システム・ドメイン・アプリケーションとともに動作する 1 つ以上のサーバによって宣言された名前に対応します。エクスポートされたサービスは、TUXCONFIG ファイルの SERVICES セクション内のエントリのサービスに対して指定された特性か、またはデフォルトの特性を継承します。継承される特性として、LOAD、PRIO、AUTOTRAN、ROUTING、BUFTYPE、TRANTIME があります。

以下は、オプション・パラメータです。

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・ドメインはこのリストを使用して、リモート・ドメインがこのサービスに対して行う要求を制限します。ACL の名前は、DM_ACCESS_CONTROL セクションで定義します。このパラメータを指定しないと、このサービスに対する要求についてアクセス制御が実行されません。

LDOM = *identifier*
: このサービスをエクスポートするローカル・ドメインの (識別子) 名前を指定します。このキーワードを指定しない場合、DM_LOCAL_DOMAINS セクションで定義したすべてのローカル・ドメインは、このローカル・サービスに対する要求を受け付けます。

INBUFTYPE = *type[:subtype]*
: このサービスが受け付けるデータ・タイプのバッファ・タイプ・ネーミング・スペースを、1 つのバッファ・タイプに制限します。UDT ASE アプリケーション・コンテキストを使用する OSITP タイプのゲートウェイからサービスを使用する場合は、このパラメータを定義する必要があります。このパラメータは TDOMAIN には適用されません。

OUTBUFTYPE = *type[:subtype]*
: このサービスが返すデータ・タイプのバッファ・タイプ・ネーミング・スペースを、1 つのバッファ・タイプに制限します。UDT ASE アプリケーション・コンテキストを使用する OSITP タイプのゲートウェイからサービスを使用する場合は、このパラメータを定義する必要があります。OSITP タイプのゲートウェイからサービスを使用する場合は、FML バッファ・タイプを使用できません。このパラメータは TDOMAIN には適用されません。

RNAME = *string*
: リモート・ドメインにエクスポートされるサービスの名前を指定します。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・ドメインは任意のローカル・サービス名を使用してサービスを要求できます。

DM_REMOTE_SERVICES セクション

このセクションは、リモート・ドメイン上に「インポート」されて使用可能になったサービスに関する情報を提供します。DM_REMOTE_SERVICES セクションの行は、次の形式をとります。

service [*optional_parameters*]

service は、ローカル BEA Tuxedo システム・ドメイン・アプリケーションが特定のリモート・サービスに対して使用する *identifier* の名前です。リモート・サービスは特定のリモート・ドメインと関連付けられています。

以下は、オプション・パラメータです。

CONV = {Y | N}
: リモート・サービスが会話サービスであるかどうかを指定します。会話サービスの場合は Y を、それ以外の場合は N を指定します。デフォルトは N です。

LDOM = *identifier*

このリモート・サービスに要求をルーティングするローカル・ドメインの名前を指定します。このローカル・ドメインと関連付けられたゲートウェイ・グループは、この *service* が使用可能であることを、BEA Tuxedo システム・ドメインの掲示板で知らせます。このパラメータを指定しないと、すべてのローカル・ドメインは、このリモート・サービスに対する要求を受け付けることができ、サービス要求は同じタイプのリモート・ドメインに転送されます (後述の RDOM キーワードを参照)。

INBUFTYPE = *type*[:*subtype*]

このサービスが受け付けるデータ・タイプのバッファ・タイプ・ネーミング・スペースを、1 つのバッファ・タイプに制限します。UDT ASE アプリケーション・コンテキストを使用する OSITP タイプのゲートウェイからサービスを使用する場合は、このパラメータを定義する必要があります。OSITP タイプのゲートウェイからサービスを使用する場合は、FML バッファ・タイプを使用できません。このパラメータは、TDOMAIN には適用されません。

OUTBUFTYPE = *type*[:*subtype*]

このサービスが返すデータ・タイプのバッファ・タイプ・ネーミング・スペースを、1 つのバッファ・タイプに制限します。UDT ASE アプリケーション・コンテキストを使用する OSITP タイプのゲートウェイからサービスを使用する場合は、このパラメータを定義する必要があります。OSITP タイプのゲートウェイからサービスを使用する場合は、FML バッファ・タイプを使用できません。このパラメータは、TDOMAIN には適用されません。

RDOM = *identifier1*[,*identifier2*][,*identifier3*]

このサービスを実行するリモート・ドメインの名前を指定します。このパラメータとルーティング基準が指定されないと、ローカル・ドメインは、同じ種類のリモート・ドメインはどれもこのサービスの要求を受け入れるものと見なします。

引数 *identifier2* および *identifier3* を指定して代替のリモート・ドメインを設定したい場合は、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* が設定されていると、フェイルオーバーに使用されます (*identifier1* に指定されたリモート・ドメインがアクセス不可能なため、*identifier2* に指定されたリモート・ドメインが使用される)。同様に、*identifier3* に指定された値もフェイルオーバーに使用されます (*identifier1* および *identifier2* に指定されたリモート・ドメインがアクセス不可能なため、*identifier3* に指定されたリモート・ドメインが使用される)。

RNAME = *string*

リモート・ドメインが要求する実際のサービス名を指定します。このパラメータを指定しない場合、リモート・サービス名は *service* で指定した名前になります。

ROUTING = *identifier*

複数のリモート・ドメイン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・ドメインはデータ依存型ルーティングを実行できます。*identifier* は、このデータ依存型ルーティングで使用する基準名を指定します。このパラメータの指定がないと、このサービスに対してデータ依存型ルーティングは行われません。*identifier* は、15 文字までで指定しなければなりません。サービス名が同じで、異なる RDOM パラメータを指定したエントリが複数個存在する場合は、これらのすべてのエントリで同じ ROUTING パラメータを指定する必要があります。

TRANTIME = *integer*

このサービス用に自動的に開始されるトランザクションに対して、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 2147483648 未満でなければなりません。デフォルトは 30 秒です。0 を指定すると、マシンの最大タイムアウト値に設定されます。

DM_RESOURCES

このオプションのセクションでは、グローバル・ドメイン・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。

このセクションのパラメータは、下記だけです。

VERSION = *string*

string は、ユーザがカレント・ドメイン・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。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_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、XML、VIEW、VIEW32、X_C_TYPE、および X_COMMON を使用した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提供します。DM_ROUTING セクションの行は、次の形式をとります。

CRITERION_NAME required_parameters

CRITERION_NAME は、サービス・エントリで指定されたルーティング・エントリの *identifier* の名前です。*CRITERION_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 = *identifier*

ルーティング・フィールドの名前を指定します。このパラメータは 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、あるいは X_COMMON バッファの場合) で識別されたフィールド名です。2 つの環境変数 FLDTBLDIR と FIELDTBLS または FLDTBLDIR32 と FIELDTBLS32 を使用して、FML フィールド・テーブルを検索します。同様に、2 つの環境変数 VIEWDIR と VIEWFILES または VIEWDIR32 と VIEWFILES32 を使用して FML VIEW テーブルを検索します。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。

XML ドキュメント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、大文字小文字を区別します。

XML についての詳細は、World Wide Web Consortium Web サイト http://www.w3c.org/XML をご覧ください。

FIELDTYPE = *type*

FIELD パラメータに指定されたルーティング・フィールドのフィールド・タイプを指定します。このパラメータは、XML バッファのルーティングでのみ使用します。値 *type* は、次のいずれかに設定できます。CHAR、SHORT、LONG、FLOAT、DOUBLE、または STRING です。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*"

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・ドメイン名 (*RDOM*) を指定します。*string* は二重引用符で囲みます。*string* の形式は、範囲 /*RDOM* のペアをカンマで区切って順番に並べます (後述の使用例を参照)。

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、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 (その後の範囲は無視されます)。

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。浮動小数点数は、C コンパイラまたは `atof()` 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ときに、関連する RDOM 値には、要求がルーティングされるリモート・ドメインを指定します。RDOM 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ることを示します。

範囲と RDOM のペアでは、範囲と RDOM が ":" で区切られます。

XML の要素の内容および属性値は 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るデータ型に変換可能な場合のみルーティングに使用できます。

ルーティングに使用する際には、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、または CDATA セクションを含むことはできません。

XML 属性値は (UTF-8 で符号化される) は、属性の対象である要素が定義される場合に、ルーティングで使用されます。

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

これは、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。タイプは FML、FML32、XML、VIEW、VIEW32、X_C_TYPE、または X_COMMON でなければなりません。FML、FML32、または XML に対してはサブタイプを指定することはできず、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 / サブタイプのペアが一意であれば、複数のルーティング・エントリに対して同じ基準名を指定することができます。このパラメータは必須です。1 つのルーティングに対して複数のバッファ・タイプを指定した場合は、それぞれのバッファ・タイプのルーティング・フィールドのデータ・タイプを同じにする必要があります。

(FML または FML32 バッファに対して) フィールド値が設定されていない場合、またはフィールド値が特定の範囲と一致しないで、ワイルドカード範囲が指定されていない場合は、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。

DMCONFIG(5) に関する追加情報

ファイル　　BDMCONFIG 環境変数を使用して、BDMCONFIG コンフィギュレーション・ファイルを検索します。

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 ドメイン・コンフィギュレーション・ファイル
#
#
*DM_LOCAL_DOMAINS
# <local domain name> <Gateway Group name> <domain type> <domain id> <log device>
```

```
#      [<audit log>] [<blocktime>]
#      [<log name>] [<log offset>] [<log size>]
#      [<maxrdom>] [<maxrdtran>] [<maxtran>]
#      [<maxdatalen>] [<security>]
#      [<tuxconfig>] [<tuxoffset>]
#
#
DEFAULT:SECURITY = NONE

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

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

#
*DM_REMOTE_DOMAINS
#remote <domain name> <domain type> <domain id>
#
b01  TYPE = TDOMAIN
     DOMAINID = "BA.BANK01"

b02  TYPE = TDOMAIN
     DOMAINID = "BA.BANK02"

b03  TYPE = TDOMAIN
     DOMAINID = "BA.BANK03"

b04  TYPE = OSITP
     DOMAINID = "BA.BANK04"

*DM_TDOMAIN
#
# <local or remote domain name> <network address> [<nwdevice>]
#
# 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
#
#<local or remote domain name> <apt> <aeq>
#   [<aet>] [<acn>] [<apid>] [<aeid>]
#   [<profile>]
#
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_LOCAL_SERVICES
#<service_name> [<Local Domain name>] [<access control>] [<exported svcname>]
#       [<inbuftype>] [<outbuftype>]
#
open_act ACL = branch
close_act ACL = branch
credit
debit
balance
loan   LDOM = c02 ACL = loans

*DM_REMOTE_SERVICES
#<service_name> [<Remote domain name>] [<local domain name>]
#        [<remote svcname>] [<routing>] [<conv>]
#        [<trantime>] [<inbuftype>] [<outbuftype>]
#
tlr_add LDOM = c01 ROUTING = ACCOUNT
tlr_bal LDOM = c01 ROUTING = ACCOUNT
tlr_add RDOM = b04 LDOM = c02 RNAME ="TPSU002"
tlr_bal RDOM = b04 LDOM = c02 RNAME ="TPSU003"
tlr_bal RDOM = b02, b03 LDOM = c02


*DM_ROUTING
# <routing criteria> <field> <typed buffer> <ranges>
#
ACCOUNT FIELD = branchid BUFTYPE ="VIEW:account"
   RANGES ="MIN - 1000:b01, 1001-3000:b02, *:b03"

*DM_ACCESS_CONTROL
#<acl name> <Remote domain list>
#
branch ACLIST = b01, b02, b03
loans  ACLIST = b04
```

使用例 2　この例は、1 つの Bank Branch (BANK01) の BEA Tuxedo ドメイン・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 ドメイン・コンフィギュレーション・ファイル
#
#
*DM_LOCAL_DOMAINS
#
b01  GWGRP = auth
     TYPE = TDOMAIN
     DOMAINID = "BA.BANK01"
     DMTLOGDEV = "/usr/apps/bank/DMTLOG"

*DM_REMOTE_DOMAINS
#
c01  TYPE = TDOMAIN
     DOMAINID = "BA.CENTRAL01"

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

*DM_REMOTE_SERVICES
#

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 アドレス指定機能を使用していて、アドレスは 155.2.193.18 であり、backus.company.com という名前になっていると仮定します。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。ポート番号 2334 は bankapp-gwtaddr 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仮定した場合、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
```

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。その後に IP アドレス 155.2.193.12 の各要素が 16 進数に変換されています。つまり 155 が 9B になり、2 が 02 になるように次々に変換されています。

関連項目

dmadmin(1)、dmloadcf(1)、dmunloadcf(1)、tmboot(1)、tmshutdown(1)、DMADM(5)、GWTOPEND(5) の DMCONFIG、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』

GWTOPEND(5) の DMCONFIG

名前

DMCONFIG for GWTOPEND—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 ドメイン機能を利用して、通信およびサービスの共有が可能な 2 つ以上のドメイン (またはアプリケーション) のセットです。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、各ドメインのドメイン・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 ファイルと呼ばれます (環境変数によって、実際に使用されたファイル名は保持されます)。

DMCONFIG ファイルは以下を定義します。

- ローカル・ドメインが通信できるリモート・ドメイン
- リモート・ドメインがアクセス可能なローカル資源 (サービスやキューなど)
- ローカル・ドメインがアクセス可能なリモート資源
- どのローカル資源およびリモート資源に、どのゲートウェイを通してアクセス可能か

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。dmadmin(1) コマンドは、BDMCONFIG (またはそのコピー) を使用して、実行時アプリケーションを監視します。

ドメイン機能を使用するマルチ・ドメイン・コンフィギュレーションで、各ドメインに 1 つの BDMCONFIG ファイルが必要です。

DMCONFIG ファイルと BDMCONFIG ファイルの関係は、BEA Tuxedo アプリケーショ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。

GWTOPEND 用の DMCONFIG 全体に関連する追加情報については、109 ページ「GWTOPEND(5) の DMCONFIG に関する追加情報」を参照してください。

定義

BEA Tuxedo システムにおけるドメイン・アプリケーションは、環境の定義として単一の TUXCONFIG ファイル内で記述されます。BEA Tuxedo システムのアプリケーションは、ドメイン・ゲートウェイ・グループを介して、別の BEA TUXDO システムのアプリケーションや別の TP アプリケーションと通信できます。「BEA Tuxedo システム・ドメイン」の用語では、アプリケーションと TP ドメインは同義です。

ゲートウェイ・グループは、特定のタイプの TP ドメインと共に通信サービスを提供するドメイン・ゲートウェイ・プロセスの集合です。

ドメイン・ゲートウェイは、別の TP ドメインとの間で要求と応答を中継する BEA Tuxedo システムのドメイン・プロセスです。

ローカル・ドメインは、ほかのドメインからアクセス可能なアプリケーションの一部 (サービスのセットまたはサブセット) です。ローカル・ドメインは、常にドメイン・ゲートウェイ・グループとして表されるため、これらの 2 つの用語は同義語として使用されます。

リモート・ドメインは、ゲートウェイ・グループを介してアクセスされるリモート・アプリケーションです。リモート・アプリケーションは、ほかの BEA Tuxedo システムのドメイン・アプリケーションまたはほかの TP システムで動作するアプリケーションになります。

リモート・サービスは、アプリケーションがゲートウェイ・グループを介してアクセスするリモート・ドメインのサービスです。

ローカル・サービスは、リモート・ドメインがゲートウェイ・グループを介してアクセスするローカル・ドメインのサービスです。

コンフィギュレーション・ファイルのフォーマット

ドメイン・コンフィギュレーション・ファイルの形式は、次のとおりです。

このファイルには複数の指定セクションがあります。使用可能なセクションは、DM_LOCAL_DOMAINS、DM_REMOTE_DOMAINS、DM_LOCAL_SERVICES、DM_REMOTE_SERVICES、DM_TOPEND、DM_RESOURCES、DM_ROUTING、および DM_ACCESS_CONTROL です。ほかのゲートウェイ・タイプだけに適用される追加セクション名は、次のとおりです。DM_TDOMAINS、DM_OSITP、DM_SNACRM、DM_SNASTACKS、および DM_SNALINKS。DM_LOCAL_DOMAINS セクションは、DM_REMOTE_DOMAINS セクションの前になければなりません。

注記 このリファレンス・ページでは、TOPEND ドメイン・ゲートウェイの設定方法のみを説明します。TDOMAIN ドメインの設定方法については、DMCONFIG(5) を参照してください。OSITP または SNAX ドメインの設定方法については、BEA eLink マニュアル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 (=) のいずれの側にでも、空白類 (空白またはタブ文字) を使用できます。この形式で *KEYWORD* が *value* に設定されます。有効なキーワードは、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、ほかの DEFAULT: 行によってリセットされるか、または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 (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 などの BEA Tuxedo ドメイン機能に既知の文字列値) の場合、一般的に標準 C 規則が使用されます。標準 C *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。識別子に使用できる最大文字数は 30 文字です (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

リリース 7.1 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、クラスおよび属性の新しい用語を使用して、ローカル・ドメインとリモート・ドメイン間の相互作用を説明しています。新しい用語は以前のドメイン用語より正確になっていますが、ドメイン関連のマニュアルとエラー・メッセージで新しい用語が使用されているのはリリース 7.1 以降だけです。新しい用語は、DM_MIB クラス、リファレンス・ページ、エラー・メッセージ、DMCONFIG ファイル構文、および各種 DMCONFIG エラー・メッセージで使用されています。

下位互換性のため、このリリース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。このリリースでは、DMCONFIG は両方の用語を使用できます。詳細については、DM_MIB(5) リファレンス・ページの 114 ページ「Domains 関連の新しい用語」を参照してください。

DM_LOCAL_DOMAINS セクション

このセクションでは、ローカル・ドメインと関連するゲートウェイ・グループを識別します。このセクションには、各ゲートウェイ・グループ (ローカル・ドメイン) 用のエントリが必要です。各エントリでは、グループで実行中のドメイン・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。エントリは、1 つの BEA TOP END システムと関連付けられた GWTOPEND インスタンス (および関連する GWADM) を定義します。ローカル・ドメインは、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。

エントリの形式は次のとおりです。

LDOM required_parameters [*optional_parameters*]

LDOM は、ローカル・ドメインの名前として使用する *identifier* です。*LDOM* は、特定のコンフィギュレーション内で一意な値でなければなりません。
DM_LOCAL_SERVICES セクションで説明するとおり、*LDOM* はローカル・サービスおよびリモート・サービスを特定のゲートウェイ・グループに関連付ける識別子です。

以下は、必須パラメータです。

GWGRP = *identifier*
: このローカル・ドメインを表すゲートウェイ・サーバ・グループの名前 (TUXCONFIG ファイルで指定される名前) を指定します。DOMAINID (下記参照) とゲートウェイ・サーバ・グループ名は、1 対 1 の関係です。

TYPE = *identifier*
: このパラメータを使用して、複数のローカル・ドメインをグループ化します。TYPE には、TOPEND、TDOMAIN、SNAX、または OSITP のいずれかを指定できます。TOPEND は、このローカル・ドメインが BEA TOP END システムとのみ通信できることを示します。TDOMAIN は、このローカル・ドメインがほかの BEA Tuxedo システム・ドメインとのみ通信できることを示します。SNAX は、このローカル・ドメインが SNA プロトコルを介してほかの TP ドメインと通信できることを示します。OSITP は、このローカル・ドメインが OSI TP プロトコルを介してほかの TP ドメインと通信できることを示します。ドメインのタイプは、$TUXDIR/udataobj/DMTYPE ファイルで指定する必要があります。

DOMAINID = *string*

このパラメータを使用して、ローカル・ドメインを識別します。DOMAINID は、ローカル・ドメインとリモート・ドメインの両方に渡り、一意である必要があります。タイプ TOPEND のローカル・ドメインの場合、この値は、BEA TOP END システムに対して行われる要求の BEA TOP END ユーザ ID として使用されます。対応するパスワードは dmadmin(1) サブコマンド topendpasswd を使用して入力できます。BEA TOP END ユーザ ID は、最後のヌルを除いて 1 から 12 文字までです。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。

オプション・パラメータは、ドメイン・ゲートウェイの操作で使用する資源と最大値 / 最小値を指定します。

AUDITLOG = *string*

このローカル・ドメインに対する動作記録ログ・ファイルの名前を指定します。動作記録ログ機能は、dmadmin(1) コマンドによって起動し、このローカル・ドメインで行われるすべての動作を記録します。動作記録ログ機能がオンになっている場合に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 キーワードで指定されるディレクトリに、*DMmmddyy*.LOG ファイル (*mm* = 月、*dd* = 日、*yy* = 年) が作成されます。

BLOCKTIME = *numeric*

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 ファイルで指定した SCANUNIT パラメータの乗数を設定します。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 ファイルで指定した BLOCKTIME パラメータの値がデフォルトとして使用されます。タイムアウトの発生は、関連する要求が失敗したことを示します。トランザクション内で要求が発行されると必ず、TUXCONFIG ファイルでトランザクションに対して指定したタイムアウトが使用されます。

CONNECTION_POLICY = *string*

ローカル・ドメイン・ゲートウェイがリモート・ドメインに対して接続を試行するときの条件を指定します。指定可能な値は、ON_DEMAND、ON_STARTUP、および INCOMING_ONLY です。接続方針が ON_DEMAND の場合は、クライアントがリモート・サービスを要求したとき、または管理コマンド "connect" が実行されたとき、のいずれかの場合に接続が試行されます。CONNECTION_POLICY のデフォルト設定は 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を DM_TOPEND セクションで指定できます。つまり、接続方針が ON_DEMAND として設定されていると、再接続は行われません。

接続方針が 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みます。リモート・ドメインへの接続において、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する場合は、リモート・ドメインの複数のエントリを DM_TOPEND セクションで指定できます。CONNECTION_POLICY が ON_STARTUP に設定されていると、リモート・ドメインへの接続が確立された場合にのみリモート・サービス (ローカル・ドメイン・ゲートウェイによって宣言されたサービス) は宣言されます。つまり、リモート・ドメインとの接続が確立されていない場合、リモート・サービスは中断されます。デフォルトでは、60 秒おきに、異常終了した接続が再試行されるよう設定されています。すべてのネットワーク・アドレスが試行されます。再接続の間隔は変更することもできます (MAXRETRY および RETRY_INTERVAL を参照)。

接続方針が 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行しません。そのため、初期のリモート・サービスは中断されています。ドメイン・ゲートウェイは、リモート・ドメインからの受信時接続で利用できます。また、管理コマンド "connect" で要求されたときに、接続が試行されます。リモート・サービスは、ローカル・ドメイン・ゲートウェイが受信時接続を受信したとき、または管理コマンドで接続が行わ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時において、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する場合は、リモート・ドメインの複数のエントリを DM_TOPEND セクションで指定できます。INCOMING_ONLY が接続方針として設定されていると、再接続は行われません

DMTLOGDEV = *string*

このマシンのドメイン・トランザクション・ログ (DMTLOG) を含む BEA Tuxedo ファイル・システムを指定します。DMTLOG 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ます。このパラメータが指定されないと、ドメイン・ゲートウェイ・グループはトランザクション・モードで要求を処理できません。同じマシン上で動作しているローカル・ドメイン間で DMTLOGDEV 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・ドメインは、DMTLOGNAME キーワード (下記参照) 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 = *identifier*

このドメインに対するドメイン・トランザクション・ログの名前を指定します。複数のローカル・ドメイン間で DMTLOGDEV 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。このパラメータが指定されない場合、デフォルトの文字列である "DMTLOG" が使用されます。名前は 30 文字以内で指定する必要があります。

DMTLOGSIZE = *numeric*

このマシンに対するドメイン・トランザクション・ログのサイズをページ数で指定します。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い値を指定する必要があります。指定されない場合、デフォルトの 100 ページが設定されます。

MAXRDTRAN = *numeric*

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。指定されない場合、デフォルトの 16 が設定されます。

MAXRETRY = {*numeric* | MAXLONG}

ドメイン・ゲートウェイがリモート・ドメインへ接続を試みる回数を指定します。最小値は 0 であり、最大値は MAXLONG です。MAXLONG を指定すると、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。接続方針が ON_STARTUP の場合、MAXRETRY はデフォルトで MAXLONG に設定されます。MAXRETRY=0 を指定すると、自動再接続処理は行われません。その他の接続方針では、自動再接続処理機能は無効です。

MAXRETRY パラメータは、接続方針が ON_STARTUP の場合にのみ有効です。

MAXTRAN = *numeric*

このローカル・ドメイン上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。0 以上で、TUXCONFIG ファイルで指定した MAXGTT パラメータ以下の値を指定する必要があります。このパラメータが指定されない場合、デフォルトの MAXGTT が設定されます。

RETRY_INTERVAL = *numeric*

リモート・ドメインへの接続が自動的に試行される間隔を秒単位で指定します (すべてのネットワーク・アドレスが試行されます)。最小値は 0 であり、最大値は 2147483647 です。RETRY_INTERVAL のデフォルト値は 60 です。
MAXRETRY が 0 に設定されている場合、RETRY_INTERVAL は指定できません。

接続方針が ON_STARTUP に設定されている場合のみ、RETRY_INTERVAL パラメータは有効です。その他の接続方針では、自動再接続処理機能は無効です。

SECURITY = *value*

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 ドメインの SECURITY パラメータに指定できる値は、NONE、CLEAR、SAFE、PRIVATE の 4 つです。値 NONE はデフォルトで、認証、認可、またはノード間メッセージ保護に BEA TOP END セキュリティが使用されないことを示します。NONE 以外の値は、BEA TOP END システムとゲートウェイで BEA TOP END 認証および認可が使用されることを示します。また、値 CLEAR はノード間メッセージに対して保護が必要ないことを示します。値 SAFE は、メッセージが Kerberos SAFE メッセージ・チェックサムを使用して送信されることを示します。値 PRIVATE は、メッセージが DES の Kerberos 4 インプリメンテーションを使用して暗号化されることを示します。SECURITY の値は、各 BEA TOP END ノード上の nm_config (4T) ファイル内の対応する 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している必要があります。この一致に関しては、リモート BEA TOP END ノードとの接続確立時に検証されます。

DM_REMOTE_DOMAINS セクション

このセクションは、既知のリモート・ドメインとその特性を識別します。

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。

エントリの形式は次のとおりです。

RDOM required_parameters

RDOM は、このコンフィギュレーションで既に使用されている各リモート・ドメインを識別する *identifier* の値です。*RDOM* は、コンフィギュレーション内で一意な値である必要があります。

各 *RDOM* は BEA TOP END *LDOM* が接続する BEA TOP END システムのノードを定義します。*LDOM* は、*LDOM.* 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。BEA TOP END 隣接ノード・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は複数のノードに存在することが可能です。そのため、TEDG *LDOM* は、要求された BEA TOP END サービスが存在する BEA TOP END ノードへの接続を定義するための複数の *RDOM* エントリを必要とする場合があります。

TYPE パラメータおよび DOMAINID パラメータは、必須パラメータです。MTYPE は、タイプ TOPEND のリモート・ドメインでは使用されません。

TYPE = *identifier*

このパラメータを使用して、リモート・ドメインをグループ化します。TYPE には、TOPEND、TDOMAIN、SNAX、または OSITP のいずれかを指定できます。TOPEND は、このリモート・ドメインが BEA TOP END システムであり、TOPEND タイプのローカル・ドメインとのみ通信できることを示します。TDOMAIN は、このリモート・ドメインがほかの BEA Tuxedo システム・ドメインとのみ通信できることを示します。SNAX は、このドメインが SNA プロトコルを介してほかの TP ドメインと通信することを示します。OSITP は、このリモート・ドメインが OSI TP プロトコルを介してほかの TP ドメインと通信することを示します。

DOMAINID = *string*

このパラメータを使用して、リモート・ドメインを識別します。DOMAINID は、30 以下のオクテット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)。DOMAINID は、各リモート・ドメインに対して一意な識別子でなければなりません。*string* の値は、一連の文字か、または "0x" で始まる 16 進数で指定できます。DOMAINID 値は、この *RDOM* 接続上で BEA TOP END システムにより BEA Tuxedo システムに対して行われる要求の BEA Tuxedo ユーザ ID として、BEA TOP END ゲートウェイによって使用されます。

DM_TOPEND セクション

このセクションは、TOPEND タイプのドメインに必要なアドレス指定情報を定義します。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 (ゲートウェイ・グループ) で受け付けられる場合、このセクションには、ローカル・ドメインごとに 1 つのエントリが必要となります。また、定義されたローカル・ドメインがアクセスできるリモート・ドメインごとに、1 つ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。

DOM required_parameters [*optional_parameters*]

DOM は、DM_LOCAL_DOMAINS セクションまたは DM_REMOTE_DOMAINS セクションで、ローカル・ドメイン (*LDOM*) またはリモート・ドメイン (*RDOM*) を識別する値です。*DOM* 識別子は、DM_LOCAL_DOMAINS セクションで定義した *LDOM*、または DM_REMOTE_DOMAINS セクションで定義した *RDOM* と一致していなければなりません。

ローカル・ドメインとリモート・ドメイン、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 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設定する必要があります。BEA TOP END ネットワーク・インターフェイス・プロトコルは、複数のゲートウェイ接続をサポートしません。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、このような環境設定をこのコンフィギュレーション・ファイルで完全に有効にすることはできません。

以下は、必須パラメータです。

NWADDR = *string*

このパラメータを使用して、ローカル・ドメインまたはリモート・ドメインに関連するネットワーク・アドレスを指定します。ローカル・ドメインと関連がある場合は、NWADDR を使用して、BEA TOP END システムからの接続を受け付けます。リモート・ドメインと関連がある場合は、NWADDR を使用して接続を開始します。プロセスで使用するネットワーク・アドレスを、接続指示を受け付けるアドレスとして指定します。ドメイン・ゲートウェイに対する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・コンポーネントの通信手段となります。

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

"//*host.name*:*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名前解決機能を使ってアドレスが結合されるときに、*host.name* は TCP/IP ホスト・アドレスに解決されます。2 列目の #.#.#.# 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 の範囲の 10 進数で、指定された文字列の 16 進の表現です。

注記 一部のポート番号は、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ことがあります。お使いのシステムでどの番号が予約されているかを調べるには、トランスポート・プロトコルのマニュアルを参照してください。

このパラメータは、ローカルまたはリモート・ドメインが接続を受け付けるために使用するネットワーク・ポート・アドレスを指定します。文字列の形式が "0x*hex-digits*" の場合、偶数桁の 16 進数値を含める必要があります。

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 *LDOM* 接続指示受け付けアドレスとして 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。

注記 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 NI が TEDG から出された接続要求を受け取るとき、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 の定義されたホスト名と一致する必要があります。TEDG の定義されたホスト名と解決された名前が、大文字と小文字の相違も含めて異なる場合、NI 接続は異常終了します。このような異常終了は、GWTOPEND ログ・ファイルからも、リモート BEA TOP END NI ログ・ファイルからも明らかにならない場合があります。一般に、ホスト名の定義は DMCONFIG ファイル、TOP END NI コンフイギュレーション・ファイル、TOP END nodemap ファイル、TOP END tp_alias ファイル、およびローカル名の解決機能において一致していることが必要です。NI 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。

NWDEVICE = *string*

このパラメータは、ローカル・ドメインまたはリモート・ドメインの接続指示受け付けアドレスにバインディングするときに使用するデバイス・ファイル名を指定します。NWDEVICE パラメータは不要です。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。

TP_SYSTEM = *string*

コンフィギュレーション・ファイルの DM_LOCAL_DOMAINS または DM_REMOTE_DOMAINS セクションで定義される *LDOM* および *RDOM* と関連する BEA TOP END システムを定義します。このパラメータは、BEA TOP END システム名に対応する文字列を受け付けます。BEA TOP END システム名には、最後のヌルを除いて 1 文字から 8 文字まで指定できます。ASCII 文字 " "(32) から "~" (126) を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。

エントリが BEA TOP END ローカル・ドメインの場合 (つまり、*DOM* が以前に指定した *LDOM* と一致している場合)、NWADDR は着信に対する接続指示の受け付けに使用するネットワーク・アドレスになります。ローカル・ドメインに対しては 1 つのエントリだけを設定できます。

リモート・ドメインに対しては、複数のネットワーク・アドレスで接続試行が順に行われるように、複数のエントリを設定することができます。最初の指定は一次アドレス (リモート・ドメインへの接続が最初に試行されるときの接続先) と見なされます。一次エントリの NWADDR を使用してネットワーク接続を確立できない場合、2 番目のエントリに関連付けられた NWADDR が使用されます。それまでのエントリで接続がすべて異常終了すると、順に次のエントリが使用されます。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・ドメインに関連するエントリは、何回でも指定できます。ネットワーク・アドレスの設定が多すぎる場合、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。

エントリが二次リモート・ドメインを指す場合 (つまり、*DOM* が以前に指定した *RDOM* と一致している場合)、エントリはネットワーク接続が一次エントリの NWADDR (および前の二次エントリの NWADDR) を使って確立しないときにだけ使用します。各二次エントリで、

- 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しなければなりません。

二次リモート・ゲートウェイ定義は、TOP END ドメイン・ゲートウェイでは使用しないようにしてください。

DM_ACCESS_CONTROL セクション

このセクションは、ローカル・ドメインが使用するアクセス制御リストを指定します。

このセクションの行は、次の形式をとります。

ACL_NAME required_parameters

ACL_NAME はアクセス制御リストを指定するための *identifier* 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 は、カンマで区切った 1 つ以上のリモート・ドメイン名 (RDOM) で構成されます。ワイルドカード文字 (*) を使用すると、DM_REMOTE_DOMAINS セクションで定義したすべてのリモート・ドメインがローカル・ドメインにアクセスできることを指定できます。

DM_LOCAL_SERVICES セクション

このセクションでは、BEA Tuxedo サービスおよび /Q キュー・スペースを BEA TOP END システムで利用するために必要なマッピング情報を定義します。TEDG 以外のドメイン・ゲートウェイのために記述された DMCONFIG ファイルの場合、このセクションのエントリは、ローカル・サービスを対応するリモート名にマップするために使用します。一方、TEDG のために記述された DMCONFIG ファイルでは、タイプ・エントリは要求 / 応答および会話サービスのマッピングを定義するために使用します。さらに、BEA Tuxedo キュー・スペース・マッピングとキュー名マッピングを定義するためにも、このセクションのタイプ・エントリを使用します。SERVICE、QSPACE、または QNAME のエントリを設定することができます。各エントリは TYPE パラメータで識別されます。このセクションは TOP END ドメイン・ゲートウェイでは必須です。

このセクションの行の形式は、次のとおりです。

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。

SERVICE エントリは、TEDG によって BEA TOP END システムに宣言される BEA Tuxedo サービス (製品、関数、ターゲット) を定義します。BEA TOP END システムに宣言される BEA Tuxedo サービスのエントリには、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を含む必要があります。これらのサービス識別子は、BEA TOP END 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。

このセクションの QSPACE エントリは、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 (制限が適用されます)。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 で利用可能になります。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 サーバと同じように動作して処理します。次に、各要求はこの QSPACE エントリで識別される BEA Tuxedo キュー・スペースにマップされます。メッセージをキューに入れるために、QSPACE エントリと QNAME エントリの両方が必要です。

QNAME エントリは、RTQ を通して BEA Tuxedo システムのキューに入れられる要求に対して、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は宣言されません。QSPACE エントリと QNAME エントリは独立しています。QSPACE 識別子と 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、特定の LDOM 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、あるいは 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 *LDOM* パラメータで特定のローカル・ドメインに対して設定されていない場合、すべての TOPEND ローカル・ドメインに適用できます。特定の LDOM に対して設定されたエントリは、そのドメインのゲートウェイだけに適用されます。

SERVICE および QSPACE エントリは、BEA TOP END サービスとして宣言される BEA TOP END サービス識別子を設定するので、これらの識別子が特定の LDOM に対して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 エントリでは、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。したがって、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と、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。

コンフィギュレーションに複数の BEA TOP END システムの *LDOM* を含む場合、または複数のタイプのゲートウェイを含む場合は、LDOM パラメータをローカル・サービス・エントリで指定する必要があります。LDOM を指定せずに、混在型のコンフィギュレーションを作成することは避けてください。ゲートウェイが正しく初期化しないことがあります。疑わしい場合は、明示的に LDOM を設定してください。

次に、エントリ・タイプごとに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、ACL、LDOM、INBUFTYPE、 OUTBUFTYPE、TE_TARGET、 TE_QUALIFIER

QSPACE	TYPE、TE_RTQGROUP、TE_RTQNAME	ACL、LDOM、TE_TARGET
QNAME	TYPE、TE_PRODUCT、TE_FUNCTION	LDOM、INBUFTYPE、TE_TARGET、TE_QUALIFIER

必須パラメータとオプション・パラメータについて、以下に説明します。

TYPE = SERVICE | QSPACE | QNAME
: 定義されるエントリのタイプを指定します。値 SERVICE は、そのエントリが、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・パラメータを定義することを指定します。値 QSPACE は、そのエントリが、BEA TOP END システムによって RTQ キューとして利用されるローカル BEA Tuxedo キュー・スペースに適用可能なッピング・パラメータを定義することを指定します。値 QNAME は、そのエントリが、RTQ を通して BEA Tuxedo システムのキューに入れられる要求に対して、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することを指定します。デフォルト値は SERVICE です。

TE_PRODUCT = *string*
: 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_FUNCTION = *string*
: 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_TARGET = *string*
: BEA TOP END Message 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。*string* の値の長さは、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。SERVICE および QSPACE エントリでは、最後の非空白文字としてアスタリスクを使用できます。DMCONFIG ファイルの 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 および QSPACE エントリの場合、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_QUALIFIER = *integer*
: BEA TOP END 関数修飾子を指定します。有効な値の範囲は、0 から 2147483647 までです。デフォルト値は "0" です。

TE_RTQGROUP = *string*
: BEA TOP END RTQ グループ名を指定します。*string* の値の長さは、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_RTQNAME = *string*
: BEA TOP END RTQ キュー名を指定します。*string* の値の長さは、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。

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 システムがこの SERVICE または QSPACE に対して行う要求を制限します。ACL は、DMCONFIG ファイルの DM_ACCESS_CONTROL セクションで定義します。このパラメータを指定しないと、このエントリに対する要求についてアクセス制御が実行されません。

LDOM = *identifier*
: この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、あるいはキュー名エントリが適用されるローカル・ドメインの名前を指定します。このキーワードを指定しない場合、エントリは、DM_LOCAL_DOMAINS セクションで定義したすべての TOPEND ローカル・ドメインに適用されます。

INBUFTYPE = *type*[:*subtype*]
: このサービスまたはキュー名が使用できる入力バッファ・タイプを、1 つのバッファ・タイプに制限します。BEA TOP END サービスおよびキュー名エントリでは、有効なタイプの値は FML32、CARRAY、および X_OCTET です。

OUTBUFTYPE = *type*[:*subtype*]
: このサービスから受け付けられる出力バッファ・タイプを、1 つのバッファ・タイプに制限します。BEA TOP END サービス・エントリでは、有効なタイプの値は FML32、CARRAY、および X_OCTET です。

DM_REMOTE_SERVICES セクション

このセクションは、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利用するために必要なマッピング情報を定義します。TEDG 以外のドメイン・ゲートウェイのために記述された DMCONFIG ファイルの場合、このセクションのエントリは、ローカル・サービスを対応するリモート名にマップするために使用します。

一方、TEDG のために記述された DMCONFIG ファイルでは、このセクションのエントリの目的は異なります。

- TYPE パラメータは、要求 / 応答および会話サービスのマッピングを定義するために使用します。
- BEA Tuxedo キューとキュー名のマッピングを定義するためにも、このセクションのタイプ・エントリを使用します。

SERVICE、QSPACE、または QNAME のエントリを設定することができます。各エントリは TYPE パラメータで識別されます。このセクションは TOP END ドメイン・ゲートウェイでは必須です。

このセクションの行の形式は、次のとおりです。

```
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。

SERVICE エントリは、TEDG によって BEA Tuxedo ドメインに宣言される BEA TOP END サービスを定義します。アプリケーションに宣言される BEA TOP END サービスのエントリは、BEA Tuxedo サービス名を BEA TOP END サービス識別子にマップしなければなりません。これらのサービス名は、tpcall(3c) および tpacall(3c) 関数で使用します。

このセクションの QSPACE エントリは、TEDG によって、BEA Tuxedo ドメインで BEA Tuxedo キュー・スペース同様に利用できるようになる BEA TOP END RTQ キューを定義します (制限が適用されます)。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 システムで利用可能になります。ゲートウェイは、そのキュー・スペース名に送信された tpenqueue 要求を、TMQUEUE サーバと同じように動作して処理します。次に、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。メッセージをキューに入れるために、QSPACE エントリと QNAME エントリの両方が必要です。

QNAME エントリは、BEA TOP END システムのキューに入れられる要求に対して、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 BEA Tuxedo システムにサービスとしては宣言されません。QSPACE エントリと QNAME エントリは独立しています。QSPACE 識別子と QNAME 識別子は、tpenqueue(3c) 関数を使用して、アプリケーションにより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 LDOM で一意でなければなりません。同じ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、あるいは 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 LDOM パラメータで特定のローカル・ドメインに対して設定されていない場合、各 TOPEND ローカル・ドメインに適用できます。特定の LDOM に対して設定されたエントリは、そのドメインのゲートウェイだけに適用されます。

SERVICE および QSPACE エントリは、BEA Tuxedo サービスとして宣言されるサービス識別子およびキュー・スペース識別子を設定するので、これらの識別子が特定の LDOM に対して重複してはいけません。ただし、ロード・バランシングのためには、タイプと識別子が同じエントリを複数設定できます。同じサービス識別子のエントリはすべて、CONV パラメータの値が同じである必要があります。

コンフィギュレーションに複数の BEA TOP END システムの LDOM を含む場合、または複数のタイプのゲートウェイを含む場合は、LDOM パラメータを DMCONFIG ファイルの DM_REMOTE_SERVICES セクションで指定する必要があります。RDOM 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は、その値は LDOM タイプ (TOPEND) および TP_SYSTEM の値と一致しなければなりません。LDOM を指定せずに、あるいはタイプまたは TP_SYSTEM が混在する RDOM 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。ゲートウェイが正しく初期化しないことがあります。疑わしい場合は、明示的に LDOM を設定し、また (RDOM パラメータまたは ROUTING を介して) リモート・ドメインを指定してください。リモート・ドメインに対する「ワイルドカード」仕様は、単一のゲートウェイ・タイプを定義するときだけ使用してください。

次に、エントリ・タイプごとに必須パラメータとプション・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、LDOM、RDOM、INBUFTYPE、OUTBUFTYPE、CONV、TE_TARGET、TE_QUALIFIER、TRANTIME、ROUTING
QSPACE	TYPE、TE_RTQGROUP、TE_RTQNAME	LDOM、RDOM、TE_TARGET、TRANTIME、ROUTING
QNAME	TYPE、TE_PRODUCT、TE_FUNCTION	LDOM、INBUFTYPE、TE_TARGET、TE_QUALIFIER

必須パラメータとオプション・パラメータについて、以下に説明します。

TYPE = SERVICE | QSPACE | QNAME
: 定義されるエントリのタイプを指定します。値 SERVICE は、そのエントリが、BEA TOP END サービスをローカル BEA Tuxedo サービスとして利用するために必要なマッピング・パラメータを定義することを指定します。値 QSPACE は、そのエントリが、BEA TOP END RTQ キューをローカル BEA Tuxedo キュー・スペースとして利用するために必要マッピング・パラメータを定義することを指定します。値 QNAME は、そのエントリが、/Q を通して BEA TOP END システムのキューに入れられる要求に対して、BEA Tuxedo キュー名を BEA TOP END サービス名にマッピングするために必要なパラメータを定義することを指定します。デフォルト値は SERVICE です。

TE_PRODUCT = *string*
: 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_FUNCTION = *string*
: 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_TARGET = *string*
: 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。デフォルトは空白です。

TE_QUALIFIER = *integer*
: BEA TOP END 関数修飾子を指定します。有効な値の範囲は、0 から 2147483647 (符号なし long の最大値) までです。デフォルト値は "0" です。

TE_RTQGROUP = *string*
: BEA TOP END RTQ グループ名を指定します。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_RTQNAME = *string*
: BEA TOP END RTQ キュー名を指定します。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。

LDOM = *identifier*

このサービスまたは RTQ キューをインポートするローカル・ドメイン、あるいはキュー名エントリが適用されるローカル・ドメインの名前を指定します。このキーワードを指定しない場合、エントリは、DM_LOCAL_DOMAINS セクションで定義したすべての TOPEND ローカル・ドメインに適用されます。

RDOM = *identifier1*[,*identifier2*][,*identifier3*]

このサービスまたは RTQ キューを提供するリモート・ドメインの名前を指定します。リモート・ドメインは、タイプ TOPEND で、このエントリが適用されるローカル・ドメインと同じ TP_SYSTEM の一部でなければなりません。このパラメータとルーティング基準が指定されないと、ローカル・ドメインは、TP_SYSTEM の値がローカル・ドメインと同じで、タイプが TOPEND のリモート・ドメインはどれも、サービスまたは RTQ キューを提供するものと見なします。

引数 *identifier2* および *identifier3* を指定して代替のリモート・ドメインを設定する場合は、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* が設定されていると、フェイルオーバーに使用されます (*identifier1* に指定されたリモート・ドメインがアクセス不可能なため、*identifier2* に指定されたリモート・ドメインが使用される)。同様に、*identifier3* に指定された値もフェイルオーバーに使用されます (*identifier1* および *identifier2* に指定されたリモート・ドメインがアクセス不能の場合、*identifier3* に指定されたリモート・ドメインが使用される)。リモート・サービスでは、ロード・バランシング (リモート・サービス・エントリの複数設定) とドメイン・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・ドメインの使用) の両方を使用できます。

ROUTING = *identifier*

複数のリモート・ドメインが同じサービスを提供する場合、このパラメータを指定すれば、ローカル・ドメインはデータ依存型ルーティングを実行できます。*identifier* は、このデータ依存型ルーティングで使用する基準名を指定します。このパラメータを指定しないと、データ依存型ルーティングはこのサービスで使用できません。*identifier* は 15 文字以内で指定します。サービス名またはキュー・スペース名が同じで、異なる RDOM パラメータを指定したエントリが複数存在する場合は、これらのすべてのエントリで同じ ROUTING パラメータを指定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 (*RDOM* を使用して) 設定されたリモート・ドメインは、このエントリが適用されるローカル・ドメインと同じ TP_SYSTEM の一部でなければなりません。

INBUFTYPE = *type*[:*subtype*]
: このサービスまたはキュー名が使用できる入力バッファ・タイプを、1 つのバッファ・タイプに制限します。BEA TOP END サービスおよびキュー名エントリでは、有効なタイプの値は FML32、CARRAY、および X_OCTET です。

OUTBUFTYPE = *type*[:*subtype*]
: このサービスから受け付けられる出力バッファ・タイプを、1 つのバッファ・タイプに制限します。BEA TOP END サービス・エントリでは、有効なタイプの値は FML32、CARRAY、および X_OCTET です。

CONV = {Y | N}
: 擬似会話を BEA TOP END サーバ・アプリケーションで管理する場合、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 に設定した場合は、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、要求 / 応答 XATMI 関数 (tpcall、tpacall) をこのサービスで使用しなければなりません。デフォルトは N です。

TRANTIME = *integer*
: 関連サービスに対して自動的に開始されたトランザクションに対して、デフォルトの間隔 (タイムアウト値) を秒単位で指定します。この値は、0 以上 2147483648 未満でなければなりません。デフォルトは 30 秒です。0 を指定すると、マシンの最大タイムアウト値、2147483648-1 に設定されます。

DM_RESOURCES

このオプションであるセクションでは、グローバル・ドメイン・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。

このセクションの唯一のパラメータは、次のとおりです。

VERSION = *string*

string は、ユーザがカレント・ドメイン・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。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_ROUTING セクション

このセクションでは、FML32 を使用した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提供します。この説明は TOPEND タイプのゲートウェイだけに適用されます。TDOMAIN タイプのゲートウェイのルーティング情報については、DMCONFIG(5) リファレンス・ページを参照してください。

DM_ROUTING セクションの行は、次の形式をとります。

CRITERION_NAME required_parameters

CRITERION_NAME は、サービス・エントリで指定されたルーティング・エントリの *identifier* の名前です。*CRITERION_NAME* は 15 文字以内で指定します。

以下は、必須パラメータです。

FIELD = *identifier*
: ルーティング・フィールドの名前を指定します。長さは、30 文字までです。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。環境変数 FLDTBLDIR32 と FIELDTBLS32 を使用して、FML フィールド・テーブルを見つけます。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、フィールド番号は 8191 以下でなければなりません。

RANGES = "*string*"
: 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・ドメイン名 (RDOM) を指定します。*string* は二重引用符で囲みます。*string* の値は、範囲 /RDOM のペアをカンマで区切って順番に並べます (このリファレンス・ページで後述する「使用例」の項を参照)。

  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)。

  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、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。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 (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(): 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ときに、関連する RDOM 値には、要求がルーティングされるリモート・ドメインを指定します。RDOM 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ることを示します。

範囲と RDOM のペアでは、範囲と RDOM がコロン (:) で区切られます。

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

これは、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 ドメイン・ゲートウェイの場合、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。

(FML32 バッファに対して) フィールド値が設定されていない場合、またはフィールド値が特定の範囲と一致しないで、ワイルドカード範囲が指定されていない場合は、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。

GWTOPEND(5) の DMCONFIG に関する追加情報

ファイル　　BDMCONFIG 環境変数を使用して、BDMCONFIG コンフィギュレーション・ファイルを検索します。

使用例　　以下の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 システムと単一接続を持つ TOP END ドメイン・ゲートウェイを含むように拡張されています。このシナリオで、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、ある一定の BEA Tuxedo サービスを、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ことが必要です。簡単なキューイングの例も含まれています。

このコンフィギュレーションの DMCONFIG ファイルを以下に示します。元のコア DMCONFIG(5) ファイルの使用例からの変更個所は、太字で示しています。

```
#  CENTRAL BANK 用の BEA Tuxedo ドメイン・コンフィギュレーション・ファイル
#
#
*DM_LOCAL_DOMAINS
# <local domain name> <Gateway Group name> <domain type> <domain id> <log device>
#       [<audit log>] [<blocktime>]
#       [<log name>] [<log offset>] [<log size>]
#       [<maxrdtran>] [<maxtran>]
#       [<maxdatalen>] [<security>]
#       [<tuxconfig>] [<tuxoffset>]
#
#
デフォルト :SECURITY = NONE


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

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

c03 GWGRP = bankg3
    TYPE = TOPEND
    DOMAINID = "CENTRALBKGW"
```

```
    DMTLOGDEV = "/usr/apps/bank/DMTLOG"
    DMTLOGNAME = "DMTLG_C03"
    SECURITY = CLEAR

#
*DM_REMOTE_DOMAINS
#<remote domain name> <domain type> <domain id>
#
b01  TYPE = TDOMAIN
     DOMAINID = "BA.BANK01"

b02  TYPE = TDOMAIN
     DOMAINID = "BA.BANK02"

b03  TYPE = TDOMAIN
     DOMAINID = "BA.BANK03"

b04  TYPE = OSITP
     DOMAINID = "BA.BANK04"

b05 TYPE = TOPEND
    DOMAINID = "BANK05"

*DM_TDOMAIN
#
# <local or remote domain name> <network address> [<nwdevice>]
#
# 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
#
#<local or remote domain name> <apt> <aeq>
#   [<aet>] [<acn>] [<apid>] [<aeid>]
#   [<profile>]
#
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_LOCAL_SERVICES
#<service_name> [<Local Domain name>] [<access control>] [<exported svcname>]
#       [<inbuftype>] [<outbuftype>]
#
# TOP END で利用不可、マッピングなし
open_act ACL = branch LDOM=c01
close_act ACL = branch LDOM=c01
credit LDOM=c01
debit LDOM=c01
loan   LDOM = 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" LDOM=c03

# TOP END で利用可能なキュー
qspace TYPE=QSPACE TE_RTQGROUP="TUXQUEUE" TE_RTQNAME="TUXQ" LDOM=c03
qname TYPE=QNAME TE_PRODUCT="TUX" TE_FUNCTION="QSERV" LDOM=c03

*DM_REMOTE_SERVICES
#<service_name> [<Remote domain name>] [<local domain name>]
#        [<remote svcname>] [<routing>] [<conv>]
#        [<trantime>] [<inbuftype>] [<outbuftype>]
#
tlr_add LDOM = c01 ROUTING = ACCOUNT
tlr_bal LDOM = c01 ROUTING = ACCOUNT
tlr_add RDOM = b04 LDOM = c02 RNAME ="TPSU002"
tlr_bal RDOM = b04 LDOM = c02 RNAME ="TPSU003"

#
#BEA Tuxedo で利用可能な新しい TOP END サービス
DEFAULT:LDOM=c03 RDOM=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"

#
# BEA Tuxedo で利用可能な TOP END RTQ キュー
DEFAULT:LDOM=c03 RDOM=b05 TYPE=QSPACE


tuxqspace TE_RTQGROUP="TEQGROUP" TE_RTQNAME="TEQNAME"

#
# tpenqueue と RTQ を介して BEA Tuxedo で利用可能な TOP END サービス
DEFAULT:LDOM=c03 RDOM=b05 TYPE=QNAME
te_report  TE_PRODUCT="EBANK" TE_FUNCTION="REPORT"
te_update  TE_PRODUCT="EBANK" TE_FUNCTION="UPDATE"

*DM_ROUTING
# <routing criteria> <field> <typed buffer> <ranges>
#
ACCOUNT FIELD = branchid BUFTYPE ="VIEW:account"
        RANGES ="MIN - 1000:b01, 1001-3000:b02, *:b03"

*DM_ACCESS_CONTROL
#<acl name> <Remote domain list>
#
branch ACLIST = b01, b02, b03
loans  ACLIST = b04
```

ネットワーク・アドレス

TEDG 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仮定します。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。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仮定します。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。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 ヘッダ、DOMAINS を含む */
```

Domains 関連の新しい用語

BEA Tuxedo リリース 7.1 では、Domains 用の MIB でクラスおよび属性の新しい用語を使用して、ドメイン間の相互作用を説明しています。新しい用語は DMCONFIG ファイルの構文にも適用されています。

「ドメイン」という用語の多用を避け、動作をより明確に記述する用語を使用するように改善されました。たとえば、「アクセス・ポイント」いう用語は、別のオブジェクトにアクセスするために通るオブジェクトを定義します。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・ポイントを通ってアクセスし、リモート・ドメインはローカル・ドメインにローカル・ドメイン・アクセス・ポイントを通ってアクセスします。次の表は、用語「ドメイン」の多用を避けた場合に、DMCONFIG セクション名がどのように変更されるかを反映しています。

DMCONFIG セクション名	変更後のセクション名
DM_LOCAL_DOMAINS	DM_LOCAL
DM_REMOTE_DOMAINS	DM_REMOTE

これらのセクション内で、以下のパラメータ名が変化しています。

パラメータ名	変更後のパラメータ名
DOMAINID	ACCESSPOINTID
MAXRDOM	MAXACCESSPOINT
MAXRDTRAN	MAXRAPTRAN

これらの DMCONFIG セクションに相当する DM_MIB クラスは、それぞれ T_DM_LOCAL と T_DM_REMOTE です。

一定のコンフィギュレーションでは、キュー・スペースとキュー名など、利用可能なサービスとリソースの両方をインポートおよびエクスポートしなければなりません。その場合、DMCONFIG セクション名 DM_LOCAL_SERVICES および DM_REMOTE_SERVICES は、必要なアクティビティを正確に記述していません。これらのセクション名をそれぞれ DM_EXPORT および DM_IMPORT に置き換えることにより、実行されるアクションが正確に記述されます。つまり、単一 BEA Tuxedo ドメインから見て、リソースはローカル・アクセス・ポイントを通してドメインからエクスポートされ、リモート・アクセス・ポイントを通してドメインにインポートされます。次の表は、これらの DMCONFIG セクション名の変更を示しています。

DMCONFIG セクション名	変更後のセクション名
DM_LOCAL_SERVICES	DM_EXPORT
DM_REMOTE_SERVICES	DM_IMPORT

これらのセクション内で、以下のパラメータ名が変化しています。

パラメータ名	変更後のパラメータ名
LDOM	LACCESSPOINT
RDOM	RACCESSPOINT

これらの DMCONFIG セクションに相当する 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 で採用されたドメイン関連の新しい用語は、以前と比べてより正確な表現になりました。ただし、ドメイン関連のマニュアルやエラー・メッセージへの影響はわずかです。新しい用語は、DM_MIB クラス、リファレンス・ページ、エラー・メッセージ、DMCONFIG ファイル構文、および各種 DMCONFIG エラー・メッセージで使用されています。

下位互換性のため、このリリース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。リリース 7.1 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 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際に使用するクラスのセットを定義します。このリファレンス・ページは、読者が BEA Tuxedo システムの Domains 機能を十分に理解していることを想定しています。

管理要求をフォーマットしたり管理応答の意味を解釈するには、DM_MIB(5) を共通の MIB リファレンス・ページ MIB(5) とともに使用する必要があります。

DM_MIB で記述されたクラスや属性を利用して MIB(5) の手順に従ってフォーマットした要求は、アクティブなアプリケーション中に存在するさまざまな ATMI インターフェイスを利用して管理サービスを要求するために使用できます。DM_MIB(5) のすべてのクラス定義に関連する追加情報については、178 ページ「DM_MIB(5) に関する追加情報」を参照してください。

DM_MIB(5) には次のクラスがあります。

DM_MIB クラス

クラス名	属性
T_DM_ACL	ドメイン・アクセス制御リスト
T_DM_CONNECTION	2 つのドメイン間の接続状態
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
T_DM_OSITPX	アクセス・ポイントの OSI TP 4.0 以降固有のコンフィギュレーション
T_DM_PASSWORD	ドメイン・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・ドメイン・コンフィギュレーション情報
T_DM_ROUTING	アクセス・ポイント・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	ローカル・アクセス・ポイントの SNA-CRM 固有のコンフィギュレーション

DM_MIB **クラス (続き)**

クラス名	属性
T_DM_SNALINK	リモート・アクセス・ポイントの SNAX 固有のコンフィギュレーション
T_DM_SNASTACK	特定の SNA CRM によって使用される SNA スタック
T_DM_TDOMAIN	アクセス・ポイントの Tdomain 固有のコンフィギュレーション
T_DM_TOPEND	アクセス・ポイントの BEA TOP END 固有のコンフィギュレーション
T_DM_TRANSACTION	ローカル・アクセス・ポイントと関連するトランザクション・エントリ

それぞれのクラスの説明は、4 つのセクションで構成されます。

- クラスの属性についての高レベルな説明
- 属性表 — クラスの各属性に関する名前、タイプ、パーミッション、値、およびデフォルト値を示す表。属性表の形式を以下に示します。
- 属性の意味 — クラスの各属性の意味を定義します。
- 制限事項 — このクラスの使用および解釈に関する制限事項

属性表の形式

属性表はクラス内の属性をリストし、さらに管理者、オペレータ、一般ユーザが属性を使用してアプリケーションとインターフェイスをとる方法を示しています。

属性表の各属性記述には 5 つの構成要素 (名前、タイプ、パーミッション、値、デフォルト値) があります。各項目については MIB(5) で説明します。

TA_FLAG 値

MIB(5) は、共通の TA_FLAGS 属性を定義します。共通およびコンポーネント MIB 固有のフラグ値の両方が入った long 値フィールドです。ここでは、DM_MIB 固有のフラグ値は定義していません。

FML32 フィールド・テーブル

このマニュアル・ページに記述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で区切ったリストに、またフィールド・テーブル名 tpadm は FIELDTBLS 環境変数で指定されるカンマで区切ったリストに、アプリケーションによって指定されなければなりません。

相互運用性

この MIB のヘッダ・ファイルやフィールド・テーブルには、BEA Tuxedo リリース 7.1 以降のサイト (ネイティブおよびワークステーションの両方) でのみアクセスできます。リリース 5.0 以前のサイトがアプリケーションでアクティブの場合、これらのサイトでゲートウェイ・グループがグローバル情報を更新すること (「SET」操作) はできません。

リリース 5.0 以前のサイトのローカル情報アクセスは利用できません。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。

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・ページで定義するヘッダ・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・プラットフォームおよびワークステーション・プラットフォームで利用可能です。

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

DM_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..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を作成するとき必要
(k) — オブジェクト検索用のキー・フィールド
(*) — クラスに対するすべての SET 操作で必要なキー・フィールド

属性の意味

TA_DMACLNAME:*string* [1..15]

アクセス制御リスト名。ドメイン・コンフィギュレーションの T_DM_ACL エントリ名の範囲内で一意です。

TA_DMRACCESSPOINTLIST:*string* [0..1550]

このアクセス制御リストと関連するリモート・ドメイン・アクセス・ポイントのリスト。TA_DMRACCESSPOINTLIST は、カンマで区切ったリモート・アクセス・ポイント名のリストです (つまり、有効な T_DM_REMOTE オブジェクトの TA_DMRACCESSPOINT 属性の値)。リストには、50 までのリモート・アクセス・ポイント識別子要素が含まれます。この属性を "*" に設定すると、コンフィギュレーション内のすべてのリモート・ドメインがこのエントリと関連することを意味します。"" は、このエントリと関連するリモート・アクセス・ポイントが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 オブジェクトに対するコンフィギュレーション情報を検索します。GET 要求に対して返される TA_STATE の意味を次に示します。リスト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、非アクティブです。このクラスでは、有効な状態は VALid だけです。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 オブジェクトに対するコンフィギュレーション情報を更新します。次に、SET 要求で TA_STATE に設定できる値の意味を説明します。記述されていない状態を設定することはできません。

"NEW"	新しいオブジェクトが作成されます。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	既存のオブジェクトを変更します。"INValid" 状態ではこの組み合わせは使用できません。正常終了すると、オブジェクトの状態は変わりません。
"INValid"	オブジェクトは削除されます。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　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・ポイント間の接続状態を表します。

属性表

DM_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: のとき使用可能な属性				
TA_DMCURENCRYPTBITS	string	r--------	"{0 | 40 | 56 | 128}" 注 1	"0"

(k) — オブジェクト検索用のキー・フィールド
(*) — クラスに対するすべての SET 操作で必要なキー・フィールド

注 1 リンク・レベルの暗号化の値 40 ビットは、下位互換性のために用意されています。

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 操作や SET 操作の実行時には、特定のローカル・ドメイン・アクセス・ポイントをこの属性に対して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 操作や SET 操作の実行時に、TA_DMRACCESSPOINT の値が指定されていないと、TA_DMLACCESSPOINT で指定されるローカル・アクセス・ポイントに対するすべての 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" 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。GET 要求に対して返される TA_STATE の意味を次に示します。リストされていない状態は返されません。

"ACTive"	接続はアクティブです。
"SUSpended"	接続は待機中です。
"INItializing"	接続は初期化中です。
"INActive"	指定されたドメイン・アクセス・ポイントの接続が解除されます。この状態は、BEA Tuxedo リリース 7.1 以降を実行するゲートウェイでしか返されません。
"UNKnown"	指定されたドメイン・アクセス・ポイントの接続状態は不明です。

SET:"{ACTive | INActive}"
SET 操作は、接続に対する実行時情報を更新します。次に、SET 要求で TA_STATE に設定できる値の意味を説明します。記述されていない状態を設定することはできません。

"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" で接続は "INItializing" になります。それ以外の場合、状態は変わりません。
"INActive"	指定されたドメイン・アクセス・ポイントの接続を解除し、オブジェクトを削除します。

TA_DMTYPE=TDOMAIN のとき使用可能な属性

TA_DMCURENCRYPTBITS: "{0 | 40 | 56 | 128}"
この接続で使用する暗号化レベル。"0" は暗号化を行わないことを示します。"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しか利用できません。それ以外のゲートウェイでは、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 ビットは、下位互換性のために用意されています。

制限事項

ドメイン・ゲートウェイ管理 (GWADM) サーバと TA_DMLACCESSPOINT で指定されるローカル・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ければ、そのアクセス・ポイントへの接続で GET 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・ポイントを通して 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

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]	* (すべての意)
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_DMRESOURCETYPE	string	rw-r--r--	"{SERVICE | QSPACE | QNAME}"	"SERVICE"
TA_DMREMOTENAME	string	rw-r--r--	*string*[1..30]	N/A
TA_DMTYPE=SNAX|OSITPX|TOPEND のリモート・アクセス・ポイントから使用可能な属性				
TA_DMINBUFTYPE	string	rw-r--r--	*string*[0..513]	N/A
TA_DMOUTBUFTYPE	string	rw-r--r--	*string*[0..513]	N/A

属性	タイプ	パーミッション	値	デフォルト値
TA_DMTYPE=OSITPX のリモート・アクセス・ポイントから使用可能な属性				
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]	""
TA_DMTYPE=TOPEND のリモート・アクセス・ポイントから使用可能な属性				
TA_DMTE_PRODUCT	string	rw-r--r--	*string*[1..32]	
TA_DMTE_FUNCTION	string	rw-r--r--	*string*[1..8]	
TA_DMTE_TARGET	string	rw-r--r--	*string*[1..8]	Spaces
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]	

(r) — 新しいオブジェクトを作成するとき必要
(k) — オブジェクト検索用のキー・フィールド
(*) — クラスに対するすべての SET 操作で必要なキー・フィールド

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) のエントリに対するローカル・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 T_SVCGRP オブジェクトの TA_SERVICENAME 属性の値と対応します。このリソースは、同じ名前または TA_DMREMOTENAME あるいは TA_DMTE* で定義されるエイリアスで、ほかのドメインにエクスポートされます。

TA_DMLACCESSPOINT:*string*[1..30]
: ローカル・アクセス・ポイント名。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を利用できることを意味します。

TA_STATE:

GET:"{VALid}"
: GET 操作は、T_DM_EXPORT オブジェクトに対するコンフィギュレーション情報を検索します。GET 要求に対して返される TA_STATE の意味を次に示します。リスト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 オブジェクトに対するコンフィギュレーション情報を更新します。次に、SET 要求で TA_STATE に設定できる値の意味を説明します。記述されていない状態を設定することはできません。

"NEW"	新しいオブジェクトが作成されます。
unset	既存のオブジェクトを変更します。"INValid" 状態ではこの組み合わせは使用できません。正常終了すると、オブジェクトの状態は変わりません。
"INValid"	オブジェクトは削除されます。

TA_DMACLNAME:*string*[1..15]

このローカル・サービスでセキュリティのために使用する T_DM_ACL オブジェクトの名前。TOPEND リモート・アクセス・ポイントからアクセスが許可されるとき、TA_DMRESOURCETYPE="SERVICE" または "QSPACE" であれば、このオプションの属性を指定できます。この属性は、TA_DMRESOURCETYPE="QNAME" の場合は使用できません。

TA_DMCONV:"{Y | N}"

このローカル・サービスが会話サービスであるかどうかを指定します。TOPEND リモート・アクセス・ポイントからアクセスが許可されるときは、TA_DMRESOURCETYPE="QSPACE" または "QNAME" のエントリに対して、この属性を "N" に設定する必要があります。

TA_DMRESOURCETYPE:"{SERVICE | QSPACE | QNAME}"

このエントリが "SERVICE"、"QSPACE"、または "QNAME" のいずれに対するものかを指定します。デフォルトは "SERVICE" です。

TA_DMREMOTENAME:*string*[1..30]

タイプ "SERVICE" または "QSPACE" のエントリに対して、この属性は TOPEND 以外のリモート・アクセス・ポイントを通してエクスポートされる名前を指定します。

TA_DMTYPE=SNAX| OSITPX| TOPEND のリモート・アクセス・ポイントから使用可能な属性

TA_DMINBUFTYPE:*string*[0..513]

type[:*subtype*]—入力バッファ・タイプ。オプションで後にサブタイプが続きます。この属性が存在すると、受け付けられるバッファ・タイプおよびサブタイプが定義されます。この属性は、UDT アプリケーション・コンテキストで OSITPX を使用してリモート・アクセス・ポイントからアクセスが許可されるとき、または SNAX を使用するとき、TA_DMRESOURCETYPE="SERVICE" のエントリに対して定義しなければなりません。TOPEND リモート・アクセス・ポイントからアクセスが許可されるとき、TA_DMRESOURCETYPE="SERVICE" または "QNAME" であれば、このオプションの属性を指定できます。この属性は、TA_DMRESOURCETYPE="QSPACE" の場合は使用できません。

BEA TOP END サービスおよびキュー名のエントリでは、有効なタイプの値は FML32、CARRAY、および X_OCTET です。

TA_DMOUTBUFTYPE:*string*[0..513]

type[:*subtype*]—出力バッファ・タイプ。オプションで後にサブタイプが続きます。この属性が存在すると、サービスによって出力されるバッファ・タイプおよびサブタイプが定義されます。この属性は、UDT アプリケーション・コンテキストで OSITPX を使用してリモート・アクセス・ポイントからアクセスが許可されるとき、または SNAX を使用するとき、TA_DMRESOURCETYPE="SERVICE" のエントリに対して定義しなければなりません。TOPEND リモート・アクセス・ポイントからアクセスが許可されるとき、TA_DMRESOURCETYPE="SERVICE" であれば、このオプションの属性を指定できます。この属性は、TA_DMRESOURCETYPE="QSPACE" または "QNAME" の場合は使用できません。

BEA TOP END サービスおよびキュー名のエントリでは、有効なタイプの値は FML32、CARRAY、および X_OCTET です。

TA_DMTYPE=OSITPX のリモート・アクセス・ポイントから使用可能な属性

TA_DMCOUPLING: *string*"{TIGHT|LOOSE}"

オプション。このパラメータの有効な値は、"TIGHT" または "LOOSE" です。

TA_DMINRECTYPE および TA_DMOUTRECTYPE: *string*[1..78]

オプション。これらのパラメータに対する入力は、type[:subtype] フィールドを含む TA_DMINBUFTYPE および TA_DMOUTBUFTYPE と同じです。

TA_DMTYPE=TOPEND のリモート・アクセス・ポイントから使用可能な属性

TA_DMTE_PRODUCT:*string*[1..32]
: BEA TOP END 製品名。TOPEND リモート・アクセス・ポイントからアクセスが許可されるとき、TA_DMRESOURCETYPE="SERVICE" および "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" の場合は使用できません。

TA_DMTE_FUNCTION:*string*[1..8]
: BEA TOP END 関数名。TOPEND リモート・アクセス・ポイントからアクセスが許可されるとき、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" の場合は使用できません。

TA_DMTE_TARGET:*string*[1..8]
: 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、TOPEND リモート・アクセス・ポイントからアクセスが許可されるとき、TA_DMRESOURCETYPE="SERVICE"、"QSPACE"、または "QNAME" のエントリに対してオプションです。

TA_DMTE_QUALIFIER:0 <= *num* <= MAXLONG
: BEA TOP END 関数修飾子。この属性は、TOPEND リモート・アクセス・ポイントからアクセスが許可されるとき、TA_DMRESOURCETYPE="SERVICE" または "QNAME" のエントリに対してオプションです。この属性は、TA_DMRESOURCETYPE="QSPACE" の場合は使用できません。

TA_DMTE_RTQGROUP:*string*[1..32]
: BEA TOP END リカバリ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・アクセス・ポイントからアクセスが許可されるとき、TA_DMRESOURCETYPE="QSPACE" に対して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" の場合は使用できません。

TA_DMTE_RTQNAME:*string*[1..8]
: BEA TOP END キュー名。TOPEND リモート・アクセス・ポイントからアクセスが許可されるとき、TA_DMRESOURCETYPE="QSPACE" に対して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" の場合は使用できません。

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・ドメインで利用可能になるリモート・リソースを表します。

属性表

DM_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_DMLACCESSPOINT(k)(*)	string	rw-r--r--	*string*[1..30]	* (すべての意)
TA_DMRACCESSPOINTLIST(k)(*)	string	rw-r--r--	*string*[1..92]	* (すべての意)
TA_STATE(r)	string	rwxr-xr--	GET:"VAL" SET:"{NEW | INV}"	N/A N/A
TA_DMAUTOTRAN	string	rw-r--r--	"{Y | N}"	"N"
TA_DMCONV	string	rw-r--r--	"{Y | N}"	"N"
TA_DMLOAD	short	rw-r--r--	1 <= *num* <= 32,767	50
TA_DMPRIO	short	rw-r--r--	1 <= *num* <= 100	50
TA_DMRESOURCETYPE	string	rw-r--r--	"{SERVICE | QSPACE | QNAME}"	"SERVICE"
TA_DMREMOTENAME	string	rw-r--r--	*string*[1..30]	N/A
TA_DMROUTINGNAME	string	rw-r--r--	*string*[1..15]	N/A
TA_DMTRANTIME	long	rw-r--r--	0 <= *num* <= 32,767	30
TA_DMTYPE=SNAX|OSITPX|TOPEND のリモート・アクセス・ポイントから使用可能な属性				
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・アクセス・ポイントから使用可能な属性				

DM_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_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]	""
TA_DMTYPE=TOPEND のリモート・アクセス・ポイントから使用可能な属性				
TA_DMTE_PRODUCT	string	rw-r--r--	*string*[1..32]	
TA_DMTE_FUNCTION	string	rw-r--r--	*string*[1..8]	
TA_DMTE_TARGET	string	rw-r--r--	*string*[1..8]	空白
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]	

(r) — 新しいオブジェクトを作成するとき必要
(k) — オブジェクト検索用のキー・フィールド
(*) — クラスに対するすべての SET 操作で必要なキー・フィールド

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) のエントリに対するリモート・リソースの名前。このリソースは、同じ名前または TA_DMREMOTENAME か TA_DMTE* で定義されたエイリアスのリモート・ドメインからインポートされます。

TA_DMLACCESSPOINT:*string*[1..30]
: このインポートされたリソースを利用できるようにするためのローカル・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・アクセス・ポイントを通してリソースを利用できます。

TA_DMRACCESSPOINTLIST:*string*[1..92]
: このリソースをインポートするリモート・ドメイン・アクセス・ポイントを識別します。TA_DMRACCESSPOINTLIST は、カンマで区切ったフェイルオーバー・ドメインのリストです。30 文字以内のリモート・ドメイン・アクセス・ポイントを 3 つまで指定できます。この属性を "*" に設定すると、すべてのリモート・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_STATE:

GET:"{VALid}"
: GET 操作は、T_DM_IMPORT オブジェクトに対するコンフィギュレーション情報を検索します。GET 要求に対して返される TA_STATE の意味を次に示します。記述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 オブジェクトに対するコンフィギュレーション情報を更新します。次に、SET 要求で TA_STATE に設定できる値の意味を説明します。記述されていない状態を設定することはできません。

"NEW"	新しいオブジェクトが作成されます。状態の変更は "INValid" 状態でのみ可能で、結果は "ACTive" 状態になります。
unset	既存のオブジェクトを変更します。"INValid" 状態ではこの組み合わせは使用できません。正常終了すると、オブジェクトの状態は変わりません。
"INValid"	オブジェクトは削除されます。状態の変更は "ACTive" 状態でのみ可能で、結果は "INValid" 状態になります。

TA_DMAUTOTRAN:"{Y | N}"
: まだトランザクション内にないリソースに対して要求が受信されると、この属性は自動的にこのリソースに対してトランザクションを開始します。デフォルトは "N" です。

TA_DMCONV:"{Y | N}"
: このサービスが会話サービス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・アクセス・ポイントからアクセスが許可されるときは、TA_DMRESOURCETYPE="QSPACE" または "QNAME" のエントリに対して、この属性を "N" に設定する必要があります。

TA_DMLOAD:1 <= *num* <= 32,767
: サービス負荷。

TA_DMPRIO:1 <= *num* <= 100
: キューから取り出す際の優先順位。優先順位の高い要求からサービスが提供されます。

TA_DMRESOURCETYPE:"{SERVICE | QSPACE | QNAME}"
: このエントリが "SERVICE"、"QSPACE"、または "QNAME" のいずれに対するものかを指定します。デフォルトは "SERVICE" です。

TA_DMREMOTENAME:*string*[1..30]
: タイプ "SERVICE" または "QSPACE" のエントリに対して、この属性は TOPEND でないリモート・アクセス・ポイントを通してインポートされる名前を指定します。

TA_DMROUTINGNAME:*string*[1..15]
: この "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_DMTRANTIME:1 <= *num* <= 32,767
: この "SERVICE" または "QSPACE" に対して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 (秒)。トランザクション・モード以外の要求が受信され、TA_DMAUTOTRAN 属性が "Y" のとき、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更新しても、アクティブな要求には反映されません。

TA_DMTYPE=SNAX| OSITPX| TOPEND のリモート・アクセス・ポイントから使用可能な属性

TA_DMINBUFTYPE:*string*[0..256]
: *type*[:*subtype*]- 入力バッファ・タイプ。オプションで後にサブタイプが続きます。この属性が存在すると、受け付けられるバッファ・タイプおよびサブタイプが定義されます。この属性は、UDT アプリケーション・コンテキストで OSITPX を使用してリモート・アクセス・ポイントにアクセスが許可されるとき、または SNAX を使用するとき、DMRESOURCETYPE="SERVICE" のエントリに対して定義する必要があります。TOPEND リモート・アクセス・ポイントからアクセスが許可されるとき、TA_DMRESOURCETYPE="SERVICE" または "QNAME" であれば、このオプションの属性を指定できます。この属性は、TA_DMRESOURCETYPE="QSPACE" の場合は使用できません。

  BEA TOP END サービスおよびキュー名のエントリでは、有効なタイプの値は FML32、CARRAY、および X_OCTET です。

TA_DMOUTBUFTYPE:*string*[0..256]

type[:*subtype*]- 出力バッファ・タイプ。オプションで後にサブタイプが続きます。この属性が存在すると、サービスによって出力されるバッファ・タイプおよびサブタイプが定義されます。この属性は、UDT アプリケーション・コンテキストで OSITPX を使用してリモート・アクセス・ポイントにアクセスが許可されるとき、または SNAX を使用するとき、DMTYPE="SERVICE" のエントリに対して定義する必要があります。TOPEND リモート・アクセス・ポイントからアクセスが許可されるとき、TA_DMRESOURCETYPE="SERVICE" であれば、このオプションの属性を指定できます。この属性は、TA_DMRESOURCETYPE="QSPACE" または "QNAME" の場合は使用できません。

BEA TOP END サービスおよびキュー名のエントリでは、有効なタイプの値は FML32、CARRAY、および X_OCTET です。

TA_DMTYPE=OSITPX のリモート・アクセス・ポイントから使用可能な属性

TA_DMAUTOPREPARE: *string*"{Y|N}"

オプション。このパラメータの有効な値は、"Y" または "N" です。

TA_DMINRECTYPE および TA_DMOUTRECTYPE: *string*[1..78]

オプション。これらのパラメータに対する入力は、type[:subtype] フィールドを含む TA_DMINBUFTYPE および TA_DMOUTBUFTYPE と同じです。

TA_DMTPSUTTYPE: *string*"{INTEGER|PRINTABLESTRING"}

オプション。このパラメータに対する入力は、"INTEGER" または "PRINTABLESTRING" です。

TA_DMREMTPSUT: *string*[1..64]

オプション。TA_DMTPSUTTYPE パラメータが "INTEGER" に設定されている場合、このパラメータには MINLONG から MAXLONG までの範囲内の整数値を指定する必要があります。TA_DMTPSUTTYPE パラメータが "PRINTABLESTRING" に設定されている場合、このパラメータには PRINTABLESTRING 形式の TPSU-Title を 64 文字以内で指定する必要があります。

TA_DMTYPE=TOPEND のリモート・アクセス・ポイントから使用可能な属性

TA_DMTE_PRODUCT:*string*[1..32]

BEA TOP END 製品名。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" の場合は使用できません。

TA_DMTE_FUNCTION:*string*[1..8]
: BEA TOP END 関数名。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" の場合は使用できません。

TA_DMTE_TARGET:*string*[1..8]
: 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、TA_DMRESOURCETYPE ="SERVICE"、"QSPACE"、または "QNAME" のエントリに対してオプションです。

TA_DMTE_QUALIFIER:0 <= *num* <= MAXLONG
: BEA TOP END 関数修飾子。この属性は、TA_DMRESOURCETYPE="SERVICE" および "QNAME" のエントリに対してオプションです。この属性は、TA_DMRESOURCETYPE="QSPACE" のエントリに対しては使用できません。

TA_DMTE_RTQGROUP:*string*[1..32]
: BEA TOP END リカバリ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。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" の場合は使用できません。

TA_DMTE_RTQNAME:*string*[1..8]
: BEA TOP END RTQ キュー名。TA_DMRESOURCETYPE="QSPACE" であれば、この属性を指定する必要があります。TA_DMRESOURCETYPE="SERVICE" または "QNAME" の場合は使用できません。

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されます。

属性表

DM_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|OSITPX|SNAX | TOPEND}"	"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..78]	N/A
TA_DMBLOCKTIME	short	rw-r--r--	0 <= *num* <= 32,767	TA_BLOCKTIME in T_DOMAIN 注1
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
TA_DMSECURITY	string	rw-r--r--	"{NONE | APP_PW | DM_PW | DM_USER_PW | CLEAR | SAFE | PRIVATE}"	"NONE"
TA_DMTLOGDEV	string	rw-r--r--	*string*[1..78]	N/A
TA_DMTLOGNAME	string	rw-r--r--	*string*[1..30]	"DMTLOG"
TA_DMTLOGSIZE	long	rw-r--r--	1 <= *num* <= 2048	100
TA_DMTYPE=TDOMAIN|TOPEND のとき使用可能な属性				
TA_DMCONNECTION_POLICY	string	rwxr--r--	"{ON_DEMAND|ON_STARTUP | INCOMING_ONLY}"	"ON_DEMAND"
TA_DMRETRY_INTERVAL	long	rwxr--r--	0 <= *num* <= MAXLONG	60
TA_DMMAXRETRY	long	rwxr--r--	0 <= *num* <= MAXLONG	0
TA_DMCONNPRINCIPALNAME	string	rwxr--r--	*string*[0..511]	""

DM_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_DMMACHINETYPE	string	rw-r--r--	*string*[0..15]	""
TA_DMTYPE=OSITPX|TOPEND のとき使用可能な属性				
TA_DMBLOB_SHM_SIZE	long	rw-r--r--	1 <= *num* <= MAXLONG	1000000

(r) — 新しいオブジェクトを作成するとき必要
(k) — オブジェクト検索用のキー・フィールド
(*) — クラスに対するすべての SET 操作で必要なキー・フィールド

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

属性の意味

TA_DMACCESSPOINT:*string*[1..30]
この T_DM_LOCAL エントリの名前。ドメイン・コンフィギュレーション内の T_DM_LOCAL および T_DM_REMOTE エントリ名の範囲内で一意な識別子です。

TA_DMACCESSPOINTID:*string*[1..30]
ドメイン・アクセス・ポイント識別子。この識別子は、すべてのローカル・ドメイン・アクセス・ポイントおよびリモート・ドメイン・アクセス・ポイントで一意な値です。

TA_DMSRVGROUP:*string*[1..30]
ローカル・ドメインの管理サーバとゲートウェイ・プロセスが存在するグループ名。

TA_DMTYPE:"{TDOMAIN | OSITPX | SNAX | TOPEND}"
ドメインのタイプを指定します。BEA Tuxedo システムのドメインの場合は "TDOMAIN"、OSI TP 4.0 以降のドメインの場合は "OSITPX"、SNA ドメインの場合は "SNAX"、BEA TOP END ドメインの場合は "TOPEND" 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_STATE:

GET: "{VALid}"
GET 操作は、T_DM_LOCAL オブジェクトに対するコンフィギュレーション情報を検索します。GET 要求に対して返される TA_STATE の意味を次に示します。記述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 オブジェクトに対するコンフィギュレーション情報を更新します。次に、SET 要求で TA_STATE に設定できる値の意味を説明します。記述されていない状態を設定することはできません。

"NEW"	新しいオブジェクトが作成されます。状態の変更は "INValid" 状態でのみ可能で、結果は "VALid" 状態になります。
unset	既存のオブジェクトを変更します。"INValid" 状態ではこの組み合わせは使用できません。正常終了すると、オブジェクトの状態は変わりません。
"INValid"	オブジェクトは削除されます。状態の変更は "VALid" 状態でのみ可能で、結果は "INValid" 状態になります。

TA_DMAUDITLOG:*string*[1..78]

このローカル・ドメインに対する監査ログ・ファイルの名前。

TA_DMBLOCKTIME:0 <= *num* <= 32,767

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 オブジェクトに指定した SCANUNIT パラメータの乗数を設定します。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 オブジェクトで指定した TA_BLOCKTIME 属性の値がデフォルトとして使用されます。タイムアウトの発生は、関連する要求が失敗したことを示します。トランザクション内で要求が発行されると必ず、T_DOMAIN でトランザクションに対して指定したタイムアウトが使用されます。

TA_DMMAXRAPTRAN:0 <= *num* <= 32,767

単一トランザクションに関与できるリモート・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・ポイントで同時に開始できるトランザクションの最大数。数値には、T_DOMAIN.TA_MAXGTT 以上の値を指定します。

TA_DMSECURITY:"{NONE | APP_PW | DM_PW | DM_USER_PW | CLEAR | SAFE | PRIVATE}"

このドメインで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"

セキュリティは有効になりません。

"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・パスワードの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・パスワードの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_DMTLOGDEV:*string*[1..78]

このローカル・ドメイン・アクセス・ポイントのドメイン TLOG を収めたデバイス (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。高信頼性を得るため、デバイス (raw スライス) の使用をお勧めします。

TA_DMTLOGNAME:*string*[1..30]
: このローカル・ドメイン・アクセス・ポイントのドメイン TLOG 名。同じデバイス上に複数の TLOG がある場合は、それぞれの TLOG が一意の名前を持ってい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・ポイントの TLOG のサイズ (ページ数)。サイズは、TA_DMTLOGDEV で識別されるデバイスで利用可能なスペースによって制限されます。

TA_DMTYPE = TDOMAIN | TOPEND のとき使用可能な属性

TA_DMCONNECTION_POLICY:"{ON_DEMAND | ON_STARTUP | INCOMING_ONLY}"
: ローカル・ドメイン・ゲートウェイがリモート・ドメインに対して接続を試行するときの条件を指定します。指定可能な値は、"ON_DEMAND"、"ON_STARTUP"、または "INCOMING_ONLY" のいずれかです。

  "ON_DEMAND"
  : クライアントがリモート・サービスを要求したとき、または管理コマンド "connect" が実行されたときのいずれかで、接続が試行されます。TA_DMCONNECTION_POLICY のデフォルト設定は、"ON_DEMAND" です。"ON_DEMAND" 接続方針では、TA_DMCONNECTION_POLICY を明示的に利用できなかった以前のリリースと同等の振る舞いが行われます。この接続方針が指定されていると、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・アクセス・ポイントとの接続を試みます。そのリモート・ドメイン・アクセス・ポイントへの接続が確立された場合にのみ、リモート・サービス (このローカル・アクセス・ポイントのドメイン・ゲートウェイによって宣言されたサービス) は宣言されます。つまり、リモート・ドメイン・アクセス・ポイントへ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ます。TA_DMMAXRETRY および TA_DMRETRY_INTERVAL 属性を使用して、再接続の間隔を変更することもできます。

"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・アクセス・ポイントへの接続を試みません。そのため、初期のリモート・サービス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・アクセス・ポイントからの受信時接続で利用できます。また、リモート・サービスは、このローカル・ドメイン・アクセス・ポイントのドメイン・ゲートウェイが受信時接続を受信したときに宣言されます。接続方針が "INCOMING_ONLY" に指定されていると、再接続は行われません。

TA_DMRETRY_INTERVAL: 0 <= *num* <= MAXLONG

リモート・ドメイン・アクセス・ポイントへの自動接続が試行される間隔を秒単位で指定します。最小値は 0 であり、最大値は 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 は指定できません。

この属性は、TA_DMCONNECTION_POLICY 属性が "ON_STARTUP" に設定されている場合のみ有効です。その他の接続方針では、自動再接続処理機能は無効です。

TA_DMMAXRETRY: 0 <= *num* <= MAXLONG

ドメイン・ゲートウェイがリモート・ドメイン・アクセス・ポイントへ再接続を試みる回数を指定します。最小値は 0 であり、最大値は MAXLONG です。MAXLONG を指定すると、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。接続方針が "ON_STARTUP" の場合、TA_DMMAXRETRY のデフォルトは MAXLONG です。この属性に 0 を指定すると、自動再接続は行われません。その他の接続方針では、自動再接続処理機能は無効です。

TA_DMMAXRETRY 属性は、接続方針が "ON_STARTUP" の場合のみ有効です。

TA_DMCONNPRINCIPALNAME:string[0..511]

接続プリンシパル名識別子。リモート・ドメイン・アクセス・ポイントへの接続を確立するとき、このローカル・ドメイン・アクセス・ポイントのアイデンテイティを確認するために使用する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を実行する TDOMAIN タイプのドメインだけに適用されます。

TA_DMCONNPRINCIPALNAME 属性には、最大 511 文字を指定できます (最後のヌル文字を除く)。この属性を指定しないと、接続プリンシパル名はデフォルトにより、このローカル・ドメイン・アクセス・ポイントの TA_DMACCESSPOINTID 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 TA_DMCONNPRINCIPALNAME 属性に値を割り当てるときは、このローカル・ドメイン・アクセス・ポイントの TA_DMACCESSPOINTID 属性に割り当てられる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: Unable to acquire credentials.
```

TA_DMMACHINETYPE: *string*[0..15]
: ドメインをグループ化し、ドメイン間のメッセージの符号化と復号化を省略できるようにします。
domains can be bypassed.TA_DMMACHINETYPE が指定されていない場合、デフォルトで符号化や復号化が実行されます。接続の T_DM_LOCAL クラスと T_DM_REMOTE クラスの TA_DMMACHINETYPE フィールドに同じ値を設定した場合、データの符号化と復号化は省略されます。TA_DMMACHINETYPE の値には、15 文字までの文字列値を設定できます。この値は比較のためだけに使用します。

この属性は TA_DMTYPE=TDOMAIN の場合のみ有効です。

TA_DMTYPE = OSITPX | TOPEND のとき使用可能な属性

TA_DMBLOB_SHM_SIZE:1 <= *num* <= MAXLONG
: ローカル・ドメイン・アクセス・ポイントのエントリだけに関連します。OSITPX または TOPEND に固有の、大きなバイナリ・オブジェクト・ログ情報を格納するために割り当てられる共用メモリの容量を指定します。

制限事項

TA_DMLACCESSPOINT 属性で指定されるローカル・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のとき、SET を実行して TA_STATE を INValid にしたり、次の属性を更新することはできません。TA_DMACCESSPOINTID、TA_DMMAXRAPTRAN、TA_DMMAXTRAN、TA_DMSRVGROUP、TA_DMTYPE、TA_DMTLOGDEV、TA_DMTLOGNAME、TA_DMTLOGSIZE、TA_DMMACHINETYPE、または TA_DMCODEPAGE。

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) 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

DM_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_DMMAXENCRYPTBITS	short	rw-r--r--	"{0}"	"0"
TA_DMMINENCRYPTBITS	short	rw-r--r--	"{0}"	"0"
TA_DMMULTIPLEXING	string	rw-r--r--	0<= *num* <= 32767	0
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を作成するとき必要
(k) — オブジェクト検索用のキー・フィールド
(*) — クラスに対するすべての SET 操作で必要なキー・フィールド

属性の意味

TA_DMACCESSPOINT:*string*[1..30]
このエントリが提供するプロトコル固有のコンフィギュレーション情報の対象であ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・ポイントに対してプロトコル非関連の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エントリで与えられるドメイン・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オブジェクトに対するコンフィギュレーション情報を検索します。GET要求に対して返されるTA_STATEの意味を次に示します。リスト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オブジェクトに対するコンフィギュレーション情報を更新します。次に、SET要求でTA_STATEに設定できる値の意味を説明します。記述されていない状態を設定することはできません。

"NEW"	新しいT_DM_OSITPXオブジェクトが作成されます。状態の変更は"INValid"状態でのみ可能で、結果は"VALid"状態になります。
unset	既存のT_DM_OSITPXオブジェクトを変更します。"INValid"状態ではこの組み合わせは使用できません。正常終了すると、オブジェクトの状態は変わりません。
"INValid"	T_DM_OSITPXオブジェクトは削除されます。状態の変更は"VALid"状態でのみ可能で、結果は"INValid"状態になります。

TA_DMAET: *string*[1..78]
ドメイン・アクセス・ポイントのアプリケーション構成要素タイトル。この属性は、アプリケーション・プロセス・タイトルをオブジェクト識別子として含みます。オブジェクト識別子の後には、次に示すように、アプリケーション構成要素修飾子が整数値として続きます。
{object identifier},{integer qualifier}

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・アドレスのリスト。セミコロンで区切ります。この属性では次の形式を使用します。

address[,{IP|CONS|CLNS}][;address[,{IP|CONS|CLNS}]...]

リストには、アドレスとその後にオプションのアドレス・タイプを指定します。オプションのアドレス・タイプを指定していない場合は、デフォルト値として IP が設定されます。アドレス・タイプが IP の場合、ネットワーク・アドレスは次のいずれかの形式で指定します。

//hostname[:port_number]

#.#.#.#[:port_number]

port_number を指定していない場合は、デフォルト値として RFC 1006 ポート "102" が使用されます。アドレス・タイプが CONS または CLNS の場合は、たとえば、X、121、または先頭に 0x が付く 1 ～ 20 の 16 進数オクテットで構成される **NSAP** アドレスのいずれかを使用できます。

0x990001001200000 1234abcde9900

ローカル・ドメイン・アクセス・ポイントの場合、最大 8 個の待機用アドレスを指定できます。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]

このドメイン・アクセス・ポイントで使用する Transport Service Access Point アドレス。1 ～ 32 の ASCII 非制御文字か、先頭に 0x が付く 1 ～ 32 の 16 進数オクテットのいずれかで指定します。

TA_DMDNRESOLUTION: "{STARTUP | RUNTIME}"

アドレス解決のために DNS が使用される際の、ホスト名形式のネットワーク・アドレスを定義します。これが STARTUP (デフォルト) に設定されている場合、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。

注記 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関連し、リモート・ドメイン・アクセス・ポイントに対しては無視されます。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、この属性はヌル文字列に設定されます。

TA_DMMAXENCRYPTBITS: "{0}"

この属性は現在使用されていません。これは今後の OSITPX への拡張用に予約されており、最大暗号化レベルのネゴシエートに使用される予定です。

`TA_DMMINENCRYPTBITS: "{0}"`

この属性は現在使用されていません。これは今後の OSITPX への拡張用に予約されており、最小暗号化レベルのネゴシエートに使用される予定です。

`TA_DMMULTIPLEXING: 0 <= num <= 32767`

この属性は現在使用されていません。これは今後の OSITPX への拡張用に予約されており、目的のリモート・ドメイン・アクセス・ポイントに対して実行する多重化のレベルの指定に使用される予定です。

`TA_DMPSEL: string[1..10]`

これはオプションの属性です。このドメイン・アクセス・ポイントで使用する Presentation Service Access Point アドレスを指定します。1 ～ 4 の ASCII 非制御文字か、先頭に `0x` が付く 1 ～ 4 の 16 進数オクテットのいずれかで指定します。

`TA_DMSSEL: string[1..34]`

これはオプションの属性です。このドメイン・アクセス・ポイントで使用する Session Service Access Point アドレスを指定します。1 ～ 16 の ASCII 非制御文字か、先頭に `0x` が付く 1 ～ 16 の 16 進数オクテットのいずれかで指定します。

`TA_DMTAILORPATH: string[1..78]`

これはオプションの属性です。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 OSI TP 指定ファイルのフルパス名を指定します。この属性に値を指定していない場合、または値をヌル文字列に設定している場合は、OSI TP スタックは調整パラメータのデフォルト値を使用して実行されます。

注記 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関連し、リモート・ドメイン・アクセス・ポイントに対しては無視されます。

```
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。このパラメータは RDROM の記述でのみ有効です。有効な値は次のとおりです。

`"CAE"`

`"PRELIMINARY"`

`"OLTP_TM2200"`

`"NATIVE_A_SERIES"`

注記 この属性はリモート・ドメイン・アクセス・ポイントにのみ関連し、ローカル・ドメイン・アクセス・ポイントに対しては無視されます。

TA_DMEXTENSIONS: *string*[1..78]
: これはオプションの属性です。これは今後の OSITPX への拡張用に予約されている文字列で、LDOM または RDOM セクションへの新しいパラメータの追加で使用される予定です。

TA_DMOPTIONS: "{SECURITY_SUPPORTED}"
: これはオプションの属性です。DM_OSITPX 構造体のビット・マスク・フィールドです。このパラメータは RDOM の記述でのみ有効です。有効な値は SECURITY_SUPPORTED です。

制限事項

以下のシナリオでは、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・ポイントに関連し、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。
- 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・ポイントに関連し、任意の OSI TP 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。

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 属性で指定される特定のドメイン・アクセス・ポイントが、T_DM_LOCAL クラスか T_DM_REMOTE クラスのいずれか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_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_PASSWORD クラスは、タイプ TDOMAIN のアクセス・ポイントを通るドメイン間認証に対するコンフィギュレーション情報を表します。

属性表

DM_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

DM_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_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を作成するとき必要
(k) — オブジェクト検索用のキー・フィールド
(*) — クラスに対するすべての SET 操作で必要なキー・フィールド

属性の意味

TA_DMLACCESSPOINT:*string*[1..30]
パスワードが適用されるローカル・ドメイン・アクセス・ポイントの名前。

TA_DMRACCESSPOINT:*string*[1..30]
パスワードが適用されるリモート・ドメイン・アクセス・ポイントの名前。

TA_DMLPWD:*string*[1..30]
TA_DMLACCESSPOINT で識別されるローカル・ドメイン・アクセス・ポイントと TA_DMRACCESSPOINT で識別されるリモート・ドメイン・アクセス・ポイント間の接続の認証に使用されるローカル・パスワード。

TA_DMRPWD:*string*[1..30]
TA_DMLACCESSPOINT で識別されるローカル・ドメイン・アクセス・ポイントと TA_DMRACCESSPOINT で識別されるリモート・ドメイン・アクセス・ポイント間の接続の認証に使用されるリモート・パスワード。

TA_STATE:

GET:"{VALid}"
GET 操作は、選択した T_DM_PASSWORD オブジェクトに対するコンフィギュレーション情報を検索します。GET 要求に対して返される TA_STATE の意味を次に示します。リスト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 オブジェクトに対するコンフィギュレーション情報を更新します。次に、SET 要求で TA_STATE に設定できる値の意味を説明します。記述されていない状態を設定することはできません。

"NEW"	新しいオブジェクトが作成されます。状態の変更は "INValid" 状態でのみ可能で、結果は "VALid" 状態になります。
unset	既存のオブジェクトを変更します。"INValid" 状態ではこの組み合わせは使用できません。
"INValid"	オブジェクトは削除されます。状態の変更は "VALid" 状態でのみ可能で、結果は "INValid" 状態になります。
"RECrypt"	新しい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 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

任意のドメイン・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する (SET TA_STATE を "RECrypt" に設定する) 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

DM_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

DM_MIB(5): T_DM_PRINCIPAL_MAP **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を作成するとき必要
(k) — オブジェクト検索用のキー・フィールド
(*) — クラスに対するすべての SET 操作で必要なキー・フィールド

属性の意味

TA_DMLACCESSPOINT:*string*[1..30]
プリンシパル・マッピングが適用されるローカル・ドメイン・アクセス・ポイント。

TA_DMRACCESSPOINT:*string*[1..30]
プリンシパル・マッピングが適用されるリモート・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・マッピングが適用される方向。

"IN"
これは、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・ポイント経由の BEA Tuxedo ドメインへの Incoming であることを示します。

"OUT"
これは、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・ポイント経由の BEA Tuxedo ドメインからの 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 エントリに対するコンフィギュレーション情報を検索します。GET 要求に対して返される TA_STATE の意味を次に示します。記述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 エントリに対するコンフィギュレーション情報を更新します。次に、SET 要求で TA_STATE に設定できる値の意味を説明します。記述されていない状態を設定することはできません。

"NEW"	新しいオブジェクトが作成されます。状態の変更は "INValid" 状態でのみ可能で、結果は "VALid" 状態になります。
unset	既存のオブジェクトを変更します。"INValid" 状態ではこの組み合わせは使用できません。
"INValid"	オブジェクトは削除されます。状態の変更は "VALid" 状態でのみ可能で、結果は "INValid" 状態になります。

制限事項

BEA Tuxedo リリース 7.1 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・ドメインでアクセス可能になります。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

DM_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 | OSITPX | SNAX | TOPEND}"	"TDOMAIN"
TA_STATE(r)	string	rw-r--r--	GET: "VAL" SET: "{NEW | INV}"	N/A N/A

DM_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_DMPRIORITY_TYPE	string	rw-r--r--	"{LOCAL_RELATIVE | LOCAL_ABSOLUTE | GLOBAL}"	"LOCAL_ RELATIVE"
TA_DMINPRIORITY	string	rw-r--r--	-99<=num<=99	0 or 50 (see text)
TA_DMTYPE=SNAX および OSITPX のとき使用可能な属性				
TA_DMCODEPAGE	string	rw-r--r--	*string*[1..20]	N/A
TA_DMTYPE=TDOMAIN and OSITPX のとき使用可能な属性				
TA_DMACLPOLICY	string	rwxr--r--	"{LOCAL | GLOBAL}"	"LOCAL"
TA_DMLOCALPRINCIPALNAME	string	rwxr--r--	*string*[0..511]	""
TA_DMTYPE=TDOMAIN のとき使用可能な属性				
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]	""

(r) — 新しいオブジェクトを作成するとき必要
(k) — オブジェクト検索用のキー・フィールド
(*) — クラスに対するすべての SET 操作で必要なキー・フィールド

属性の意味

TA_DMACCESSPOINT:*string*[1..30]
この T_DM_REMOTE エントリの名前。TA_DMACCESSPOINT は、ドメイン・コンフィギュレーション内の T_DM_LOCAL および T_DM_REMOTE エントリ名の範囲内で一意な識別子です。

TA_DMACCESSPOINTID:*string*[1..30]
アクセス・ポイント識別子。この識別子は、すべてのローカル・ドメイン・アクセス・ポイントおよびリモート・ドメイン・アクセス・ポイントで一意な値です。

TA_DMTYPE:"{TDOMAIN | OSITPX | SNAX | TOPEND}"

ドメインのタイプを指定します。BEA Tuxedo システムのドメインの場合は "TDOMAIN" 、OSI TP 4.0 以降のドメインの場合は "OSITPX" 、SNA ドメインの場合は "SNAX" 、BEA TOP END ドメインの場合は "TOPEND" 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_STATE:

GET:"{VALid}"

GET 操作は、T_DM_REMOTE オブジェクトに対するコンフィギュレーション情報を検索します。GET 要求に対して返される TA_STATE の意味を次に示します。記述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 オブジェクトに対するコンフィギュレーション情報を更新します。次に、SET 要求で TA_STATE に設定できる値の意味を説明します。記述されていない状態を設定することはできません。

"NEW"	新しいオブジェクトが作成されます。
unset	既存のオブジェクトを変更します。"INValid" 状態ではこの組み合わせは使用できません。正常終了すると、オブジェクトの状態は変わりません。
"INValid"	オブジェクトは削除されます。

TA_DMPRIORITY_TYPE = {LOCAL_RELATIVE | LOCAL_ABSOLUTE | GLOBAL}

TA_DMINPRIORITY = number

このリモート・ドメインのメッセージの優先順位に関する処理を指定します。このパラメータは、BEA Tuxedo 8.0 以降のソフトウェアを実行しているドメインに適用されます。LOCAL_RELATIVE および LOCAL_ABSOLUTE パラメータは、すべてのドメイン・タイプに適用できます。ただし、GLOBAL パラメータは TDOMAIN のドメイン・タイプにしか適用されません。

LOCAL_RELATIVE パラメータは、tpsprio 呼び出しなどによってリモート・ドメインからの要求の優先順位がローカル・ドメインに伝達されないことを意味します。TA_DMINPRIORITY を設定すると、リモート・ドメインから受信する要求の優先順位には TA_DMINPRIORITY の値を基準にして値が設定されます。TA_DMINPRIORITY の値は -99 ～ +99 です (-99 と +99 を含む)。TA_DMINPRIORITY が設定されていない場合は、0 がデフォルト値になります。TA_DMINPRIORITY の設定値により、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。最も高い優先順位は 100 です。リモート・ドメインに対する要求では、要求の優先順位も一緒にリモート・ドメインに送信されます。

LOCAL_ABSOLUTE パラメータは、リモート・ドメインからの要求の優先順位がローカル・ドメインに伝達されないことを意味します。ただし、受信した要求の優先順位は、TA_DMINPRIORITY の絶対値に設定されます。TA_DMINPRIORITY が設定されていない場合は、デフォルト値として 50 が設定されます。TA_DMINPRIORITY の値の範囲は 1 ～ 100 (1 と 100 を含む) です。100 は最も高い優先順位です。リモート・ドメインに対する要求では、要求の優先順位も一緒にリモート・ドメインに送信されます。

GLOBAL パラメータは、リモート・ドメインからの要求の優先順位がローカル・ドメインに伝達されることを意味します。特定のサービスに対するデフォルト値はローカルでは無視されます。TA_DMINPRIORITY を設定すると、リモート・ドメインから受信する要求の優先順位が TA_DMINPRIORITY の値に加算されます。受信した要求には、優先順位の合計の絶対値が設定されます。TA_DMINPRIORITY の値は -99 ～ +99 です (-99 と +99 を含む)。TA_DMINPRIORITY が設定されていない場合は、0 がデフォルト値になります。TA_DMINPRIORITY の設定値により、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。最も高い優先順位は 100 です。TA_DMINPRIORITY を 0 に設定した場合は、リモート・ドメインから優先順位が直接受け取られます。リモート・ドメインに対する要求では、要求の優先順位も一緒にリモート・ドメインに送信されます。

値を指定していない場合は、デフォルト値として LOCAL_RELATIVE が設定され、TA_DMINPRIORITY は 0 に設定されます。

TA_DMTYPE が SNAX および OSITPX のとき使用可能な属性

TA_DMCODEPAGE:*string*[1..20]
: このアクセス・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。

TA_DMTYPE が TDOMAIN および OSITPX のとき使用可能な属性

TA_DMACLPOLICY:{LOCAL | GLOBAL}

このリモート・ドメイン・アクセス・ポイントに対する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を実行する TDOMAIN タイプのドメイン、および BEA Tuxedo 8.0 以降のソフトウェアを実行する OSITPX タイプのドメインに適用されます。

LOCAL は、ローカル・ドメインがこのリモート・ドメインから受け取ったサービス要求のアイデンティティを、このリモート・ドメイン・アクセス・ポイントの TA_DMLOCALPRINCIPALNAME 属性で指定されるプリンシパル名に変更することを意味します。GLOBAL は、ローカル・ドメインがインバウンドのサービス要求から受け取ったクリデンシャルを使用することを意味します。リモート・ドメインからクリデンシャルを受け取っていない場合は、クリデンシャルがないままサービス要求がサービスに転送されますが、通常これは失敗します。このパラメータを指定しないと、デフォルトの LOCAL が使用されます。

この属性は、リモート・ドメインからのクリデンシャルをローカル・ドメインで受け付けるかどうかを制御します。この属性に関連する属性として TA_DMCREDENTIALPOLICY 属性があり、これはローカル・ドメインがリモート・ドメインにクリデンシャルを送信するかどうかを制御します。

TA_DMLOCALPRINCIPALNAME: string[0..511]

接続プリンシパル名の識別子。これは、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アイデンティティです。この属性は、BEA Tuxedo 7.1 以降のソフトウェアを実行する TDOMAIN タイプのドメイン、および BEA Tuxedo 8.0 以降のソフトウェアを実行する *OSITPX* タイプのドメインに適用されます。

TA_DMLOCALPRINCIPALNAME 属性は、このリモート・ドメインの TA_DMACLPOLICY 属性が LOCAL に設定 (またはデフォルトで設定) されている場合に有効です。TA_DMLOCALPRINCIPALNAME 属性には最大 511 文字を指定できます (最後のヌルを除く)。この属性を指定していない場合、ローカル・プリンシパル名には、このリモート・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として設定されます。

TA_DMTYPE=TDOMAIN のとき使用可能な属性

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・ポイントへの接続を確立するときに、このリモート・ドメイン・アクセス・ポイントのアイデンティティを確認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を実行する TDOMAIN タイプのドメインに適用されます。

TA_DMCONNPRINCIPALNAME 属性には最大 511 文字を指定できます (最後のヌルを除く)。この属性を指定していない場合、接続プリンシパル名には、このリモート・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として設定されます。

デフォルトの認証プラグインで、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、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: Unable to initialize administration key for domain *domain_name*.

TA_DMCREDENTIALPOLICY:{LOCAL|GLOBAL}

このリモート・ドメイン・アクセス・ポイントのクリデンシャル方針。この属性は、BEA Tuxedo 8.0 以降のソフトウェアを実行する TDOMAIN タイプのドメインに適用されます。

クリデンシャル方針が LOCAL の場合、ドメインは、呼び出しに対して要求を発信したユーザのクリデンシャルをリモート・ドメインにアタッチしません。この方針が GLOBAL の場合は、ドメインは、呼び出しに対して要求を発信したユーザのクリデンシャルをリモート・ドメインにアタッチします。このパラメータを指定しないと、デフォルトの LOCAL が使用されます。

このパラメータは、ユーザのクリデンシャルをリモート・ドメインに送信するかどうかを制御します。このパラメータに関連するパラメータとして TA_DMACLPOLICY があり、これは受信したクリデンシャルをドメインで受け付けるかどうかを制御します。

TA_DMMACHINETYPE:*string*[0..15]
この属性を使用して、ドメインをグループ化し、ドメイン間のメッセージの符号化と復号化を省略できるようにします。TA_DMMACHINETYPE が指定されていないと、デフォルトで符号化や復号化が実行されます。TA_DMMACHINETYPE フィールドに設定した値がドメイン・コンフィギュレーション・ファイルの DM_LOCAL および 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。TA_DMMACHINETYPE には、15 文字までの文字列値を設定できます。この値は比較のためだけに使用します。

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、次の属性を更新することはできません。TA_DMACCESSPOINTID、TA_DMTYPE、TA_DMMACHINETYPE、または TA_DMCODEPAGE。

T_DM_ACL、T_DM_IMPORT、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、T_DM_REMOTE クラス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

DM_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を作成するとき必要
(k) — オブジェクト検索用のキー・フィールド
(*) — クラスに対するすべての SET 操作で必要なキー・フィールド

属性の意味

TA_DMVERSION:*string*[1..30]
ドメイン・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

DM_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を作成するとき必要
(k) — オブジェクト検索用のキー・フィールド
(*) — クラスに対するすべての SET 操作で必要なキー・フィールド

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 — ドメイン・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 . . .]] . . .]"

このルーティング項目が適用されるデータ・バッファのタイプおよびサブ・タイプのリスト。タイプとサブタイプの組み合わせを 32 まで使用できます。タイプは FML、FML32、XML、VIEW、VIEW32、X_C_TYPE、または X_COMMON のいずれかでなければなりません。FML 、FML32、または XML タイプにはサブタイプを指定できません。また、VIEW、VIEW32、X_C_TYPE、および X_COMMON タイプにはサブタイプが必要です ("*" 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 (*) といった記号は使用できません。同じルーティング基準名に対して重複するタイプとサブタイプのペアは指定できませんが、タイプとサブタイプのペアが一意であれば、複数のルーティング・エントリが同じ基準名を持つことはできます。1 つのルーティングに対して複数のバッファ・タイプを指定した場合は、それぞれのバッファ・タイプのルーティング・フィールドのデータ・タイプを同じにする必要があります。

TA_DMFIELD:*string*[1..30]

ルーティングが行わ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。ルーティングが実行される際に、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。ルーティングが実行される際に、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。

バッファをその正しいリモート・ドメイン・アクセス・ポイントにルーティングするとき、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) あるいはルーティング要素の属性名を指定します。

XML バッファ・タイプに対する TA_DMFIELD 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の要素のタイプであると想定されます。インデックスはサポートされていません。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、メッセージ送信時に、データ依存型ルーティングのため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なければなりません。

属性は、定義された要素の XML ドキュメントまたはデータグラム属性であると想定されます。この情報は、メッセージ送信時に、データ依存型ルーティングのため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なければなりません。

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。

ルーティング・フィールドのタイプは、TA_DMFIELDTYPE 属性によって指定できます。

TA_DMFIELDTYPE:"{CHAR | SHORT | LONG | FLOAT | DOUBLE | STRING}"

TA_DMFIELD 属性で指定されるルーティング・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、および関連するリモート・アクセス・ポイント。string の形式は、カンマで区切って並べられた範囲 / グループ名の組合せのになります。範囲 / グループ名の組合せの形式は下記の通りです。

"*lower*[-*upper*]:*raccesspoint*"

lower と *upper* は、符号を持つ数値、またはシングル・クォーテーション・マークに挟まれた文字列です。*lower* は、*upper* 以下でなければなりません。文字文字列の値にシングル・クォーテーション・マークを使用するためには、シングル・クォーテーション・マークの前に 2 つのバック・スラッシュを入力します (例 : 'O\\'Brien')。MIN という値を使用すると、マシンの関連フィールドのデータ・タイプに対する最小値を示すことができます。また、MAX という値を使用すると、マシンの関連フィールドのデータ・タイプの最大値を示すことができます。たとえば、"MIN--5" は、-5 以下のあらゆる数字を指し、"6-MAX" は、6 以上のあらゆる数字を指します。

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。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 (その後の範囲は無視されます)。

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。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。

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 オブジェクトに対するコンフィギュレーション情報を検索します。GET 要求に対して返される TA_STATE の意味を次に示します。リスト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 オブジェクトに対するコンフィギュレーション情報を更新します。次に、SET 要求で TA_STATE に設定できる値の意味を説明します。記述されていない状態を設定することはできません。

"NEW"	新しいオブジェクトが作成されます。
unset	既存のオブジェクトを変更します。"INValid" 状態ではこの組み合わせは使用できません。正常終了すると、オブジェクトの状態は変わりません。
"INValid"	オブジェクトは削除されます。

制限事項　T_DM_IMPORT クラスのインスタンスによって T_DM_ROUTING クラスのインスタンスが参照される場合、T_DM_ROUTING クラスのインスタンスを削除することはできません。

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

DM_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..30]	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を作成するとき必要
(k) — オブジェクト検索用のキー・フィールド
(*) — クラスに対するすべての SET 操作で必要なキー・フィールド

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、ドメイン・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、ドメイン・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_DMRACCESSPOINT 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 オブジェクトに対するコンフィギュレーション情報を検索します。GET 要求に対して返される TA_STATE の意味を次に示します。リスト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 オブジェクトに対するコンフィギュレーション情報を更新します。次に、SET 要求で TA_STATE に設定できる値の意味を説明します。記述されていない状態を設定することはできません。

"NEW"	新しいオブジェクトが作成されます。状態の変更は "INValid" 状態でのみ可能で、結果は "VALid" 状態になります。
unset	既存のオブジェクトを変更します。"INValid" 状態ではこの組み合わせは使用できません。
"INValid"	オブジェクトは削除されます。状態の変更は "VALid" 状態でのみ可能で、結果は "INValid" 状態になります。

制限事項

BEA Tuxedo リリース 7.1 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、名前を指定されたローカル・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

DM_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

DM_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_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を作成するとき必要
(k) — オブジェクト検索用のキー・フィールド
(*) — クラスに対するすべての SET 操作で必要なキー・フィールド

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されるドメイン・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 で使用されるローカル・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 オブジェクトに対するコンフィギュレーション情報を検索します。GET 要求に対して返される TA_STATE の意味を次に示します。記述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 オブジェクトに対するコンフィギュレーション情報を更新します。次に、SET 要求で TA_STATE に設定できる値の意味を説明します。記述されていない状態を設定することはできません。

"NEW"	新しいオブジェクトが作成されます。状態の変更は "INValid" 状態でのみ可能で、結果は "VALid" 状態になります。

unset	既存のオブジェクトを変更します。"INValid" 状態ではこの組み合わせは使用できません。
"INValid"	オブジェクトは削除されます。状態の変更は "VALid" 状態でのみ可能で、結果は "INValid" 状態になります。

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 間の通信のためのネットワーク・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 間の通信で使用するネットワーク・アドレスを指定します。

制限事項

参照されるローカル・アクセス・ポイントに対するドメイン・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 で指定されるローカル・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・ポイントに対する 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

DM_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

DM_MIB(5): T_DM_SNALINK **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_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を作成するとき必要
(k) — オブジェクト検索用のキー・フィールド
(*) — クラスに対するすべての SET 操作で必要なキー・フィールド

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される、ドメイン・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される 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・ポイント名を識別します。

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されるローカル 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されるリモート SYSID。

TA_DMLUNAME:*string*[1..8]
リモート・ドメイン・アクセス・ポイントと関連する 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と関連する名前を指定します。

TA_STATE:

GET:"{VALid}"
GET 操作は、T_DM_SNALINK オブジェクトに対するコンフィギュレーション情報を検索します。GET 要求に対して返される TA_STATE の意味を次に示します。リスト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 オブジェクトに対するコンフィギュレーション情報を更新します。次に、SET 要求で TA_STATE に設定できる値の意味を説明します。記述されていない状態を設定することはできません。

"NEW"	新しいオブジェクトが作成されます。
unset	既存のオブジェクトを変更します。"INValid" 状態ではこの組み合わせは使用できません。
"INValid"	オブジェクトは削除されます。

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 に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 で COLDSTART が設定されます。"AUTO" に設定すると、SNACRM はドメイン・ゲートウェイとともに、LU に対して COLDSTART または WARMSTART を選択します。

TA_DMMAXSNASESS:0 <= *num* <= 32767
: リモート LU と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 に対してサポートできる最大 SYNC LEVEL。

制限事項

以下の条件下では、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。T_DM_SNASTACK クラスのインスタンスが参照する T_DM_SNACRM クラスのインスタンスがローカル・ドメイン・アクセス・ポイントを参照し、そのローカル・ドメイン・アクセス・ポイントのドメイン・ゲートウェイ管理 (GWADM) サーバがアクティブである。

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、次のことが必要です。

- TA_DMRACCESSPOINT 属性で指定される特定のドメイン・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 によって使用される SNA スタックを定義します。

属性表

DM_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

DM_MIB(5): T_DM_SNASTACK 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_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を作成するとき必要
(k) — オブジェクト検索用のキー・フィールド
(*) — クラスに対するすべての SET 操作で必要なキー・フィールド

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、ドメイン・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・スタック定義が使用される SNA CRM の T_DM_SNACRM エントリを識別します。

TA_DMSTACKTYPE:*string*[1..30]
使用されるプロトコル・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されるセッションで使用される LU 名を指定します。

TA_DMTPNAME:*string*[1..8]
SNA スタックと関連する TP 名を指定します。値 "*" を指定すると、どの TP 名でも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 オブジェクトに対するコンフィギュレーション情報を検索します。GET 要求に対して返される TA_STATE の意味を次に示します。リスト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 オブジェクトに対するコンフィギュレーション情報を更新します。次に、SET 要求で TA_STATE に設定できる値の意味を説明します。記述されていない状態を設定することはできません。

"NEW"	新しいオブジェクトが作成されます。状態の変更は "INValid" 状態でのみ可能で、結果は "VALid" 状態になります。
unset	既存のオブジェクトを変更します。"INValid" 状態ではこの組み合わせは使用できません。
"INValid"	オブジェクトは削除されます。状態の変更は "VALid" 状態でのみ可能で、結果は "INValid" 状態になります。

制限事項

このクラスのインスタンスが参照する T_DM_SNACRM オブジェクトがローカル・ドメイン・アクセス・ポイントを参照し、そのローカル・ドメイン・アクセス・ポイントのドメイン・ゲートウェイ管理 (GWADM) サーバがアクティブのとき、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 属性で指定される SNA CRM 名が、T_DM_SNACRM クラス内に存在しなければなりません。名前が存在しない場合、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

DM_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..78]	N/A
TA_STATE(r)	string	rw-r--r--	GET:"VAL" SET:"{NEW | INV}"	N/A N/A

DM_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
TA_DMNWDEVICE	string	rw-r--r--	*string*[1..78]	N/A
TA_DMCMPLIMIT	long	rw-rw-r--	0 <= *num* <= MAXLONG	MAXLONG
TA_DMFAILOVERSEQ	short	rw-rw-r--	0 <= *num* <= 32767	下記参照。
TA_DMMINENCRYPTBITS	string	rw-------	"{0 | 40 | 56 | 128}" 注 1	"0"
TA_DMMAXENCRYPTBITS	string	rw-------	"{0 | 40 | 56 | 128}" 注 1	"128"

(r) — 新しいオブジェクトを作成するとき必要
(k) — オブジェクト検索用のキー・フィールド
(*) — クラスに対するすべての SET 操作で必要なキー・フィールド

注 1 リンク・レベルの暗号化の値 40 ビットは、下位互換性のために用意されています。

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・ポイント名。

ドメイン・レベルのフェイルオーバーが使用されているとき、同じ TA_DMACCESSPOINT 属性値で複数の T_DM_TDOMAIN クラス・エントリを定義することができます。

TA_DMNWADDR:*string*[1..78]

アクセス・ポイントと関連するネットワーク・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、この属性は着信に対する接続指示の受け付けに使用するアドレスを提供します。リモート・ドメイン・アクセス・ポイントの場合、この属性はリモート・ドメイン・アクセス・ポイントに接続する際に使用する宛先アドレスを提供します。このフィールドの値は、すべての T_DM_TDOMAIN エントリで一意でなければなりません。

TA_STATE:

GET:"{VALid}"

GET 操作は、T_DM_TDOMAIN オブジェクトに対するコンフィギュレーション情報を検索します。GET 要求に対して返される TA_STATE の意味を次に示します。記述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 オブジェクトに対するコンフィギュレーション情報を更新します。次に、SET 要求で TA_STATE に設定できる値の意味を説明します。記述されていない状態を設定することはできません。

"NEW"	新しいオブジェクトが作成されます。状態の変更は "INValid" 状態でのみ可能で、結果は "VALid" 状態になります。
unset	既存のオブジェクトを変更します。"INValid" 状態ではこの組み合わせは使用できません。
"INValid"	オブジェクトは削除されます。状態の変更は "VALid" 状態でのみ可能で、結果は "INValid" 状態になります。

TA_DMNWDEVICE:*string*[1..78]

使用されるネットワーク・デバイスを指定します。ローカル・ドメイン・アクセス・ポイントの場合、この属性は、接続指示を受け付けるために使用され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、この属性は、リモート・ドメイン・アクセス・ポイントに接続する際に使用され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・ポイントだけに関連します。このアクセス・ポイントへのトラフィックを圧縮するしきい値メッセージ・サイズです。

TA_DMFAILOVERSEQ:0 <= *num* <= 32767

リモート・ドメイン・アクセス・ポイントだけに関連します。このリモート・ドメイン・アクセス・ポイントに対して、このアドレス指定セットのフェイルオーバー・シーケンスにおける位置を指定します。フェイルオーバー・シーケンス番号が指定されないと、このリモート・ドメイン・アクセス・ポイントに対する最初のエントリに、リモート・ドメイン・アクセス・ポイントとして認識されている最も大きなフェイルオーバー・シーケンス番号より 10 大きな値が割り当てられます。したがって、最初のエントリは 10、2 番目のエントリは 20 というように番号が割り当てられます。

ドメイン・ゲートウェイは、特定のリモート・ドメイン・アクセス・ポイントに対して、T_DM_TDOMAIN アドレス指定エントリをフェイルオーバー・シーケンス番号順に、最小番号から最大番号の方向に使用します。

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だけに適用され、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無視されます。

TA_DMMINENCRYPTBITS: "{0 | 40 | 56 | 128}"
: リモート・ドメイン・アクセス・ポイントだけに関連します。この属性は、このアクセス・ポイントへの接続を確立する場合に必要な最小レベルの暗号化を指定します。"0" は暗号化を行わないことを示します。"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。

  値 40 ビットは下位互換性のために用意されています。

注記 この属性が変更されても、既に確立されている接続に影響はありません。

TA_DMMAXENCRYPTBITS: "{0 | 40 | 56 | 128}"
: リモート・ドメイン・アクセス・ポイントだけに関連します。この属性は、このアクセス・ポイントへのネットワーク・リンクを確立する場合に利用できる最大レベルの暗号化を指定します。"0" は暗号化を行わないことを示します。"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。デフォルトの値は "128" です。

  値 40 ビットは下位互換性のために用意されています。

注記 この属性が変更されても、既に確立されている接続に影響はありません。

制限事項

以下のシナリオでは、このクラスのインスタンスを削除する、またはこのクラスのインスタンスの TA_DMNWDEVICE 属性を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・ポイントに関連し、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。
- 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・ポイントに関連し、任意の 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。

T_DM_TOPEND クラスの定義

概要

T_DM_TOPEND クラスは、BEA TOP END システムに固有のローカルまたはリモート・ドメイン・アクセス・ポイントのコンフィギュレーションを定義します。

属性表

DM_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

DM_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_DMFAILOVERSEQ	short	rw-rw-r--	0 <= *num* <= 32767	下記の説明参照
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を作成するとき必要
(k) — オブジェクト検索用のキー・フィールド
(*) — クラスに対するすべての SET 操作で必要なキー・フィールド

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・ポイント名を指定します。

TA_DMFAILOVERSEQ:0 <= *num* <= 32767

リモート・ドメイン・アクセス・ポイントだけに関連します。このリモート・ドメイン・アクセス・ポイントに対して、このアドレス指定セットのフェイルオーバー・シーケンスにおける位置を指定します。フェイルオーバー・シーケンス番号が指定されないと、このリモート・ドメイン・アクセス・ポイントに対する最初のエントリに、リモート・ドメイン・アクセス・ポイントとして認識されている最も大きなフェイルオーバー・シーケンス番号より 10 大きな値が割り当てられます。したがって、最初のエントリは 10、2 番目のエントリは 20 というように番号が割り当てられます。

ドメイン・ゲートウェイは、特定のリモート・ドメイン・アクセス・ポイントに対して、T_DM_TOPEND アドレス指定エントリをフェイルオーバー・シーケンス番号順に、最小番号から最大番号の方向に使用します。

注記 この属性は、BEA Tuxedo リリース 7.1 を実行するゲートウェイだけに適用され、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無視されます。

TA_DMNWADDR:*string*[1..78]

アクセス・ポイントと関連するネットワーク・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、この属性は着信に対する接続指示の受け付けに使用するアドレスを提供します。リモート・ドメイン・アクセス・ポイントの場合、この属性はリモート・ドメイン・アクセス・ポイントに接続する際に使用する宛先アドレスを提供します。このフィールドの値は、すべての T_DM_TOPEND エントリ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・ポイントはすべて、同じ BEA TOP END システム名を持っていなければなりません。

TA_STATE:

GET:"{VALid}"

GET 操作は、T_DM_TOPEND オブジェクトに対するコンフィギュレーション情報を検索します。GET 要求に対して返される TA_STATE の意味を次に示します。リスト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 オブジェクトに対するコンフィギュレーション情報を更新します。次に、SET 要求で TA_STATE に設定できる値の意味を説明します。記述されていない状態を設定することはできません。

"NEW"	新しいオブジェクトが作成されます。状態の変更は "INValid" 状態でのみ可能で、結果は "VALid" 状態になります。
unset	既存のオブジェクトを変更します。"INValid" 状態ではこの組み合わせは使用できません。正常終了すると、オブジェクトの状態は変わりません。
"INValid"	オブジェクトは削除されます。状態の変更は "VALid" 状態でのみ可能で、結果は "INValid" 状態になります。

"RECrypt"	新しい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 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]
ローカルまたはリモート・ドメイン・アクセス・ポイントと関連するネットワーク・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され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・エントリだけに関連します。

T_DM_TRANSACTION クラスの定義

概要

T_DM_TRANSACTION クラスは、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して、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見つけることができます。

GET 操作では、特定のトランザクションを選択するために、属性 TA_DMTPTRANID、TA_DMTXACCESSPOINT、および TA_DMTXNETTRANID を指定できます。

属性表

DM_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

DM_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
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
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を作成するとき必要
(k) — オブジェクト検索用のキー・フィールド
(*) — クラスに対するすべての SET 操作で必要なキー・フィールド

属性の意味

TA_DMLACCESSPOINT:*string*[1..30]
トランザクションと関連するローカル・ドメイン・アクセス・ポイントの名前。これは GET 操作で必須の属性です。SET 操作では TA_DMLACCESSPOINT を指定する必要があります。

TA_DMTPTRANID:*string*[1..78]
文字列表現にマップされた、tpsuspend(3c) から返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 オブジェクトに対する実行時情報を検索します。GET 要求に対して返される TA_STATE の意味を次に示します。記述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ます。
"ABortonlY"	トランザクションはロールバックされるように識別されています。
"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"	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"	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。ブランチ状態により、どのブランチがヒューリスティックに完了したかを知ることができます。
"REAdy"	トランザクションは 2 フェーズ・コミットの第 1 フェーズを完了しました。参加グループおよびリモート・ドメインのすべてがコミットの第 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断できませんでした。

SET:"{INValid}"

SET 操作は、選択した T_DM_TRANSACTION オブジェクトに対する実行時情報を更新します。次に、SET 要求で TA_STATE に設定できる値の意味を説明します。記述されていない状態を設定することはできません。

"INValid"	指定され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 属性値が指定されないと、指定されたローカル・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で開始された場合、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、リモート・ドメイン・アクセス・ポイントから受信される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 には TA_DMTPTRANID 属性と同じ値が入ります。

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だけで利用可能であり、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ヌル文字列 "" 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。ブランチ情報を利用できないドメイン・ゲートウェイの場合、この値はゼロです。

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]
: このブランチに対してリモート・ドメイン・アクセス・ポイントで使用され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 はリモート・ドメイン・アクセス・ポイントへのブランチに対して、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・タイプで利用可能な場合)。

"ABorteD"	トランザクション・ブランチはロールバックされます。
"ABortonlY"	トランザクション・ブランチはロールバックとして識別されています。
"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"	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"	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・ブランチの通信またはロールバックは完了しました。リモート・ドメインが、コミットまたはロールバックに使用され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ないと報告しています。
"REAdy"	トランザクションは 2 フェーズ・コミットの第 1 フェーズを完了しました。参加グループおよびリモート・ドメインのすべてがコミットの第 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断できませんでした。

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だけで利用可能であり、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作成されます。管理者はこのオブジェクトに対して、状態を "INValid" にしか設定できません。状態を "INValid" に設定すると、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。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が "INValid" に設定された場合、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 操作の実行時には、特定のローカル・ドメイン・アクセス・ポイントを TA_DMLACCESSPOINT 属性に対して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。そうでないと、"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、システムのオペレータが把握している必要のあるあらかじめ定義された特定のイベント (おもに異常終了) を検出し、通知します。それぞれのイベント・レポートはいずれも FML32 バッファで、このバッファにはイベントに付いて記述した共通のフィールドとそのイベントに関連のあるオブジェクトについて記述したその他のフィールドが含まれます。

BEA Tuxedo システムは、システム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ことを見つけると、システム WARN または ERROR イベントを通知します。これらのイベントは、条件が解消されるまで通知されます。

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されるすべてのシステム・イベントをリストします。システム・イベント名はドット (.) 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する必要があります。

条件がごく短時間存在する場合、システムの制限の限界に関するイベントは通知されないことがあります (たとえば、.SysMachineFullMaxgtt)。

共通のイベント通知フィールド

TA_OPERATION:*string*
: EVT というリテラル文字列で、このバッファがイベント通知バッファであることを示します。

TA_EVENT_NAME:*string*
: このイベントを特定する文字列。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*
: ERROR、WARN、または INFO という文字列で、イベントの重要度を示します。

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よるイベント検出時間 (秒) を含む long の整数。

TA_EVENT_USEC:*long*
: イベントを検出したマシンのクロックによるイベント検出時間 (マイクロ秒) を含む long の整数。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*
: イベントについての要約を述べた 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。イベント通知バッファには、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が含まれます。

TA_ULOGCAT:*string*
: メッセージ・カタログ中のメッセージを使用した場合には、そのカタログの名前を示します。

TA_ULOGMSGNUM:*num*
: メッセージ・カタログ中のメッセージを使用した場合には、カタログのメッセージ番号を示します。

イベント・リスト

T_ACLPERM イベント・リスト

.SysAclPerm
: 情報 :.SysACLPerm: システム ACL パーミッション変更

T_DOMAIN イベント・リスト

.SysResourceConfig
: 情報 :.SysResourceConfig: システム構成変更

.SysLicenseInfo
: 情報 :.SysLicenseInfo:Tuxedo システムのバイナリ・ライセンス契約のユーザ数の 100% に到達、DBBL/BBL ロックアウトはキャンセルされます。
.SysLicenseInfo:Tuxedo システムのバイナリ・ライセンス契約のユーザ数の 90% に到達。
.SysLicenseInfo:Tuxedo システムのバイナリ・ライセンス契約のユーザ数の 90% に到達、DBBL/BBL ロックアウトはキャンセルされます。
.SysLicenseInfo:Tuxedo システムのバイナリ・ライセンス契約のユーザ数の 90% 以下に到達、DBBL/BBL ロックアウトはキャンセルされます。

SysLicenseWarn
: 警告 :.SysLicenseWarn:Tuxedo システムのバイナリ・ライセンス契約のユーザ数の 100% に到達。

SysLicenseError

エラー:.SysLicenseError:Tuxedo システムのバイナリ・ライセンス契約のユーザ数の 110% を超過、DBBL/BBL ロックアウトを実行、新しいクライアントがアプリケーションに結合することは不可能です。

.SysLicenseError:Tuxedo システムのバイナリ・ライセンス契約のユーザ数の 110% を超過、DBBL/BBL ロックアウトの実行前に %hour、%minutes、%seconds 経過。

T_GROUP イベント・リスト

.SysGroupState

情報:.SysGroupState: システム構成変更

T_MACHINE イベント・リスト

.SysMachineBroadcast

警告:.SysMachineBroadcast:%TA_LMID ブロードキャスト配送エラー

.SysMachineConfig

情報:.SysMachineConfig:%TA_LMID 構成変更

.SysMachineFullMaxaccessers

警告:.SysMachineFullMaxaccessers:%TA_LMID 制限の限界です。

.SysMachineFullMaxconv

警告:.SysMachineFullMaxconv:%TA_LMID 制限の限界です。

.SysMachineFullMaxgtt

警告:.SysMachineFullMaxgtt:%TA_LMID 制限の限界です。

.SysMachineFullMaxwsclients

警告:.SysMachineFullMaxwsclients:%TA_LMID 制限の限界です。

.SysMachineMsgq

警告:.SysMachineMsgq:%TA_LMID メッセージ・キューのブロッキング

.SysMachinePartitioned

エラー:.SysMachinePartitioned:%TA_LMID が分割されました。

.SysMachineSlow

警告:.SysMachineSlow:%TA_LMID DBBL に対する遅い応答

.SysMachineState

情報:.SysMachineState:%TA_LMID 状態が %TA_SATE に変わります。

.SysMachineUnpartitioned

エラー:.SysMachinePartitioned:%TA_LMID は分割されません。

T_BRIDGE イベント・リスト

.SysNetworkConfig
情報 :.SysNetworkConfig:%TA_LMID[0]->%TA_LMID[1] 構成変更

.SysNetworkDropped
エラー :.SysNetworkDropped:%TA_LMID[0]->%TA_LMID[1] 接続が切れました。

.SysNetworkFailure
エラー :.SysNetworkFailure:%TA_LMID[0]->%TA_LMID[1] 接続できません。

.SysNetworkFlow
警告 :.SysNetworkFlow:%TA_LMID[0]->%TA_LMID[1] フロー制御

.SysNetworkState
情報 :.SysNetworkState:%TA_LMID[0]->%TA_LMID[1] 状態が %TA_STATE に変わります。

T_SERVER イベント・リスト

.SysServerCleaning
エラー :.SysServerCleaning:%TA_SERVERNAME、グループ %TA_SRVGRP、id %TA_SRVID サーバをクリーニングします。

.SysServerConfig
情報 :.SysServerConfig:%TA_SERVERNAME、グループ %TA_SRVGRP、id %TA_SRVID 構成変更

.SysServerDied
エラー :.SysServerDied:%TA_SERVERNAME、グループ %TA_SRVGRP、id %TA_SRVID のサーバが停止しました。

.SysServerInit
エラー :.SysServerInit:%TA_SERVERNAME、グループ %TA_SRVGRP、id %TA_SRVID サーバの初期化が行えません。

.SysServerMaxgen
エラー :.SysServerMaxgen:%TA_SERVERNAME、グループ %TA_SRVGRP、id %TA_SRVID サーバが MAXGEN restart limit を超えました。

.SysServerRestarting
エラー :.SysServerRestarting:%TA_SERVERNAME、グループ %TA_SRVGRP、id %TA_SRVID サーバが再起動します。

.SysServerState
INFO:.SysServerState:%TA_SERVERNAME、グループ %TA_SRVGRP、id %TA_SRVID 状態が %TA_STATE に変わります。

.SysServerTpexit
エラー :.SysServerTpexit:%TA_SERVERNAME、グループ %TA_SRVGRP、id %TA_SRVID サーバが TPEXIT を要求しました。

T_SERVICE イベント・リスト

.SysServiceTimeout
エラー :.SysServiceTimeout:%TA_SERVERNAME、グループ %TA_SRVGRP、id %TA_SRVID サーバがサービス・タイムアウトのため強制終了しました。

T_CLIENT イベント・リスト

.SysClientConfig
情報 :.SysClientConfig:User %TA_USRNAME on %TA_LMID 構成変更

.SysClientDied
警告 :.SysClientDied:User %TA_USRNAME on %TA_LMID のクライアントが停止しました。

.SysClientSecurity
警告 :.SysClientSecurity:User %TA_USRNAME on %TA_LMID を認証できません。

.SysClientState
情報:.SysClientState:User %TA_USRNAME on %TA_LMID の状態が %TA_STATE に変わります。

T_TRANSACTION イベント・リスト

.SysTransactionHeuristicAbort
エラー:.SysTransactionHeuristicAbort:Transaction グループ %TA_GRPNO 内のトランザクション %TA_GTRID

.SysTransactionHeuristicCommit
エラー :.SysTransactionHeuristicCommit:Transaction グループ %TA_GRPNO 内のトランザクション %TA_GTRID

T_EVENT イベント・リスト

.SysEventDelivery
エラー :.SysEventDelivery:%TA_LMID 上のシステム・イベント・モニタ通知障害

.SysEventFailure
エラー :.SysEventFailure:%TA_LMID 上のシステム・イベント・モニタ・サブシステム障害

ファイル　${TUXDIR}/udataobj/evt_mib

関連項目　MIB(5)、TM_MIB(5)

EVENT_MIB(5)

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 のイベント・ブローカ MIB は、そのクラスを通じてイベント・ブローカを管理することのできるクラスの集合を定義したものです。

EVENT_MIB(5) は、管理の要求をフォーマットするためと管理の応答を解釈するために、共通 MIB のマニュアル・ページ MIB(5) と組み合わせて使用しなければなりません。MIB(5) およびコンポーネント MIB のマニュアル・ページに述べられているように、フォーマットされた要求は、アクティブなアプリケーション中の実在する数多くの ATMI インターフェイスのどれか 1 つを使用して、管理サービスを要求するために使用することができます。EVENT_MIB(5) クラスのすべてのクラス定義に関連する追加情報については、195 ページ「EVENT_MIB(5) に関する追加情報」を参照してください。

EVENT_MIB には次のクラスがあります。

EVENT_MIB **クラス**

クラス名	属性
T_EVENT_CLIENT	任意通知のサブスクリプション
T_EVENT_COMMAND	システム・コマンドの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	ユーザ・ログ・メッセージの書き込みのサブスクリプション

これらのクラスの中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ています。

各のクラスのパターン表現 TA_EVENT_EXPR は、それが SYSTEM EVENT 要求か USER EVENT 要求かを決定します。どちらの要求を照会するかは、次のように決定されます。

- TA_EVENT_EXPR または TA_EVENT_SERVER を指定しない基礎的な GET 要求は常に SYSTEM EVENT 要求を照会し、USER EVENT 要求は返しません。

- TA_EVENT_EXPR を指定し、TA_EVENT_SERVER を指定しない GET 要求は、表現が "¥." で始まる場合は SYSTEM EVENT 要求を照会します。それ以外の場合、USER EVENT 要求を照会します。
- TA_EVENT_SERVER に値 "SYSTEM" を指定した GET 要求は、SYSTEM EVENT 要求を照会します。値 "USER" を指定すると、USER EVENT 要求を照会します。

FML32 フィールド・テーブル

本マニュアル・ページに記述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の) 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で区切ったリストに、またフィールド・テーブル名 evt_mib は FIELDTBLS32 環境変数で指定されるカンマで区切ったリストに、アプリケーションで指定しなければなりません。

T_EVENT_CLIENT クラスの定義

概要

T_EVENT_CLIENT クラスは、クライアント・ベースの通知のために、イベント・ブローカに登録するサブスクリプションのセットを表しています。

あるイベントが検出された場合、それは T_EVENT_CLIENT オブジェクトと比較されます。もしそのイベント名が TA_EVENT_EXPR の中の値と一致していて、オプションのフィルタ規則が真であれば、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

EVENT_MIB(5): T_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) — オブジェクト検索用のキー・フィールド
(r) — 新しいオブジェクトを作成するとき必要なフィールド
(*) — GET/SET キー。SET 操作では 1 つ以上必要

パーミッションの説明については、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・パターンの表現です。この表現は、正規表現の形式で、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・フィルタの表現です。この表現が示されている場合は、ポストされたバッファの内容に関して評価されます。これは TRUE として評価されなければなりません。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・フィルタの表現です。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 つだけを指定することができます。

TA_STATE:

GET:ACTive
: GET 操作は、一致する T_EVENT_CLIENT オブジェクトについて、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 オブジェクトについてのコンフィギュレーション情報を更新します。次に、SET 要求で TA_STATE に設定できる値の意味を説明します。記述されていない状態を設定することはできません。

NEW	T_EVENT_CLIENT オブジェクトを作成します。正常終了すると、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たとき、このクライアントに任意通知型メッセージを送ります。

T_EVENT_COMMAND クラスの定義

概要

T_EVENT_COMMAND クラスは、システム・コマンドの実行を開始するイベント・ブローカが登録したサブスクリプションのセットを表しています。あるイベントが検出されたとき、それは T_EVENT_COMMAND オブジェクトの各々と比較されます。もしそのイベント名が TA_EVENT_EXPR の中の値と一致していて、オプションのフィルタ規則が真であれば、イベント・バッファがフォーマットされてシステムのコマンド・インタプリタに引き渡されます。

属性表

EVENT_MIB(5): T_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) — オブジェクト検索用のキー・フィールド
(r) — 新しいオブジェクトを作成するとき必要なフィールド
(*) — GET/SET キー。SET 操作のために 1 つ以上のキーが必要

パーミッションの説明については、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・パターンの表現です。この表現は、正規表現の形式で、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・フィルタの表現です。この表現が示されている場合は、ポストされたバッファの内容に関して評価されます。これは TRUE として評価されなければなりません。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・フィルタの表現です。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 つだけを指定することができます。

TA_STATE:

GET:ACTive

GET 操作は、一致する T_EVENT_COMMAND オブジェクトに対する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 オブジェクトに対するコンフィギュレーション情報を更新します。次に、SET 要求で TA_STATE に設定できる値の意味を説明します。記述されていない状態を設定することはできません。

NEW	T_EVENT_COMMAND オブジェクトを作成します。正常終了すると、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時、このシステムコマンドを実行します。UNIX システムプラットフォームでは、そのコマンドはバックグラウンドを用いたシステム (3) 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・ベースの通知のためにイベント・ブローカに登録するサブスクリプションのセットを表しています。あるイベントが検出されたとき、それは T_EVENT_QUEUE オブジェクトの各々と比較されます。もしイベント名が TA_EVENT_EXPR の中の値と一致していて、オプションのフィルタ規則が真であれば、イベント・バッファは指定された信頼性のあるキューに格納されます。

属性表

EVENT_MIB(5): T_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

EVENT_MIB(5): T_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_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

(k) — オブジェクト検索用のキー・フィールド
(r) — 新しいオブジェクトを作成するとき必要なフィールド
(*) — GET/SET キー。SET 操作のために 1 つ以上のキーが必要

パーミッションの説明については、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・パターンの表現です。この表現は、正規表現の形式で、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・フィルタの表現です。この表現が示されている場合は、ポストされたバッファの内容に関して評価されます。これは TRUE として評価されなければなりません。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・フィルタの表現です。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 つだけを指定することができます。

TA_STATE:

GET:ACTive
: GET 操作は、一致する T_EVENT_QUEUE オブジェクトに対する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 オブジェクトについてのコンフィギュレーション情報を更新します。次に、SET 要求で TA_STATE に設定できる値の意味を説明します。記述されていない状態を設定することはできません。

NEW	T_EVENT_QUEUE オブジェクトを作成します。正常終了すると、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たとき、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入れます。

TA_QNAME:*string*[1..15]

一致するイベントが検出されたとき、通知メッセージをこの信頼性のあるキューに入れます。

TA_QCTL_QTOP:*short*

この値が設定されている場合、これは、/Q サブシステム経由で通知を要求するため、キューの先頭に置かれるメッセージとともに、tpenqueue() の TPQCTL 制御構造体に渡されます。

TA_QCTL_BEFOREMSGID:*short*

この値が設定されている場合、これは、/Q サブシステム経由で通知を要求するため、キューの指定されたメッセージの前に置かれるメッセージとともに、tpenqueue() の TPQCTL 制御構造体に渡されます。

TA_QCTL_QTIME_ABS:*short*

この値が設定されている場合、これは、/Q サブシステム経由で通知を要求するため、指定された時間に処理されるメッセージとともに、tpenqueue() の TPQCTL 制御構造体に渡されます。

TA_QCTL_QTIME_REL:*short*

この値が設定されている場合、これは、/Q サブシステム経由で通知を要求するため、キューから取り出された時間に対する相対的な時間に処理されるメッセージとともに、tpenqueue() の TPQCTL 制御構造体に渡されます。

TA_QCTL_DEQ_TIME:*short*

この値が設定されている場合、これは tpenqueue() の TPQCTL 制御構造体に渡されます。

TA_QCTL_PRIORITY:*short*
この値が設定されている場合、これは tpenqueue() の TPQCTL 制御構造体に渡されます。

TA_QCTL_MSGID:*string*[1..31]
この値が設定されている場合、これは tpenqueue() の TPQCTL 構造体に渡されます。

TA_QCTL_CORRID:*string*[1..31]
この値が設定されている場合、これは tpenqueue() の TPQCTL 構造体に渡されます。

TA_QCTL_REPLYQUEUE:*string*[1..15]
この値が設定されている場合、これは tpenqueue() の TPQCTL 制御構造体に渡されます。

TA_QCTL_FAILUREQUEUE:*string*[1..15]
この値が設定されている場合、これは tpenqueue() の TPQCTL 制御構造体に渡されます。

TA_EVENT_PERSIST:*short*
ゼロでない場合、指定されたキューがそれ以上使えなくても、このサブスクリプションを取り消しません。

TA_EVENT_TRAN:*short*
ゼロでない場合、クライアントの tppost() 呼び出しがトランザクションに使われていれば、クライアントのトランザクションに tpenqueue() 呼び出しを組み入れます。

T_EVENT_SERVICE クラスの定義

概要

T_EVENT_SERVICE クラスは、サービス・ベースの通知のために、イベント・ブローカに登録するサブスクリプションのセットを示しています。あるイベントが検出されたとき、それは T_EVENT_SERVICE オブジェクトの各々と比較されます。もしイベント名が TA_EVENT_EXPR の中の値と一致していて、オプションのフィルタ規則が真であれば、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

EVENT_MIB(5): T_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

EVENT_MIB(5): T_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_STATE(r)	string	R-xR-xR-x	GET:ACT SET:{NEW | INV}	N/A 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) — オブジェクト検索用のキー・フィールド
(r) — 新しいオブジェクトを作成するとき必要なフィールド
(*) — GET/SET キー。SET 操作では 1 つ以上必要

パーミッションの説明については、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・パターンの表現です。この表現は、正規表現の形式で、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・フィルタの表現です。この表現が示されている場合は、ポストされたバッファの内容に関して評価されます。これは TRUE として評価されなければなりません。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・フィルタの表現です。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 つだけを指定することができます。

TA_STATE:

GET:ACTive
GET 操作は、T_EVENT_SERVICE オブジェクトについて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 オブジェクトについてのコンフィギュレーション情報を更新します。次に、SET 要求で TA_STATE に設定できる値の意味を説明します。記述されていない状態を設定することはできません。

NEW	T_EVENT_SERVICE オブジェクトを作成します。正常終了すると、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たとき、この BEA Tuxedo サービスが呼び出されます。

TA_EVENT_PERSIST:*short*
ゼロでない場合、TA_SERVICENAME サービスがそれ以上使えなくても、このサブスクリプションを取り消しません。

TA_EVENT_TRAN:*short*
ゼロでない場合、クライアントの tppost() 呼び出しがトランザクションに使われていれば、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組み入れます。

T_EVENT_USERLOG クラスの定義

概要

T_EVENT_USERLOG クラスは、システムの userlog(3c) メッセージの書き込みのために、イベント・ブローカに登録するサブスクリプションのセットを表します。あるイベントが検出されたとき、それは T_EVENT_USERLOG オブジェクトの各々と比較されます。イベント名が TA_EVENT_EXPR の値と一致していて、オプションのフィルタ規則が真であれば、イベント・バッファはフォーマットされて BEA Tuxedo の userlog(3c) 関数に送られます。

属性表

EVENT_MIB(5): T_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 SET:{NEW | INV}	N/A N/A
TA_USERLOG(r)	string	R--R-----	*string*[1..255]	N/A

EVENT_MIB(5): T_EVENT_USERLOG **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
(k) — オブジェクト検索用のキー・フィールド (r) — 新しいオブジェクトを作成するとき必要なフィールド				

パーミッションの説明については、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・パターンの表現です。この表現は、正規表現の形式で、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・フィルタの表現です。この表現が示されている場合は、ポストされたバッファの内容に関して評価されます。これは TRUE として評価されなければなりません。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・フィルタの表現です。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 つだけを指定することができます。

TA_STATE:

GET:ACTive
: GET 操作は、一致する T_EVENT_USERLOG オブジェクトについて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 オブジェクトについてのコンフィギュレーション情報を更新します。次に、SET 要求で TA_STATE に設定できる値の意味を説明します。記述されていない状態を設定することはできません。

NEW	T_EVENT_USERLOG オブジェクトを作成します。正常終了すると、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、userlog(3c) メッセージを書き込みます。

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

名前　factory_finder.ini—FactoryFinder ドメイン・コンフィギュレーション・ファイル

機能説明　factory_finder.ini は、ドメインの FactoryFinder コンフィギュレーション・ファイルです。TMFFNAME サービスは、Master NameManager として起動されると、このテキスト (ASCII) ファイルを解析します。NameManager はこのファイルに包含されている情報を使って、ほかのドメインからのファクトリ・オブジェクトのオブジェクト参照のインポートや、ほかのドメインへの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 オプションに factory_finder.ini ファイルを指定する必要があります。

定義　BEA Tuxedo システムにおけるドメイン・アプリケーションは、環境の定義として単一の TUXCONFIG ファイル内で記述されます。BEA Tuxedo システムのアプリケーションは、ドメイン・ゲートウェイ・グループを介して、別の BEA Tuxedo システムのアプリケーションや別の TP アプリケーションと通信できます。「BEA Tuxedo システム・ドメイン」の用語では、アプリケーションと TP ドメインは同義です。

リモート・ファクトリは、リモート・ドメインに存在するファクトリ・オブジェクトで、アプリケーションは 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・ドメインに存在するファクトリ・オブジェクトで、リモート・ドメインは 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、DM_REMOTE_FACTORIES と DM_LOCAL_FACTORIES の 2 つの仕様セクションによって構成されています。

- フォーマット処理のガイドライン

  パラメータは通常、KEYWORD = value です。これで KEYWORD が value に設定されます。有効なキーワードについては、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、値として使用できません。

  値が identifier の場合は、標準的な C の規則が適用されます。identifier は、英字または下線で始まり、英数字または下線のみを使用していなければなりません。KEYWORD と同じ identifier を使用することはできません。

  identifier でない値は、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。

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あり、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することができます。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・ファクトリを制限することができます。

  このセクションの行の形式は、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 (identifier) 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。

  適切なファクトリを見つけるには、TMFFNAME に factory_kind を指定する必要があります。factory_kind 値を包含していないエントリは、"FactoryInterface" の値にデフォルト設定されません。

- DM_REMOTE_FACTORIES セクション

  このセクションは、リモート・ドメイン上に「インポート」されて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。このセクションの行の形式は、次のとおりです。

  ```
  factory_id.factory_kind required parameters
  ```

  factory_id.factory_kind は、ローカル BEA Tuxedo システム・ドメイン・アプリケーションが特定のリモート・ファクトリ・オブジェクトに対して使用する名前 (identifier) 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・ドメインと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。DOMAINID 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 進数で指定できます。

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 パラメータのいずれ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 ドメイン・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 ドメイン
# 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であると見なされます。

- **例 2**

  次に示す FactoryFinder ドメイン・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・オブジェクトだけが、リモート・ドメインへのエクスポートを許可されます。ほかのファクトリに対する要求は拒否されます。

```
# BEA Tuxedo FactoryFinder ドメイン
# 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 ドメイン・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・オブジェクトはリモート・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 ドメイン
# コンフィギュレーション・ファイル
#
*DM_LOCAL_FACTORIES
   NONE
```

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

エラー状態のシンボリック名が示す数値を、多数の FML ライブラリのルーチンを実行する時に生じるエラーである Ferror に割り当てます。

Ferror という名前は、タイプ int を持つ修正可能な *lvalue* に拡張されます。この値は、FML ライブラリ・ルーチンが正のエラー番号に設定します。Ferror は、オブジェクトの識別子である必要はありません。Ferror は、関数の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、外部リンケージで宣言される識別子であるかは、指定されていません。tperrno() のマクロ定義が実際のオブジェクトにアクセスできない場合、またはプログラムが Ferror という名前で識別子を定義している場合は、動作は未定義です。

マニュアルの FML ルーチンの項目に、各ルーティンに対して起こりうるエラー状況とその場合のエラーの意味がリストされています。リスト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はありません。Ferror の値は、エラーが表示された場合にのみ、その後で確認します。つまり、構成要素の戻り値がエラーを表示し、構成要素の定義が tperrno() が設定されたことを示したときに確認します。Ferror の値をチェックするアプリケーションには、fml.h のヘッダ・ファイルが含まれることが必要です。

Ferror32 は、FML32 ルーチンのユーザに対して同様な機能を提供します。Ferror32 の値をチェックするアプリケーションには、fml32.h のヘッダ・ファイルが含まれることが必要です。

下記のリストは、FML および FML32 のルーチンが返すエラー・コードです。

```
#define FMINVAL 0   /* bottom of error message codes */
#define FALIGNERR 1  /* fielded buffer not aligned */
#define FNOTFLD 2   /* buffer not fielded */
#define FNOSPACE 3   /* no space in fielded buffer */
#define FNOTPRES 4   /* field not present */
#define FBADFLD 5   /* unknown field number or type */
#define FTYPERR 6   /* illegal field type */
#define FEUNIX 7    /* unix system call error */
#define FBADNAME 8   /* unknown field name */
#define FMALLOC 9   /* malloc failed */
#define FSYNTAX 10   /* bad syntax in boolean expression */
#define FFTOPEN 11   /* cannot find or open field table */
#define FFTSYNTAX 12  /* syntax error in field table */
#define FEINVAL 13   /* invalid argument to function */
```

```
#define FBADTBL 14   /* destructive concurrent access to field table */
#define FBADVIEW 15  /* cannot find or get view */
#define FVFSYNTAX 16  /* bad viewfile */
#define FVFOPEN 17   /* cannot find or open viewfile */
#define FBADACM 18   /* ACM contains negative value */
#define FNOCNAME 19   /* cname not found */
```

使用方法

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がないので、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ます。エラーが発生したかどうかを確認するために再度 Ferror をチェックすることができます。

DOS および OS/2 の環境では、この変数は FMLerror として知られています。

関連項目

各ルーチンから返されるエラー・コードの意味に関する詳細については、個々の FML ライブラリ・ルーチンのエラー・セクション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。このように、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 は、より大きい識別子、フィールドの長さ、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、関数名などに、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 です) を付けます。

フィールド識別子

FML の関数では、フィールド値のタイプを決めることができます。現在、次のタイプがサポートされています。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)。ptr、FML32、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。各種のフィールド・タイプの定数は fml.h (FML32 では fml32.h) に定義されています。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 でエンコードすることによりフィールドと共に渡されます。このため、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 までの番号はシステムが使用するために予約されているからです。

フィールド・マッピング

効率を考えると、コンパイル時にフィールド名からフィールド識別子への変換が行われることが望まれます。また、便利さを考えれば、この変換が実行時に行われるとよいでしょう。この双方の目的を満たすために、FML は、テキスト・ファイルにフィールド・テーブルを保持させ、また、対応する C ヘッダ・ファイルを生成するコマンドを提供しています。このため、コンパイル時のマッピングは、C プリプロセッサ cpp により通常の #define マクロを介して行われ、実行時マッピングは関数 Fldid() (FML32 では Fldid32()) により行われます。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通りです。

- 先頭に # が付いている行と空白行は無視されます。
- 先頭に $ が付いている行はマッピング関数には無視されますが、mkfldhdr() (FML32 ではコマンド名は mkfldhdr32()。mkfldhdr、mkfldhdr32(1) を参照) により生成されたヘッダ・ファイルには渡されます (このとき $ は除去されます)。たとえば、これにより、アプリケーションは C 言語の注釈、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分けし、番号を付け直すことが簡単にできるようになります。
- 先頭に * も # も付いていない行の形式は次のようになります。

  ```
  name    rel-numb   type
  ```

  ここで、

  - name は、フィールドの識別子です。cpp の制限を越えてはいけません。
  - rel-numb は、フィールドの相対数値です。この数値を現在のベース値に加えることで、フィールド番号を得ることができます。
  - 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

コマンド mkfldhdr (または mkfldhdr32) は、上述したように、フィールド・テーブルを C コンパイラで処理できるようなファイルに変換します。生成されたヘッダ・ファイルの各行の形式は次のようになります。

```
#define  name  fldid
```

ここで、name は、フィールドの名前で、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、ベース値に相対番号を加えた絶対番号です。このようにして、C プログラムに組み込むことができるファイルが生成されます。

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 に値がない場合は、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あり、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 が定義されていない場合は、カレント・ディレクトリであるとみなされます。したがって、FIELDTBLS と FLDTBLDIR が設定されていない場合は、省略時設定としてカレント・ディレクトリから fld.tbl がとられます。

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・テーブルは C ヘッダ・ファイルに変換される可能性があり (mkfldhdr コマンドによって)、同一のフィールド名があるとコンパイラ名の矛盾を示す警告が出されるため、一般にフィールド名は、フィールド・テーブル全体に渡って一意であるようにします。また、関数 Fldid は、名前を FLDID にマップしますが、その際、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

ベース値が 500 から 700 に移る場合の、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) /* number:701  type:string */
#define EMPCITY  ((FLDID)41662) /* number:702  type:string */
#define EMPID   ((FLDID)8694)  /* number:502  type:long */
#define EMPJOB  ((FLDID)16887) /* number:503  type:char */
#define EMPNAM  ((FLDID)41461) /* number:501  type:string */
#define EMPSTATE ((FLDID)41663) /* number:703  type:string */
#define EMPZIP  ((FLDID)8896)  /* number:704  type:long */
#define SRVCDAY  ((FLDID)49656) /* number: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、ゲートウェイ・グループの管理機能を持つ BEA Tuxedo システム提供のサーバです。

GWADM を、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。SVRID の値を指定するときは、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。

ゲートウェイ・グループごとに GWADM のインスタンスは 1 つだけ存在でき、そのインスタンスは、グループと関連付けられるゲートウェイに対して定義した MSSQ の一部であってはいけません。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 が起動されるとき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は、次のフォーマットをとります。

```
CLOPT="-A -- ゲートウェイ・グループ実行時パラメータ >"
```

次のパラメータが、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・ドメインに対する動作記録ログ機能をオフたはオンに切り替えます。デフォルトはオフです。dmadmin プログラムを使用して、ゲートウェイ・グループの稼働中にこの設定を変更できます (dmadmin(1) を参照)。

-t {*on* | *off*}
: このオプションは、このローカル・ドメインに対する統計集積機能をオフまたはオンに切り替えます。デフォルトはオフです。dmadmin プログラムを使用して、ゲートウェイ・グループの稼働中にこの設定を変更できます (dmadmin(1) を参照)。

GWADM サーバをブートしてから、それに対応するゲートウェイをブート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・プラットフォーム上で、BEA Tuxedo システム提供のサーバとしてサポートされています。

相互運用性

GWADM は、リリース 4.2.1 以降の BEA Tuxedo 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 ファイル内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 ドメインとの接続を提供します。BEA TOP END システムとの相互運用性を提供するには、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・プロセスに関する詳細と 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)、GWTOPEND(5) の DMCONFIG、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 = N 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 ファイル内の 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 パラメータの指定も必須であり、このパラメータ値を設定する場合は、ドメイン・グループに対して指定可能な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、REPLYQ パラメータは N に設定します。

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す。ただし、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 BEA TOP END システムの 1 つまたは複数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 グループの) GWTOPEND ゲートウェイを、異なる BEA TOP END システムにアクセスするよう設定したり、負荷を分割するように設定することもできます。GWTOPEND は要求 / 応答、擬似会話、キューイング、およびトランザクションをサポートします。

ドメイン・ゲートウェイは、UBBCONFIG ファイルおよび BDMCONFIG ファイル内の 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 パラメータの指定も必須であり、このパラメータ値を設定する場合は、ドメイン・グループに対して指定可能な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、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す。ただし、その場合は、各プロセスを別個の 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、「ファイル」セクションで示すとおり、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する必要があります。また TP_SYSTEM 名の srvtab ファイルをノードにコピーする必要もあります。long 型のノード名がサポートされる場合は、「ファイル」セクションで示すとおり、nodemap ファイルをノードにコピーする必要があります。

ファイル

$TUXDIR/udataobj/nodemap

$APPDIR/srvtab.*system* (ここで、*system* は BEA TOP END システム名)

/usr/lib/libtp_krb.so (BEA TOP END セキュリティが設定される 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 セキュリティが設定される Windows NT 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
```

UBBCONFIG(5) および GWTOPEND(5) の DMCONFIG の「使用例」も参照してください。

関連項目

tmadmin(1)、tmboot(1)、DMADM(5)、GWTOPEND(5) の DMCONFIG、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 と 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 サーバ間の接続を実現します。GWTE2TUX は、BEA TOP END クライアントと BEA Tuxedo サーバ間の接続を実現します。これらのゲートウェイ・サーバの一方、または両方をドメインに対してコンフィギュレーションすることができます。

GWTUX2TE と 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 セクションで定義されます。そのため、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 パラメータの指定も必須です。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、スループットを向上させるために、複数のインスタンスを使用することもできます。

CLOPT パラメータは、サーバのブート時にゲートウェイ・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、次の形式を用います。

```
CLOPT="-- gateway_group_runtime_parameters"
```

以下の CLOPT オプションが認識されます。

-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 システムの接続が定義されます。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・ファイルとして使用されます。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に指定する必要がありますセキュリティを確実にするため、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、「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 に 0 を設定すると、再接続は行われません。R に 0 を設定し、RESTART=Y を指定した状態で接続が失敗したり中断されると、ゲートウェイ・サーバは終了し、再起動します。

BEA TOP END システムに接続できない場合、そのシステムのサービスは、ゲートウェイ・サーバによって提供されません。

-w *wait_time*

ゲートウェイ・サーバ GWTUX2TE から BEA TOP END システムへリクエストが送信されると、ゲートウェイ・サーバは、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ありません。そのため、TUXCONFIG にある、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。デフォルトにより、ゲートウェイではゲスト特権が使用されます。

詳細については、「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。サポートされている BEA Tuxedo クライアント API 呼び出し (リクエストの送受信を行う) は、以下のとおりです。

- tpcall()
- tpacall() (TPNOREPLY フラグが設定される場合もあり)
- tpgetrply()
- tpforward()

BEA TOP END サーバでは APPL_CONTEXT フラグを設定できません。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・メッセージ (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・メッセージ (TPESVCFAIL) を返します。) を返します。

サポートされている BEA TOP END クライアント API 呼び出しは、以下のとおりです。

- tp_client_send
- tp_client_receive

バッファ・タイプ

ゲートウェイ・サーバ GWTUX2TE および GWTE2TUX でサポートされるバッファ・タイプは、BEA Tuxedo の CARRAY (X_OCTET) バッファのみです。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、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動作するため、リモート・クライアントのサービスを使用します。GWTE2TUX は BEATOP END サーバとして動作するため、リモート・サーバのサービスを使用します。そのため、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・ファイルは、『BEATOP END Remote Client Services Guide』で説明されています。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、以下のとおりです。

```
[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 TOP ENDNetwork 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。セカンダリ・ノードを複数指定する場合、BEA TOP END システムでは、ラウンド・ロビン方式で接続のロード・バランシングを行います。この方式により、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。以下は、その例です。

```
[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・パラメータである target パラメータもサポートされています。

ゲートウェイ・サーバ GWTUX2TE および GWTE2TUX では、以下のパラメータはサポートされません。コンフィギュレーション・ファイルにこれらのパラメータを含めないでください。

- 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] で指定された 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 システムへ接続するよう設定することもできます。2 つ目のコンフィギュレーション・ファイルを指すには、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、以下のとおりです。

```
*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あり、FUNCTION パラメータ、QUALIFIER パラメータ、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。

次の構文を使用して、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 パラメータが指定されない場合、関数名はサービス名と見なされます。サービス・エントリに対して QUALIFIER パラメータおよび TARGET パラメータが指定されない場合、そのサービスに対して関数修飾子または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 END Administrator'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
- 任意通知型通知
- 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
- MCC および LMA

セキュリティ

次の表 1 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 1 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。	必要なし
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。

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、ユーザ名は一般に tpusradd() を使用して作成されます。グループ名は、一般に tpgrpadd() を使用して作成されます。

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 の動作には、BEA Tuxedo リリース 6.5 以降が必要です。また、これらのゲートウェイ・サーバは、BEA TOP END 2.05 以上と相互運用します。

使用例

以下の例は、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ています。

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。リクエストは 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。リクエストは、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。

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"
```

BEA TOP END 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
```

BEA TOP END 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 (リスト の「BEA TOP END 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 Agent のポート番号と一致する必要があります。

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 クライアントは TPESVCFAIL を受け取ります。

- 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る。
- 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 または 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、BEA TOP END クライアントはエラー・メッセージ TP_RESET (詳細なステータスである TP_EXT_SERVER_APPL 付き) を受け取ります。

- 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ため、などの理由による)。
- BEA Tuxedo サービスがタイムアウトの状態である。
- BEA Tuxedo サービスから TPFAIL または TPEXIT が返される。

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、クライアントはエラー・メッセージ (TPESVCFAIL) を受け取ります。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・メッセージとは異なります。ローカル・サービスを呼び出した場合、クライアントは 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) を受け取ります。

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 データの項目を識別するため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) にあります。

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
　　年の最初の月 (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
　　年の最初の月の略称 (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 ","

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) により取り出されたものです。

項目は、LANGINFO という特別なメッセージ・カタログから取り出されています。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) 参照)。

関連項目 mklanginfo(1)、nl_langinfo(3c)、strftime(3c)、nl_types(5)

MIB(5)

名前

MIB— 管理情報ベース

```
#include <fml32.h>
#include <fml1632.h> /* Optional */
#include <tpadm.h>
#include cmib.h>  /* Component MIB Header */
```

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のために特別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 関連のマニュアル・ページで定義されています。たとえば、TM_MIB(5) のマニュアルでは、BEA Tuxedo アプリケーションの基本的な側面を管理するために使用される 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、必要なアクセス手段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あるわけではありません。MIB(5) のマニュアル・ページでは、管理者、オペレータ、あるいはユーザが定義されているコンポーネント MIB と対話するための共通のインターフェイスについて説明したものです。BEA Tuxedo システムの MIB に対する共通のインターフェイスは、2 つの主要な部分から成り立っています。

共通のインターフェイスの最初の部分は、コンポーネント MIB をサポートするための管理サービスへのアクセス手段を提供する際に、BEA Tuxedo システムで現在使用しているインターフェイスがどのように使用されるかを説明したものです。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の「入れ物」として使用されます。ATMI 要求 / 応答関数は、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され、システムが提供するサービスとして組み込まれています。管理ユーザとコンポーネント MIB と間で FML32 バッファの ATMI 関数を利用して行われるやりとりについては、このマニュアルの「FML32」および「ATMI」の各セクションで詳しく説明します。

共通なインターフェイスのもう一つの部分は、あらゆるコンポーネント MIB との間のやりとりで使用される FML32 の追加の入出力フィールドについて説明したものです。FML32 の追加のフィールドを利用した場合は、要求の機能を拡張でき (操作コードの指定などが可能になる)、新たな応答属性 (エラー・コードや説明文など) の使用が可能になるといった効果が得られます。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」のセクションで詳しく説明します。

「使用方法」のセクションでは、管理を目的としたコンポーネント MIB との対話に利用できる既存の ATMI 関数や追加の FML32 フィールドの使用例を示します。

また、このマニュアル・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 との対話方法を定義するだけでなく、コンポーネント MIB に関するマニュアルで使用されるクラス定義の形式についても明らかにします (「クラス記述」を参照のこと)。

このリファレンス・ページでは、T_CLASS と T_CLASSATT という 2 つの共通のクラスを定義しています。これら 2 つのクラスは、管理のためのクラスを識別し、クラスや属性のパーミッションを調節するために使用されます。MIB(5) のすべてのクラス定義に関連する追加情報については、248 ページ「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)。tpinit() の実行時に、管理者およびオペレータは tpsysadm または tpsysop のクライアント名を持つアプリケーションに結合するよう求めることができます。この 2 つの cltname 値は保存され、アプリケーションの管理者とオペレータにしか関連付けられません。

アプリケーションを最初に環境設定する管理者が、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。以下のセキュリティ・タイプから選択できます。

- 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＋アプリケーション固有の認証サービス

セキュリティ・タイプを選択すると、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 + アプリケーション固有認証サーバ (AUTHSVR(5) 参照) です。この方法では、ユーザが適切なパスワードを認証サーバに提供すれば、管理者は任意のユーザまたは指定されたユーザにアクセスを許可することができます。

アプリケーション固有の認証サーバがない場合、管理者またはオペレータの特別なパーミッションを得るためには、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、さらにローカル UNIX システムの BEA Tuxedo 管理者で実行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、クライアントが行な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は、このクライアントが特別な特権を持っていることをシステムに知らせる認証キーが割り当てられます。

管理者としての認証は、指定に従って、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しか適用されません。API を利用するサーバは、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()、tpsvrdone() から行わ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 バッファタイプを介してしかサポートされません。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、更新を行なうように書かれていなければなりません。FML32 を使用する 2 通りの方法については、Fintro() で詳述していますが、ここで要約します。

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、次に『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・ファイルは共同して機能し、ユーザはベースの FML インターフェイス (たとえば Fchg()) に合わせてプログラミングできますが、実際には各インターフェイスの FML32 バージョンが呼び出されるようにします。

ATMI

アプリケーション・プログラムは、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入れ、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り、その応答から結果に関する情報を取り出すことによって、コンポーネント MIB 固有の属性情報のアクセスや更新を行います。FML32 型付きバッファへ情報を入れたり取り出したりする操作には、前述した FML32 インターフェイスが関わります。バッファの割り当て、要求の送出と応答の受信は、下記の汎用 ATMI ルーチンを使い該当する指針と制約の範囲内で行なわれます。すべてのコンポーネントに対する MIB 要求は、コアのシステム /T コンポーネント MIB サービス、".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ほか、他のコンポーネント MIB に対する要求を転送します。これで、ユーザ側ではサービス名を MIB やクラスとマッチングしなくても済みます。

tpalloc()
: BEA Tuxedo システム MIB サービスへ要求を送出したり応答を受信したりするときに使用する FML32 型付きバッファを割り当てます。FML32 バッファタイプはサブタイプを持たず、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て、さらにこのサービスが返す出力を格納する割り当て済みの FML32 型付きバッファを使って、BEA Tuxedo システム MIB サービス、.TMIB を呼び出します。FML32 は自己記述型バッファタイプなので、入力バッファのバッファ・サイズを 0 と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行なわれる場合には TPNOTRAN フラグを使用しなければなりません。トランザクション内でなければ、このルーチンに対して定義されたフラグの使用については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 システムの MIB サービス、.TMIB を非同期的に呼び出します。FML32 は自己記述型バッファタイプなので、入力バッファのバッファサイズを 0 と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行なわれる場合には TPNOTRAN フラグを使用しなければなりません。トランザクション内でなければ、このルーチンに対して定義されたフラグの使用については条件や制約は一切ありません。

tpgetrply()
: 前に生成された BEA Tuxedo システムの MIB サービス、.TMIB の非同期呼び出しに対する応答を受信します。応答は前に割り当てられた FML32 型付きバッファに入れられます。このルーチンに対して定義されたフラグの使用については条件や制約は一切ありません。

tpenqueue()
: 後で処理するために、BEA Tuxedo システムの MIB サービス .TMIB に対する要求をキューに入れます。FML32 は自己記述型バッファタイプなので、入力バッファのバッファサイズを 0 と指定することができます。このルーチンに対して定義されたフラグの使用については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送出を処理するように環境設定された TMQFORWARD(5) サーバは、-n (TPNOTRAN フラグが設定された tpcall()) と -d (削除) オプションを指定して起動する必要があります。

tpdequeue()
: 前にキューに入れられた BEA Tuxedo システムの MIB サービス TMIB への要求に対する応答をキューから取り出します。応答は前に割り当てられた FML32 型付きバッファに入れられます。このルーチンに対して定義されたフラグの使用については条件や制約は一切ありません。

入力

BEA Tuxedo システムの任意の MIB に対する管理要求を特徴付けたり制御したりするときに使われる複数の FML32 フィールドがあります。これらのフィールドはヘッダ・ファイル <tpadm.h> にはもとより、このマニュアル・ページに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な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 要求バッファにも付加されます。これらのフィールドについて以下に説明し、そのあとで、各フィールドが必須指定、任意指定、または未使用のコマンド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 固有のマニュアル・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。システムが現在の検索位置を調べることができるように、アプリケーションは返された値を以後の要求に転送する必要があります。

TA_OCCURS
: GET または GETNEXT 操作時に取り出される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と、スペースが許すかぎり、一致するすべてのオブジェクトが返されます。

TA_FLAGS

共通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 マニュアル・ページ内に定義されています。共通のフラグ値と使用法を以下に示します。

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対して、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) の両方が存在します。これらのクラスから情報を取り出す要求は、効率性のために、デフォルトではローカル情報ではなくグローバル情報だけを取り出します。待ってでも複数のサイトからローカル情報を収集したいとアプリケーション・ユーザが望む場合は、取出し要求時にこのフラグをセットする必要があります。ローカル情報をもつクラスは、属性表の最後にリストされたローカル属性 (副見出しにローカル属性であることが示されている) を持っています。ローカル情報しかもたないクラスでは、このフラグ値がセットされていなくてもローカル情報を取り出します。

MIB_PREIMAGE

SET 操作が実行される前にプレイメージ・チェックに合格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、任意の MIB 固有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。一致したら、そのオブジェクトは任意の MIB 固有クラス属性のオカレンス 1 を使って更新されます。2 回以上発生しない属性はプレイメージ・チェックの対象とはみなされません。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はチェックされます。

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

返す必要が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 をもつ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 のリストはクラス固有属性を返しませんが、一致したクラス・オブジェクトの個数を返します。

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と、デフォルトの 20 に設定されます。

TA_CURSORHOLD
: 最初の GET 操作から生成されたシステム・スナップショットを、現在の GET または GETNEXT の要求が満たされた後、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た後でスナップショットを処分しなければなりません。指定しないと、この値はデフォルトの 120 に設定されます。

次表において、R は必須入力属性、O はオプションの入力属性、— は使用されない入力属性です。

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共通の出力フィールドの 1 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 は最初のオブジェクトに関連し、オカレンス 1 は 2 番目のオブジェクトに関連するという風に順次関連します。この指針の例外はコンポーネント MIB のマニュアル・ページに記載されています。FML32 定義の NULL フィールド値を操作実行後に持つことがあります。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する単一オブジェクトが返されます。GET 操作や GETNEXT 操作が正常終了すると、要求されたオカレンス数 (下記の 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。

重要なのは、オブジェクトの状態、相互運用的なリリース環境、入力要求フィルタによっては、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ついても返されるわけではないということです。管理プログラマは、出力バッファ内にある属性が存在するものと思うのではなく、属性値の存在を明示的に確認する必要があります。

繰り返しますが、正常に処理された管理要求は、すべての MIB に適用するある共通のフィールドを含んでいます。こ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。共通の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。以下に各共通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 固有のマニュアル・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
: 後で取り出すためにシステム・スナップショット内に保持されている、要求キー・フィールドが一致した追加のオブジェクトの数を示す、long 値の FML32 フィールド。SET 操作では、このフィールドは返されません。

TA_CURSOR
: システムが保持しているスナップ・ショット内の位置を示す、文字列値の FML32 フィールド。以降の 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。アプリケーション・ユーザがこのフィールドの値を解釈したり変更したりすることはできません。SET 操作では、このフィールドは返されません。

TA_ERROR
: 正常な終了を示す負値以外の戻りコードが入った、long 値の FML32 フィールド。共通のリターン・コードとその意味を以下に示します。

TAOK
: 操作が正常に実行された。アプリケーションの更新は行なわれなかった。

TAUPDATED
: アプリケーションの更新が正常に行なわれた。

TAPARTIAL
: アプリケーションの部分更新が正常に行なわれた。

MIB 固有システムサービス処理内で失敗した管理要求は、アプリケーション・サービス・エラーをアプリケーションに返します。これには、元々の要求とエラーの特徴を示す共通のフィールドが含まれています。アプリケーション・サービス・エラーは、tpcall() または tpgetrply() からの TPESVCFAIL エラー・リターン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・キューに入ります (サーバのコマンドラインで -d が指定されたと仮定して)。失敗した管理要求の特徴を示す共通のフィールドを以下に示します。

TA_ERROR
: 発生した特定のエラーを示す、long 値の FML32 フィールド。共通のエラー・コードの場合は 、このマニュアル・ページの DIAGNOSTICS セクションに記載され、コンポーネント MIB 固有のエラー・コードの場合は、個々のコンポーネント MIB マニュアル・ページに記述されます。

TA_STATUS
: エラーのテキスト記述が入る、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可能性がある場合に、そのフィールドのフィールド識別子が入る、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合せに起因する場合は、このフィールドの複数のオカレンスが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するために書かれたアプリケーション・プログラムは一定のヘッダ・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、このマニュアル・ページに記載されている 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 に固有の 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 と対話するためには、FML32 型付きバッファが該当するサービスに要求を送ることが必要です。ATMI 関数 tpalloc() は、FMLTYPE32 (<fml32.h> に定義されている) を使用してバッファを *type* 引数の値に割り当てます。FML32 バッファのサブタイプは存在しないので、tpalloc() の *subtype* 引数は NULL になる場合があります。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。これ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* に指定することで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 型付きバッファが割り当てられたら、ユーザはそのバッファに共通の MIB フィールドの値とコンポーネント MIB 固有の値を入れる必要があります。要求バッファに値を追加するときに使用する最も一般的なインターフェイスは、Fadd32() と 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() を使っ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 が設定されるようにする。
 */
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 (それらの属性はこのマニュアル・ページに定義されています)。

必須の共通属性は TA_OPERATION と TA_CLASS の 2 つです。

TA_OPERATION は、アクセスされる MIB 上で行われ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 マニュアル・ページに定義されています。TA_OPERATION が GETNEXT の場合には、さらに TA_CURSOR 属性も必要です。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、このあとの要求時に検索位置を調べるときにシステムが使用します

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 操作で取り出すオブジェクトの数を指定します。この属性を指定しないと、スペースが許すかぎり、すべてのオカレンスが取り出されます。

TA_FLAGS
: フラグ値を指定します。一部の共通フラグはこのマニュアル・ページに、他の共通フラグはコンポーネント MIB マニュアル・ページに定義されています。

TA_FILTER
: GET 操作に対して返される属性値を限定します。この属性を指定しないと、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
: コンポーネント MIB サービスでの、要求を満たすために使用できる時間 (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と、デフォルトの 20 に設定されます。

TA_CURSORHOLD

現在の GET または GETNEXT 操作が実行されたあと、最初の GET 操作から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する時間 (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た後でスナップショットを処分しなければなりません。指定しないと、この値はデフォルトの 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。以下の「MIB 要求の送信」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。
 * 上で生成された 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。以下の「MIB 要求の送信」参照。*/
```

コンポーネント MIB フィールド

GET または GETNEXT で指定されたコンポーネント MIB のキー・フィールドは、オブジェクトの集合を選択するときに使用されます。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 操作で指定されたコンポーネント 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されます。キー・フィールド以外のフィールドは、キー・フィールドによって特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が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指定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) から取られます。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は正確に一致していなければなりません。さもないと、要求は失敗します。クラスの一部で、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だけが、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); /* GET 1st matching occurrence */
/* 要求を作成。以下の「MIB 要求の送信」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 番目の一致するコピーを追加。*/
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。以下の「MIB 要求の送信」参照。*/
```

MIB 要求の送信

コンポーネント MIB 要求はすべて、コア BEA Tuxedo コンポーネント MIB サービス、.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ほか、他のコンポーネント MIB に対する要求を転送します。これで、ユーザ側ではサービス名を MIB やクラスとマッチングしなくても済みます。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()、tpenqueue()) を使って生成することができます。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フラグと機能にアクセスできます。ここで課せられる唯一の制約は、.TMIB サービスはトランザクションの範囲外で呼び出す必要があるということです。つまり、トランザクション内で tpcall() や tpacall() を使って管理要求を出すときに、TPNOTRAN フラグを使用しないと異常終了する (TPETRAN) ということです。tpenqueue() を使って要求を出すときには、送出されるサービス要求をトランザクション境界の外で行なうことができるように、TMQFORWARD サーバは -n オプションを指定して起動する必要があります。

例 :

```
/* 上に示すように要求を作成。*/
/* 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 はセットアップ済みと仮定。*/
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よって、3 通りの方法で受信できます。元の要求が tpcall() で生成された場合には、tpcall() は応答が受信されたという戻り値を返します。元の要求が tpacall() で生成された場合には、tpgetrply() を使っ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た場合には、tpdequeue() を使って応答を受信できます。これらのさまざまな呼び出し上でサポートされているフラグを適宜使用することができます。

例 :

```
/* 上に示すように要求を作成。*/
/* 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 をしようして応答を受信。バッファ・タイプの変更が必要な場合もあり。*/
flags = TPGETANY | TPSIGRSTRT;
rval = tpgetrply(rd, rpbuf, rplen, flags);
/* キューから要求を取り出す。qctl はセットアップ済みと仮定。*/
flags = TPNOCHANGE | TPSIGRSTRT;
rval = tpdequeue("queue", "replyq", qctl, rpbuf, rplen, flags);
```

MIB 応答の解釈

コンポーネント MIB に固有の属性のほかに、特定の共通 MIB フィールドが管理要求に対して返されることがあります。これらの属性は元の要求の結果の特徴を示し、必要な場合に以降の要求で使用できる値を指定します。

GET または GETNEXT 操作は、成功すると以下の値を返します。

- TA_CLASS

  クラス名

- TA_OCCURS

  取り出された一致オブジェクトの数。

- TA_MORE

  まだ取り出されていない一致オブジェクトの数。

- TA_CURSOR

  以降の検索で指定するカーソル。

- TA_ERROR

  負以外の戻り値 TAOK にセットします。

- 使用可能なすべてのコンポーネント MIB 固有属性

  各属性のオカレンス 0 は取り出された最初のオブジェクトを表し、オカレンス 1 は 2 番目のオブジェクトを表すという風に対応します。この規則の例外は、コンポーネント MIB マニュアル・ページの該当する箇所に示されています。

SET 操作は、成功すると以下の値を返します。

- TA_CLASS

  クラス名

- TA_ERROR

  負以外の戻り値にセットします。TAOK は、要求は成功したが情報は更新されなかったことを示しています。変更が指定されなかったり、指定された変更がオブジェクトの現在の状態と一致したりすると、このようなことが起こる可能性があります。TAUPDATED は 、要求が成功し情報が更新されたということを示しています。TAPARTIAL は、要求は成功したが 、更新はシステム内の一部のサイトにしか行なわれなかったことを示しています。このようなことはネットワーク障害やメッセージ輻湊が原因で起こることがあり、システムは更新されていないサイトをできるだけ早く同期させます。

- 使用可能なすべてのコンポーネント MIB 固有属性

  一度に 1 つのオブジェクトしか更新できないので、1 つのオブジェクトしか返されません。返された属性は更新後のオブジェクトを反映しています。

タイプにかかわらず、失敗した操作は以下の値を返します。

- 元の要求で指定されたフィールド。
- TA_ERROR

  失敗の原因を示す負以外の戻り値にセットされます。共通エラー・コードはこのマニュアル・ページの診断の項に定義されています。コンポーネント MIB 固有のエラー・コード (相互にも、また共通コードとも重複していない) は各 MIB マニュアル・ページに定義されています。

- TA_BADFLD

  不良フィールドのフィールド識別子。

- TA_STATUS

  エラー状態のテキスト記述。

制限事項

フィールドの複数オカレンスをもつ FML32 バッファは、オカレンスのシーケンス内の空きフィールドを考慮しません。たとえば、オカレンス 1 の値をセットし、オカレンス 0 がまだ存在しない場合、FML32 は FML32 が定義したヌル値でオカレンス 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がゼロの (ヌル) 文字列、文字フィールドに対しては '¥0' 文字です。このような制約のために、(異なる属性セットをもつオブジェクトを返すことがある) GET 操作は、オブジェクトの状態を正確に反映しないヌルの FML32 フィールドを含まないようにするために、ユーザに返されたオブジェクトの集合を人為的に解体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・クライアントは現在 64K の FML32 バッファにリンクされています。このため、システムは戻りバッファのサイズをバッファあたり 64K に制限しています。

COBOL は FML32 バッファタイプを限定的にしかサポートしていないので、ATMI の COBOL バージョンでは管理 API にアクセスできません。

コンポーネント MIB に対する要求はアプリケーション・トランザクションの一部にはなりえません。したがって、コンポーネント MIB に向けられ、アクティブ・トランザクション内で実行される 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、トランザクション内で ATMI 関数 tpenqueue() を使い、コンポーネント MIB への今後の送出に備えて要求をキューに入れることができます。この要求のキューへの登録はトランザクション内で起こりますが、コンポーネント MIB 内の処理は起こりません。このコンテキストで TMQFORWARD(5) 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 サービスに送出できるように、-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 サービスは非トランザクションの性質をもっているので、要求をリトライするのではなくサービス失敗がすぐに失敗キューに送出されるように、TMQFORWARD に対して -d オプションを指定することもお薦めします。

共通 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・アクションを混合する予定のアプリケーションは、必ずフィールド識別子を適切に割り当てる必要があります。

クラス記述

各クラスの説明セクションには、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た属性の詳細な説明

属性表
: クラスの各属性に関する名前、タイプ、パーミッション、値、およびデフォルト値を示す表。属性表の形式を以下に示します。

属性の意味
: 各属性の意味を説明します。

制限事項
: このクラスにアクセスしたり、このクラスを解釈する場合の制限事項。

属性表の形式

前述したように各クラスは 4 つの部分からなっており、1 つは属性表です。属性表はクラス内の属性をリストし、さらに管理者、オペレータ、一般ユーザが属性を使用してアプリケーションとインターフェイスをとる方法を示しています。属性表の各属性記述には 5 つの構成要素 (名前、タイプ、パーミッション、値、デフォルト値) があります。これらの各構成要素について以下に説明します。

名前：
: FML32 バッファ内のこの属性値を識別するために使用される FML32 フィールド識別子。属性は緊密に関連した属性がグループになって配置されることがあります。このグループ配置には特別な意味はなく、単に表を使いやすくするためです。名前や値の後に (r)、(k)、(x)、または (*) が付いていることがあります。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とき必要なフィールド

  (k) — オブジェクト検索用のキー・フィールドを示す

  (x) — オブジェクト検索用の正規表現キー・フィールドを示す

  (*) — このフィールドは、オブジェクト変更用の SET キー

  クラスに対する SET 操作は (上記の * を参照)、SET キーとして定義された 1 つまたは複数の属性値に対する値を含んでいなければなりません。指定する SET キーは、クラス内の 1 つのオブジェクトを正確に識別するのに十分なキーでなければなりません。SET キーは必ずオブジェクト検索用のキー・フィールドであり、したがって (k) 表記は指定されていませんが、暗黙的に (k) 表記されていると考えられます。ただし、新しい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、必要な場合は (r) 表記で示されます。

タイプ：
: 属性値のデータ型。データ型は C 言語の表記法、つまり long、char、string で定義されます。プログラムの中では、FML32 関数 Fldtype32( ) を使ってデータ型を判別することができます。この関数はデータ型を表す FML32 の define マクロ、つまり FLD_LONG、FLD_CHAR、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 (許可) 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。ただし属性表では、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を表しています。各グループについて、以下の意味を持つ 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	オブジェクト状態が ACTive または ACTive と同等な状態のときにだけ、属性を検索できます。また、各クラスの TA_STATE 属性値の説明を参照して、どの状態が ACTive 同等状態か判別してください。この属性は、オブジェクトの別々なアクティブ化にわたって一貫したものではない一時的な情報です。
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	検索または更新用のキー・フィールドとして、次にオブジェクト状態が ACTive または ACTive 同等の状態のときに、属性を指定できます。また、各クラスの TA_STATE 属性値の説明を参照して、どの状態が ACTive 同等状態か判別してください。

位置 2— 非アクティブ更新パーミッション

w	オブジェクトが INActive または INActive 同等状態のときに、属性を更新できます。各クラスの TA_STATE 属性の説明を参照して、どの状態が INActive 同等状態か判別します。
u	w パーミッション値と同様に属性を更新できます。さらに、u パーミッション文字で識別されたすべての属性値の組合せは、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。さらに、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— アクティブ更新パーミッション

x	オブジェクトが ACTive または ACTive と同等な状態のときにだけ、属性を更新できます。また、各クラスの TA_STATE 属性値の説明を参照して、どの状態が ACTive 同等状態か判別してください。

X	オブジェクトが ACTive または ACTive と同等な状態のときにだけ、属性を更新できます。また、各クラスの TA_STATE 属性値の説明を参照して、どの状態が ACTive 同等状態か判別してください。この属性は一時情報であり、この属性の更新はオブジェクトの別々のアクティベーションにわたって一貫したものではありません。
y	オブジェクトが ACTive または ACTive と同等な状態のときにだけ、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影響をおよぼす時点に対して制約があります。詳細については、クラスの「属性の意味」の項の属性の説明を参照してください。また、各クラスの TA_STATE 属性値の説明を参照して、どの状態が ACTive 同等状態か判別してください。

値 :

この属性に対してセットまたは検索 (もしくはその両方) が可能な値。属性値を表記する際の規則を以下に示します。

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 文字は含みません。
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*、*z*. の 1 つを選択します。
{*x*|*y*|*z*}	0 または *x*、*y*、*z* の 1 つを選択します。
{*x*|*y*|*z*},*	カンマで区切られた *x*、*y* または *z* の 0 または 1 つ以上のオカレンス。
low = *num*	*low* 以上の数値。
low = *num high*	*low* 以上で、*high* 未満の数値。

GET:	検索 (GET) 操作で返されるか、キー値として指定できる状態属性値。示される値はつねに 3 つの英字からなる簡略名です。完全名は該当するクラスの TA_STATE の説明中に示されています。入力指定は簡略名でも完全名でも可能で、大文字 / 小文字は区別されます。出力状態は必ずすべて大文字の完全名で返されます。
SET:	更新 (SET) 操作でセットできる状態属性値。GET の場合と同様に簡略名の使用が許されます。

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 INValid から NEW への状態変更時に使用されるデフォルト値。オブジェクトがアクティブのときに必要になるか、派生するか、または使用可能な属性については N/A の値が示されています。

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。簡潔にするために、TA_STATE 値は多くの場合 3 文字の簡略名で指定されます。TA_STATE 値の完全名が示されるときは、3 文字の簡略名は大文字で、残りの文字 (ある場合) は小文字で表します。TA_STATE 値の入力は簡略名でも完全名でも可能で、大文字 / 小文字は区別されます。TA_STATE 値の出力はかならず大文字の完全名です。TA_STATE 属性の使用例を以下に示します。

```
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。このクラスの主な用途はクラス名を識別することです。

属性表

MIB(5): T_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

MIB(5): T_CLASS **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_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) — オブジェクト検索用のキー・フィールドです。

属性の意味

TA_CLASSNAME:*string*
クラス名

TA_STATE:
GET:
GET 操作は、選択された T_CLASS オブジェクトの情報を検索します。下に示した状態は、a GET 要求に対する応答で返される TA_STATE の意味を示します。リスト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がこの状態で存在しています。この状態は、パーミッション・チェックに際しては INActive 同等状態です。

SET:
SET 操作は、このクラスでは許可されません。

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の、| で区切られたリスト。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同等の状態の、| で区切られたリスト。状態は大文字の完全名で返されます。

TA_SETSTATES:*string*
SET 操作の一部としてこのクラスのオブジェクトに対してセットされる可能性のある状態の、| で区切られたリスト。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

MIB(5): T_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*
クラス名システムが認識しているクラス名だけがアクセス可能。

TA_ATTRIBUTE:*long*
システム提供のヘッダ・ファイル（tpadm.h など）に定義されている属性フィールド識別子。

TA_STATE:

GET:VALid
GET 操作は選択された T_CLASSATT オブジェクトの情報を検索します。下に示した状態は、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_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がこの状態で存在しています。この状態は INActive と同等で、パーミッションの確認に使用されます。

SET:{NEW | INValid}

SET 操作は選択された T_CLASSATT オブジェクトの構成情報を更新します。次に、SET 要求で TA_STATE に設定できる値の意味を説明します。記述されていない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_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。VALid 状態のときにしか許されません。正常終了すると、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再設定します。状態の変更は、VALid 状態でのみ可能です。成功した場合、オブジェクトは INValid 状態また VALid 状態のままです。組み込まれている、つまりシステムに明示的に知らされている、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 状態のままで存在し続けます。クラス属性が明示的に知ら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、INValid 状態に変わります。

TA_PERM:0000 <= *num* <= 0777

このクラス属性の組合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ときに、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。属性に対して許される最大パーミッションは、管理向け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 属性は rw-r--r-- のパーミッションと記述されており、最大パーミッション rw-rw-rw- を持っています。

TA_FACTPERM:0000 <= *num* <= 0777
: BEA Tuxedo システムが出荷時に設定された、このクラス属性の組合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 操作の後に適用します。

TA_MAXPERM:0000 <= *num* <= 0777
: このクラス属性の組合せに対する最大パーミッション。

TA_ATTFLAGS:*long*
: この属性の特殊な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・フィールドである。

  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・フィールドである。

  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である。

  MIBATT_SETKEY
  : 属性はこのクラスの SET キーである。

  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オブジェクトに対して書き込み可能である。

TA_DEFAULT:*string*
: このクラスの NEW オブジェクトを作成するときの、この属性のデフォルト値。Admin API を介して NEW オブジェクトを作成できないクラスについては、この属性はかならず長さがゼロの文字列として返されることに留意してください。また、NEW オブジェクト作成時に SET できない属性も長さがゼロの文字列として返されます。*long* 値をもつ属性は、long 値を表す文字列として返されるデフォルト値を持ちます。ここで返される可能性がある以下のような特殊な値によって示される特殊な特性をもっている属性もあります。

  # Inherited:*Classname*[:*Attribute*]
  : 属性デフォルトは、このクラスの同名の属性から継承されます。*Attribute* が指定されている場合には、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`
: 属性はデフォルト値を定義するための特別な規則を持っています。詳細については、該当するコンポーネント MIB のマニュアル・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 / 属性の組合せに適用されるバリデーション（妥当性チェック）規則を表す文字列。この文字列は以下のフォーマットの 1 つをとります。

  `CHOICES=`*`string1`*`|`*`string2`*`|...`
  : 示された選択肢の 1 つだけと一致する文字列属性値。

  `RANGE=`*`min-max`*
  : *`min`* から *`max`* の値でなければならない数値属性値。

  `SIZE=`*`min-max`*
  : *`min`* から *`max`* のバイト数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バリデーションを持たない読取り専用属性。

  `SPECIAL=Y`
  : 特殊なバリデーション規則。詳細については、該当するコンポーネント MIB マニュアル・ページを参照。

  `UNKNOWN=Y`
  : 未知のバリデーション規則。通常、詳細がコアシステムに知らされていない追加のコンポーネント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 とインターフェイスする際にユーザに返されるエラーには、一般的な 2 つのタイプがあります。1 つ目は、管理要求に対する応答を検索するために使用される ATMI 関数 `tpcall()`、`tpgetrply()`、および `tpdequeue()` の 3 つが返すエラーです。これらのエラーについては、いずれも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。

第 2 のタイプのエラーは、要求がその処理機能を備えたシステム・サービスに正しく渡されたのち、そのサービスが要求を処理する際に問題があると判断した場合にアプリケーション・レベルのサービス・エラーとして返されるエラーです。サービスが要求を処理する際に問題があることが判明すると、tpcall() または tpgetrply() は tperrno() を TPESVCFAIL に設定してエラーを返すとともに、最初の要求および下に示したエラーを詳しく性格づけした TA_ERROR、TA_STATUS、あるいは TA_BADFLD フィールドを含む応答メッセージを返します。TMQFORWARD(5) 経由でシステムに転送された要求に対する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仮定します）。

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 フィールにエラーの内容について説明したテキストがセットされ、TA_ERROR という FML32 フィールドにはエラーの原因 (下記参照) を示す値がセットされます。TA_BADFLD にセットされる値については、下記のそれぞれの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、いずれも負であることが保証されています。

[TAEAPP]
: 要求が正しく処理されるためにはアプリケーションの協力を必要としたのに対し、アプリケーションが操作を終了させませんでした。サーバを停止させる場合などはアプリケーションの協力が必要になります。

[TAECONFIG]
: 要求を処理するため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。TA_BADFLD は、フィールド識別子が無効であることが示す値にセットされます。

[TAEOS]
: 要求を処理しようとした際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 の値は、システム・エラー・コード errno の値に更新されます。

[TAEPERM]
: ユーザが書き込みパーミッションを持たない属性を設定しようとしたか、または読み取りパーミッションを持たないクラスに対して GET を実行しようとしました。TA_BADFLD は、パーミッション検査にパスしなかったフィールド識別子を示す値にセットされます。

[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 を正しく実行できませんでした。TA_BADFLD は、プレイメージのチェックにパスしなかったフィールド識別子を示す値にセットされます。

[TAEPROTO]
管理要求が不正なコンテキストで送出されました。TA_STATUS には追加情報がセットされます。

[TAEREQUIRED]
必要なフィールド値が存在しません。TA_BADFLD は、抜け落ちているフィールドの識別子にセットされます。

[TAESUPPORT]
管理要求は、現在使用しているバージョンのシステムではサポートされません。

[TAESYSTEM]
要求を処理しようとした際に 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セットされます。

[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・キーを指定しませんでした。

[*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 に関するマニュアル・ページに記載されています。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、ここで定義した共通な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 に戻され、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]
操作は正しく実行されました。コンポーネント MIB のオブジェクトは更新されません。

[TAUPDATED]
操作は正しく実行されました。コンポーネント MIB のオブジェクトは更新されました。

[TAPARTIAL]
操作は一部正しく実行されました。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。共通の MIB 属性を定義したヘッダ・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 以降のバージョンで使用できます。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、それぞれのコンポーネントのマニュアル・ページで考察します。

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、さらにこのマニュアル・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・プラットフォームおよびワークステーション・プラットフォームで利用可能です。

使用例

共通の MIB の処理とインターフェイスする際の既存の API の簡単な使用例については、前述の使用方法のセクションを参照してください。詳しい使用例は、各コンポーネント MIB のマニュアル・ページで実際のコンポーネント MIB のクラスや属性を利用して説明されています。

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・タイプ

形式

```
#include <nl_types.h>
```

機能説明　このヘッダ・ファイルには、以下の定義が含まれています。

nl_catd
: メッセージ・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() が使用します。

nl_item
: langinfo() データの項目を識別するために nl_langinfo() が使用します。タイプ nl_item のオブジェクトの値は langinfo.h に定義されています。

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() が使用します。この定数は、識別子設定パラメータの値として、catgets() への以後の呼び出しに使用することができます。

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 システムのサーバで認識され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サーバは BEA Tuxedo システム提供のサーバのいずれかである場合と、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ある場合があります。

BEA Tuxedo システムでは、アプリケーションのコンフィギュレーション・ファイルで指定されたサーバ (およびその他のリソース) とともに機能する tmboot(1) コマンドと tmadmin(1) コマンドを介して、サーバが実行されます。servopts リストから必要な選択を行うと、それはコンフィギュレーション・ファイルのサーバとともに指定されます。認識されるオプションには、以下のものがあります。

-A

サーバがそれを構成しているすべてのサービスを最初から提供することを示します。BEA Tuxedo システム提供のサーバでは、-A がサービスを指定する唯一の方法です。

-s { *@filename* | *service*[,*service*...][:*func*] }

サーバを立ち上げたときに宣言されるサービスの名前を指定します。大部分のケースでは、サービスは同じ名前を持つ関数によって実行されます。すなわち、x というサービスは、x という関数によって実行されます。たとえば、次のように指定すると、

```
-s x,y,z
```

x、y、z の各サービスを最初から提供するサーバが実行されますが、これらの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。また、異なる名前の関数でサービスを実行することもできます。次のように指定すると、

```
-s x,y,z:abc
```

初期サービスが x、y、z であるサーバが実行されますが、これらのサービスは関数 abc によって処理されます。

カンマの間に空白は使用できません。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。明確なファンクション名、つまりコロンの後に指定された名前は最大 128 文字まで可能です。これらのの制限よりも長いものは短縮され、警告メッセージを伴います。`tmadmin(1)`、あるいは `TM_MIB(5)` で検索すると、最初の 15 文字だけ表示されます。

ファイル名は、-s オプションを使用してファイル名の前に "@" マークを付けることにより指定できます。このフィールドのそれぞれのライン (行) は、-s オプションに対する引数として処理されます。このファイルには、コメントを入力することができます。すべてのコメントは、"#" または ":" で始まります。-s オプションは指定サーバに対して複数回指定できます。

サーバ・ロード・モジュール内でサービス名と処理関数が実行時に関連することを、動的サービス機能と呼びます。`tmadmin advertise` コマンドを使用して、サーバ実行時に提供されるサービスのリストを変更することができます。

"." で始まるサービス名はシステムで予約されています。そのようなサービスをアプリケーション・サーバで指定すると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され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、再起動したサーバが前に起動したときと同じ標準エラー出力ファイルを必ずもつようにすることができます。このオプションを使用しない場合は、`stderr` というデフォルトのファイルが、`$APPDIR` により指定され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して、シングル・スレッドの RPC サーバと会話型サーバの両方で、サーバの自動アクティブ化と非アクティブ化をサポートすることができます。RPC サーバの場合、このオプションは、MAX に 1 より大きい値を指定した MSSQ で使用します。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。

サーバのアクティブ化 / 非アクティブ化は、キュー上にある「サーバあたり」の要求数で決定されます。ただし、RPC サーバでロード [L] 引数が使用されている場合は、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されているサーバでは期待どおりに機能しません。アイドル状態のサーバを停止するコードは `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くても、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。

使用しているサーバの種類に応じて、`-p` オプションの引数には次のような意味があります。

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 モードでのみ動作します。アクティブ化するサーバをさらに増やすかどうかは、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します。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、アクティブ化 / 非アクティブ化は発生しません。

low_water、*terminate_time*、*high_water* および *create_time*

これらの引数は、RPC サーバがサーバあたりのメッセージ数に基づいてアクティブ化 / 非アクティブ化されるときに、制御のために使用されます。ロードが、*high_water* の基準を *create_time* 秒以上にわたって超えた場合、新しいサーバがアクティブ化されます。ロードが、*low_water* の基準を *terminate_time* 秒以上にわたって下回った場合、1 つのサーバが非アクティブ化されます。*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か、または作業ロード 50 になります。*high_water* のデフォルト値は、サーバあたりの 2 つのメッセージの平均値、または作業ロード 100 になります。*create_time* のデフォルト値は 50、*terminate_time* のデフォルト値は 60 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 8.0 以降では、マルチスレッドまたは非 MSSQ 会話型サーバの自動アクティブ化に制限はありません。ただし、これらのサーバには、自動非アクティブ化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、会話型サーバがアクティブ化または非アクティブ化されるときに使用されます。一般に、会話型サーバは RPC サーバより長時間実行されるため、会話型サーバでは現在会話に接続しているサーバの *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、それぞれ関連する時間パラメータ terminate_time および create_time の値を超えた場合、サーバの最少数または最大数に余裕があれば、サーバがアクティブ化 / 非アクティブ化されます。

また、時間パラメータに値 0 秒を指定すれば、基準パーセンテージを超えたことが検出されるとすぐに、アクティブ化または非アクティブ化を実行することができます。*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

ハングアップしないようになっている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、この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、ロックするものがテキスト (TXTLOCK)、データ (DATLOCK) あるいはプロセス全体 (テキストとデータ - PROCLOCK) のいずれであるかにより決まります。詳細については、plock(2) を参照してください。サーバがルートとして実行されていない場合、ロックすることはできません。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*

引数に応じてサーバに対して nice を実行します。*prio* プロセスに高い優先順位 (負の引数) を与えるには、そのプロセスが root の UID で実行される必要があります。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、再起動したサーバが前に起動したときと同じ標準出力ファイルを必ずもつようにすることができます。このオプションを使用しない場合は、stdout というデフォルトのファイルが $APPDIR により指定されたディレクトリに作成されます。

-r

サーバに、実行したサービスのログをその標準エラー出力ファイルに記録させます。このログは txrpt(1) コマンドで分析することができます。-r を指定する場合は、ULOGDEBUG 変数を "y" に設定しないようにしてください。この変数を "y" に設定すると、デバッグ・メッセージが stderr に送信されなくなり、ファイル内のデバッグ・メッセージが、txrpt により間違って解釈されてしまいます。

-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 ソフトウェア間で相互運用を可能にします。サーバには、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、またはシステム・プロセスやアプリケーション・サーバ・プロセスを指定できます。ワークステーション・リスナ・プロセスの場合には、-t オプションを使用して起動すると、そのワークステーション・ハンドラ (WSH) プロセスのすべてに対して相互運用性が提供されます。

--

システムが認識する引数の最後と、サーバ内部のサブルーチンに渡される引数の最初にマークを付けます。このオプションは、ユーザがアプリケーション固有の引数をサーバに提供したい場合にのみ必要です。システムが認識するオプションは -- の前におき、アプリケーションの引数は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、その場合 getopt() を使用してそれらの引数を処理するようにします。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、その呼び出しが行われるとき、外部整数 optind は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>
. . . .
. . . .
```

最初の 3 行はコメントで、# で始まります。以下に続く各行には、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が示されます。buildserver コマンド行に "-s: *functionname*" 指定されている場合、どの行の *servicename* フィールドも空文字列になります。*functionname* フィールドは常に示されます。

注記 BEA Tuxedo システムでは、実行時に各サーバそれぞれのコマンド行に次のオプションを自動的に追加します。

-c dom=domainid
-c オプションは、ps コマンドの出力など、指定されたドメインに関連付けられたプロセスでレポートする、任意のコマンド出力にコメント行を追加します。コメント行には指定されたドメイン ID がレポートされます。このコメントを利用すると、複数のドメインを管理する管理者は、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できます。

使用例

UBBCONFIG(5) の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— コア 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性質を設定および管理するためのクラスの集合を定義します。これには、マシン、サーバ、ネットワークの管理、ロード・バランシングなどが含まれます。

TM_MIB(5) は、管理要求のフォーマットや管理応答の解釈を行うためには、共通 MIB のマニュアル・ページの MIB(5) とともに使用します。このマニュアルに記載されたクラスや属性を利用して MIB(5) の手順に従ってフォーマットした要求は、アクティブなアプリケーション中に存在するさまざまな ATMI インターフェイスを利用して管理サービスを要求するために使用できます。アクティブでないアプリケーションも、tpadmcall() 関数のインターフェイスを利用して管理することができます。
TM_MIB(5) のすべてのクラス定義に関連する追加情報については、403 ページ「TM_MIB(5) に関する追加情報」を参照してください。

TM_MIB(5) には次のクラスがあります。

TM_MIB のクラス

クラス名	属性
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
T_FACTORY	ファクトリ
T_GROUP	サーバ・グループ
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

TM_MIB のクラス (続き)

クラス名	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

それぞれのクラスの説明は、４つのセクションで構成されます。

- 概要： クラスの属性についての高レベルな説明
- 属性表： 属性表の形式については、以下で要約を示し、MIB(5) で詳しく説明します。
- 属性の意味： クラスの各属性の意味を定義します。
- 制限事項： このクラスの使用および解釈に関する制限事項

属性表の形式

この MIB の一部をなす個々のクラスは、下記のセクションの４つの部分で定義します。４つの部分の１つが属性表です。属性表はクラス内の属性をリストし、さらに管理者、オペレータ、一般ユーザが属性を使用してアプリケーションとインターフェイスをとる方法を示しています。

属性表には、個々の属性ごとに名前、タイプ、パーミッション、値、デフォルト値という５つの項目があります。各項目については MIB(5) で説明します。

TA_FLAG 値

MIB(5) は共通の TA_FLAGS 属性を定義します。long 型で、共通およびコンポーネントの MIB 固有フラグ値の両方が含まれます。下に示した値は、サポートされる TM_MIB(5) のフラグ値です。これらのフラグ値は共通 MIB フラグと組み合わせて使用します。

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だけをアクティブにすることを示すために使用されるフラグです。

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・プロセスだけを考慮することを示すために使用されるフラグです。このフラグは、また、T_SERVER と T_SERVERCTXT での検索をアプリケーション・サーバのみ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されたグループとサーバだけを考慮することを示すために使用されるフラグです。

TMIB_NOTIFY
: 選択した個々のサーバ・オブジェクトをアクティブまたは非アクティブにする直前および直後に、発信元のクライアントに任意通知型メッセージが送られるようにするために、T_MACHINE、T_GROUP、あるいは 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使用されるフラグです。

FML32 フィールド・テーブル

このマニュアル・ページに記述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 udataobj/tpadm で指定される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で区切ったリストに、またフィールド・テーブル名 tpadm は FIELDTBLS 環境変数で指定されるカンマで区切ったリストに、アプリケーションで含めなければなりません。

制限事項

この MIB のヘッダ・ファイルやフィールド・テーブルには、BEA Tuxedo リリース 6.1 以降のサイト (ネイティブ版および /WS 版の両方) でのみアクセスできます。

/WS 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。

プレイメージの処理（MIB_PREIMAGE フラグ・ビットをセット）のために、グローバル属性を持つクラスのローカル属性は考慮されません。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（例 : T_TLOG クラスの TA_TLOGCOUNT、TA_TLOGINDEX、TA_GRPNO、TA_TLOGDATA 属性）。

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

TM_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 操作のために 1 つ以上のキーが必要

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 このリリースでは、SET 操作は TA_NETGROUP DEFAULTNET だけを用います。GET オペレーションが両方の LMID 値のために定義された TA_NETGROUP を使うことがあります。

注 4 TA_SUSPTIME をセットできるのは、TA_STATE が現在 SUSPENDED にセットされている場合、またはこれから SUSPENDED に セットしようとする場合に限られます。
注 5 リンク・レベルでの暗号化の値 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の識別子（*LMID1*）と接続先の論理マシンの識別子（*LMID2*）。

TA_NETGROUP:*string*[1 . .30]

ネットワーク・グループの論理名。TA_LMID マシン識別子の接続先と接続元が同一の TA_NETGROUP にある場合、T_BRIDGE クラスが TA_NETGROUP ごとに関連しているフィールドのインスタンスを提供します。GET 要求のキー・フィールドとして TA_NETGROUP が使われることもあります。BEA Tuxedo リリース (リリース 6.4) の SET 操作では、DEFAULTNET 以外の TA_NETGROUP 値は使用されません。

TA_STATE:

GET:"{ACTive | INActive | SUSpended | PENding}"

GET 操作は、選択された T_BRIDGE オブジェクトについて実行時情報を検索します。1 つの論理マシンの識別子しか持たない TA_LMID 属性は、*LMID1* からアプリケーションの他のマシンへのすべての接続に一致します。この場合、それぞれの検索されたレコードは接続先の LMID が書き込まれて拡張された TA_LMID 属性の値を含みます。下に示した状態は、GET 要求に対する応答で返される TA_STATE の意味を示します。

ACTive	接続が確立され、アクティブな状態です。
INActive	接続はアクティブでない状態です。この状態は、特定の接続に関して状態を要求した場合、つまり、TA_LMID 属性で両方の LMID を指定し、接続元の論理マシンにアクセスできる場合にのみ返されます。
SUSpended	確立された接続がエラー条件の発生によって終了され、再接続が少なくとも TA_SUSPTIME 属性で指定された時間だけ保留にされたことを示します。この状態は、パーミッションの決定に際しては ACTive と同等です。
PENding	非同時接続が要求されましたが、完了していません。最終的な接続要求の結果はまだ決まっていません。

SET:"{ACTive | INActive | SUSpended | PENding}"

SET 操作は、選択された T_BRIDGE オブジェクトの実行時情報を更新します。次に、SET 要求で TA_STATE に設定できる値の意味を説明します。記述されていない状態を設定することはできません。

unset	既存の T_BRIDGE オブジェクトを変更します。この組合せは、ACTive または SUSpended 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変わりません。
ACTive	指定された論理マシン間の非同時接続を確立し、T_BRIDGE オブジェクトをアクティブな状態にします。1 つの論理マシン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、あるいは接続元の論理マシンにアクセスできない場合は、この操作は異常終了します。T_BRIDGE オブジェクトが非同時接続を確立している間、BRIDGE は他の仕事をします。PENding へ状態を変更することを推奨します。状態の変更は、INActive と SUSpended の状態にある場合にのみ許可されます。この状態の推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が考慮されます（--x--x--x）。正常終了の場合は、オブジェクトの状態は PENding になります。
INActive	指定された論理マシン間の接続を終了することによって、T_BRIDGE オブジェクトをアクティブでない状態にします。この操作は、一方のマシンしか指定しなかった場合、あるいは 2 つのマシンが接続されていない場合には異常終了します。状態の変更は、ACTive 状態でのみ可能です。正常終了すると、オブジェクトの状態は INActive になります。
SUSpended	指定された論理マシン間の接続を終了し、TA_SUSPTIME パラメータを指定された値にセットすることによって、T_BRIDGE オブジェクトを保留の状態にします。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 SUSpended になります。 制限事項 : 統計値は、接続元の論理マシンの観点から報告されるため、これらの統計値をリセットすると、接続先のマシンから同じ接続に対して報告される統計値との同期が失われます。

PENding	指定された論理マシン間の非同時接続を確立し、T_BRIDGE オブジェクトをアクティブな状態にします。1 つの論理マシン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、あるいは接続元の論理マシンにアクセスできない場合は、この操作は異常終了します。PENding の状態では、接続の要求が成功したか失敗したかは決定されません。しかし、接続が未処理でも、BRIDGE が他のイベントやデータを継続して処理します。状態の変更は、INActive と SUSpended の状態にある場合にのみ許可されます。この状態の推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が考慮されます（--x--x--x）。正常終了の場合は、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 から起算した現在の時間（単位は秒）。この属性は、下記の属性値から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 から起算した、この接続が最初に確立されたときの時間（単位は秒）。オープンされている経過時間（単位は秒）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）。この時間が過ぎると、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なることができます。

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られたバイト数。

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られたバイト数。

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ら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ら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・レベルでの暗号化の値 40 ビットは、後方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、アプリケーション中のアクティブなクライアントの実行時属性を表します。これらの属性の値は、実行中のアプリケーションのクライアントの動作を識別し、記録します。

属性表

TM_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

TM_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_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
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 . .78]	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

TM_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 クラスの属性は、すべてローカル属性です。
注 2 リンク・レベルでの暗号化の値 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_STATE:

GET:"{ACTive | SUSpended | DEAd}"

GET 操作は、選択された T_CLIENT オブジェクトについて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憶されません。したがって、最大限のパフォーマンスを実現するためには、クライアントの状態に関する照会はできるだけキー・フィールドを利用して制限する必要があります。下に示した状態は、GET 要求に対する応答で返される TA_STATE の意味を示します。

ACTive	T_CLIENT オブジェクトはアクティブな状態にあります。これは、クライアントがビジー状態かアイドル状態かを示すものではありません。TA_CURCONV 属性または TA_CURREQ 属性についての検索で非ゼロ値が得られた場合は、クライアントがビジー状態にあることを示します。
SUSpended	T_CLIENT オブジェクトがアクティブな状態にあり、次のサービス要求 (tpcall() または tpacall()) の実行や新たな会話の開始 (tpconnect()) が保留されていることを示します。詳しくは、下記の SET SUSpended の項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際しては 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な状態として識別されているのに、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。この状態は、クライアントのローカル BBL が終了を検知し、クライアントの掲示板のリソースをクリアするまでしか保持されません。この状態は、パーミッションの決定に際しては ACTive と同等です。

SET:"{ACTive | SUSpended | DEAd}"

SET 操作は、選択された T_CLIENT オブジェクトの実行時情報を更新します。次に、SET 要求で TA_STATE に設定できる値の意味を説明します。記述されていない状態を設定することはできません。

ACTive	SUSpended 状態の T_CLIENT オブジェクトをアクティブな状態に切り替えます。状態の変更は、SUSpended 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 ACTive になります。
unset	既存の T_CLIENT オブジェクトを変更します。この組合せは、ACTive または SUSpended 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変わりません。
SUSpended	T_CLIENT オブジェクトを保留の状態にし、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())、新たな要求の送出 (tpenqueue()) が実行できないようにします。トランザクションの途中のクライアントは、トランザクションを中途終了またはコミットするまではこれらのルーチンを呼び出すことができますが、その場合にはそれらは保留されます。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 オブジェクトを中途終了による非アクティブな状態に切り替えます。状態の変更は、ACTive または SUSpended 状態でのみ可能です。クライアントを非アクティブな状態にする方法としては、まず最初に警告メッセージをブロードキャストで送出し (tpbroadcast())、次にクライアントを保留の状態にし (上記の SET SUSpended を参照)、最後に状態を DEAd に設定して中途終了による非アクティブ状態に切り替えることをお勧めします。正常終了の場合は、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 の状態にセットすることはできません。 プラットフォームあるいはシグナルの制約により、システムがクライアントを *kill* することができない場合があります。そのような場合は、ネイティブなクライアントは次に ATMI にアクセスする際に途中終了され、WSH へのワークステーション・クライアント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() 実行時にデータ構造体 TPINIT の cltname メンバを通じ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最後に ATMI コールを通じてシステムと対話してからのおおよその経過時間（単位は秒）。この値の誤差は、TA_SCANUNIT の示す秒数以内です（T_DOMAIN クラス参照）。キー・フィールドとして指定した場合、正の値のときはアイドル時間の値がその値以上のすべてのクライアントが一致し、負の値のときはアイドル時間の値がその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該当します。

TA_LMID:*LMID*
: クライアントを実行している論理マシン（ネイティブのクライアント）、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 ID。ワークステーション・クライアントの場合には、この識別子はクライアントを接続するため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。呼び出し元のプロセスのクライアント情報を検索する場合は、GET 操作で負の値を指定できます。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。このデータは、tpinit() 実行時にデータ構造体 TPINIT の grpname メンバを利用してセットすることができます。

TA_USRNAME:*string*[0..30]

tpinit() 実行時にデータ構造体 TPINIT の usrname メンバを通じ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" にセットされているときは、指定されたクライアントがリモートの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" にセットされているときは、指定されたクライアントがワークステーション・ハンドラ・プロセスです。

TA_WSHCLIENTID:*string*[1..78]

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とき（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となり、それ以外の場合はこの文字列は長さがゼロの文字列として返されます。

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。/WS クライアント（TA_WSC==Y）の場合は、この値は、/WS クライアントがアプリケーションにアクセスする際に使用する「アプリケーションによって管理されるマシン」に対するメジャー・リリース番号とは異なる場合があります。

TA_WSPROTO:0 <= *num*

BEA Tuxedo システムの Workstation プロトコルのバージョン番号。この値は、/WS プロトコルを更新するたびに変更されます。/WS 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、この属性に対しては値ゼロが返されます。

TA_NUMCONV:0 <= *num*
: このクライアントが tpconnect() によって開始した会話の件数。

TA_NUMDEQUEUE:0 <= *num*
: このクライアントが tpdequeue() によって開始したキューからの取り出し操作の回数。

TA_NUMENQUEUE:0 <= *num*
: このクライアントが tpenqueue() によって開始したキューへの追加操作の回数。

TA_NUMPOST:0 <= *num*
: このクライアントが tppost() によって開始したポストの件数。

TA_NUMREQ:0 <= *num*
: このクライアントが tpcall() または tpacall() を通じて送出した要求の件数。

TA_NUMSUBSCRIBE:0 <= *num*
: このクライアントが tpsubscribe() によって行ったサブスクリプションの件数。

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件数。

TA_NUMTRANABT:0 <= *num*
: このクライアントが中途終了したトランザクションの件数。

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件数。

TA_CMTRET:"{COMPLETE | LOGGED}"
: このクライアントに対する 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() の記述を参照してください。

TA_CURCONV:0 <= *num*
: このクライアントが tpconnect() によって開始した会話のうち、現在もアクティブな会話の件数。

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、リンクの確立時に調整されます。

  注記 リンク・レベルでの暗号化の値 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。

TA_CURREQ:0 <= *num*
: このクライアントが tpcall() または tpacall() によって開始した要求のうち、現在もアクティブな要求の件数。

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 から起算した現在の時間（単位は秒）。この属性は、T_CLIENT:TA_TIMESTART 属性の値から経過時間を算出する場合に利用できます。

TA_LASTGRP:1 <= *num* < 30,000

最後になされたサービス要求、あるい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 . . 78]

ワークステーション・クライアントに対しては、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。表示不能な文字を含むネットワーク・アドレスは、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の場合にも、有効な 16 進数の偶数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 アドレスの場合には、以下のいずれかの形式が使用されます。

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

各 # 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、システムで使用される基本トランスポート・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でないクライアントでは、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する能力は、使用するトランスポート・プロバイダによって決定されます。プロバイダがこの情報を使用できるようにしない場合は、ワークステーション・クライアントが関連するアドレスを持てない場合もあります。

TA_NOTIFY:"{DIPIN | SIGNAL | THREAD | IGNORE}"

このクライアントの通知特性の設定。詳しくは、T_DOMAIN クラスのこの属性に関する説明をお読みください。

TA_NUMUNSOL:0 <= *num*

このクライアント宛てに送出された処理待ちの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。アプリケーションが動作しているプラットフォームが UNIX ベース以外の場合、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在待っている応答を受け取る際の、タイムアウトまでの残り時間（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結合した時間。T_CLIENT:TA_LMID で time(2) システム・コールから返される時間で、1970 年 1 月 1 日の 00:00:00 UTC から起算して計算されます。

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。値がゼロのときは、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、アプリケーション中のアクティブな会話の実行時属性を表します。

属性表

TM_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" SET:N/A	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 クラスの属性は、すべてローカル属性です。

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、選択された T_CONN オブジェクトの実行時属性を検索します。下に示した状態は、GET 要求に対する応答で返される TA_STATE の意味を示します。

ACTive	返されるオブジェクトは、アプリケーション中のアクティブな会話の一方または両側の状態を反映します

SET:

SET 操作は、このクラスでは許可されません。

TA_SERVICENAME:*string*[1..15]
呼び出し元によって呼び出され、下位サーバ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
会話の呼び出し元のサーバ・グループ番号。呼び出し元がクライアントである場合は、この属性の値として返される値は 30,000 という値になります。

TA_CONNOLMID:*LMID*
呼び出し元を実行している場所、あるいは呼び出し元がアプリケーションにアクセスしている場所（/WS クライアントの場合）を示す論理マシン識別子。

TA_CONNOPID:1 <= *num*
会話の呼び出し元のプロセス ID。

TA_CONNOSNDCNT:0 <= *num*
呼び出し元が tpsend() を呼び出した回数。

TA_CONNOSRVID:1 <= *num* < 30,001
会話の呼び出し元のサーバ識別子。

TA_CONNSGRPNO:1 <= *num* < 30,001
会話の下位サーバのサーバ・グループ番号。

TA_CONNSLMID:*LMID*
下位サーバを実行している場所、あるいは下位サーバがアプリケーションにアクセスしている場所（/WS クライアントの場合）を示す論理マシン識別子。

TA_CONNSPID:1 <= *num*
会話の下位サーバのプロセス ID。

TA_CONNSSNDCNT:0 <= *num*
下位サーバが tpsend() を呼び出した回数。

TA_CONNSSRVID:1 <= *num* < 30,001
会話の下位サーバのサーバ識別子。

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・リストの記憶に使用される raw ディスク・スライスあるいは UNIX システムのファイルのコンフィギュレーションと実行時属性を表します。このクラスは、raw ディスク・スライスあるいは UNIX システムのファイルに記憶されるデバイス・リストのエントリの作成および削除を可能にします。

属性表

TM_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 クラスの属性は、すべてローカル属性です。

注 2 SET 操作では、デバイス・リストの新たなエントリを作成するために状態を NEW にセットする場合を除いて、デバイス・リストの特定のエントリを指定するために TA_DEVINDEX が必要となります。デバイス・リストの新たなエントリを作成するために状態を NEW にセットする場合は、値はエントリの作成が正しく実行された時点でシステムによって割り当てられ、返されるため、TA_DEVINDEX はセットしないでください。

注 3 TA_DEVSIZE を セットできるのは、オブジェクトを作成する場合に限られます。

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、環境設定が済んでいる（少なくとも 1 つの T_MACHINE の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。環境設定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際にこの属性が指定されていた場合、この属性は無視されます。

TA_CFGDEVICE:*string*[2..64]

BEA Tuxedo のファイル・システムが記憶されている、あるいは記憶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 で指定された BEA Tuxedo システム VTOC で使用する際の、TA_DEVICE のスタート・ポイントに相当するオフセット（単位はブロック）。

制限事項 : この属性は、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) に対しては、0 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され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 に切り替える場合にのみセットできます。

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 のファイル・システムの特定のデバイスに関係のある属性を取得する場合と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、選択された T_DEVICE オブジェクトの実行時情報を検索します。下に示した状態は、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	TA_CFGDEVICE が示す BEA Tuxedo のファイル・システムが存在し、有効なデバイス・リストが含まれていることを示します。TA_DEVICE は、デバイス・インデックス telnetlchome3 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、選択された T_DEVICE オブジェクトの情報の更新、あるいは指定されたオブジェクトの追加を実行します。次に、SET 要求で TA_STATE に設定できる値の意味を説明します。記述されていない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_DEVICE オブジェクトを作成、または初期化しなおします。状態の変更は、INValid または VALid の状態にある場合にのみ許可されます。正常終了すると、オブジェクトの状態は VALid になります。INValid の状態にあるときにこの状態の切り替えが呼び出された場合はオブジェクトが作成され、それ以外の場合はオブジェクトは初期化し直されます。TA_CFGDEVICE BEA Tuxedo ファイル・システムでデバイス・リストに最初の TA_DEVICE デバイスを作成する際には、TA_CFGDEVICE 上に必要な VTOC 構造体と UDL 構造体が自動的に作成され、初期化されます。ある TA_CFGDEVICE に対して作成したデバイス・リストの最初のエントリは、TA_DEVICE 属性に対して等価の値を持ってい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 は特別なものであるため、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。これらの属性の値は、BEA Tuxedo システム のアプリケーションの識別、カスタマイズ、サイズの指定、セキュリティ調節に使用されます。ここに記載した属性値の多くは、この MIB で示す他のクラスに対してアプリケーションのデフォルト値として使用されます。

個々のアプリケーションに対しては、かならず 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、このクラスではキー・フィールドは定義されていません。このクラスに対する GET 操作は、かならずこの唯一のオブジェクトを表す情報を返します。同様に、SET 操作は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

TM_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}" SET:"{NEW | INV | ACT | INA | FIN}"	N/A 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
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

TM_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_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_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"
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"

TM_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_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)
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

TM_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_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"

(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セットします。

属性の意味

TA_IPCKEY:32,769 <= *num* < 262,144

BEA Tuxedo システムの掲示板の「周知のアドレス」に対する数値キー。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「指名」します。複数のプロセッサを使用している環境や LAN 環境では、このキーは DBBL のメッセージ・キューを指名します。また、このキーは、周知のアドレス以外のリソースの名前（たとえば、アプリケーションで使用する掲示板の名前など）を得る際にも使用されます。

TA_MASTER:*LMID1*[,*LMID2*]

マスタの論理マシン（*LMID1*）とバックアップ用の論理マシン（*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*）は、ローカル・マシンに一致していなければなりません。SHM モードのアプリケーション（下記の TA_MODEL の項参照）は、マスタの論理マシンの識別子だけをセットできます。ACTive 状態の MP アプリケーション（下記の TA_MODEL 参照）でのこの属性値の変更の意味は、次のようになります。
現在アクティブなマスタの LMID を A、現在のバックアップ・マスタの LMID を B、セカンダリの LMID をそれぞれ C、D、... とすると、実行中の MP モードのアプリケーションでの TA_MASTER 属性に対して認められた変更は、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・マスタの LMID の指定を 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 な状態にあり、アクセス可能な場合に行われます。それ以外の場合は、分断方式の移行になります。ネットワークへの新たな接続を新たに確立する場合、あるいは改めて接続を確立しなおす場合は、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。それ以外の場合は、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・マシンは、常にそのアプリケーションの他のすべてのマシンと同じかまたはそれよりも大きい BEA Tuxedo のリリース番号を持っていなければなりません。また、マスタ・マシンとバックアップ・マシンが持っているリリース番号は同じでなければなりません。TA_MASTER 属性の変更は、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 は単一マシン・コンフィギュレーションを設定します。1 つの T_MACHINE オブジェクトを指定できます。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選択することを意味します。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や実行時の情報を検索します。下に示した状態は、GET 要求に対する応答で返される TA_STATE の意味を示します。

ACTive	T_DOMAIN オブジェクトが定義されていて、マスタ・マシンがアクティブな状態にあることを示します。
INActive	T_DOMAIN オブジェクトが定義されていて、アプリケーションがアクティブでない状態にあることを示します。

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や実行時の情報を更新します。次に、SET 要求で TA_STATE に設定できる値の意味を説明します。記述されていない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_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。正常終了すると、オブジェクトの状態は INActive になります。この変更は、TA_MASTER から類推された TA_LMID、ローカルのシステム名に基づく TA_PMID、および環境変数である TUXCONFIG と TUXDIR に基づいて決定された TA_TUXCONFIG と TA_TUXDIR を持つ新しい T_MACHINE オブジェクトも作成します。T_MACHINE E クラスのその他の設定可能な属性は、T_DOMAIN NEW 要求に値をセットすることによりこの時点でセットすることができます。TA_APPDIR の値が指定されていない場合は、カレント・ディレクトリがデフォルト・ディレクトリとして使用されます。
unset	T_DOMAIN オブジェクトを変更します。変更は、ACTive または INActive の状態にある場合にのみ許可されます。正常終了すると、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、その他）をアクティブな状態に切り替えます。この状態の推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が考慮されます（--x--x--x）。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 INActive になります。
INActive	マスタ・マシンの管理プロセス（DBBL、BBL、その他）をアクティブでない状態に切り替えます。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 INActive になります。
FINactive	マスタ・マシンの管理プロセス（DBBL、BBL、その他）を強制的にアクティブでない状態に切り替えます。シャットダウンを許可するかどうかの決定に際しては、アタッチされた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 グラフィカル管理プログラムが 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 クラスの新たに環境設定され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、アクティブな、あるいはすでに環境設定の済んでいる T_MACHINE オブジェクトには影響はありません。

TA_GID:0 <= *num*
: T_MACHINE クラスの新たに環境設定され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、アクティブな、あるいはすでに環境設定の済んでいる T_MACHINE オブジェクトには影響はありません。

TA_PERM:0001 <= *num* <= 0777
: T_MACHINE クラスの新たに環境設定され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、アクティブな、あるいはすでに環境設定の済んでいる T_MACHINE オブジェクトには影響はありません。

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・マスター・バイナリでない場合は 0 文字の文字列。

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値。バイナリが BEA Tuxedo システム・マスター・バイナリでない場合は -1

TA_LICSERIAL:*string* [0..78]
マシンのバイナリのシリアル・ナンバー。

TA_MIBMASK:0 <= *num* <= 0777
属性のアクセス・マスク。この属性によって指定されたユーザ・タイプとアクセスモードの組合せは、現在ではこのマニュアルで定義したクラスと属性のあらゆる組合せに使用できるわけではありません。たとえば、0003 とセットした場合は、管理者または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デフォルトの最大数。指定されない場合、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 クラスで変更することができます。

BBL、restartsrv、cleanupsrv、tmshutdown()、tmadmin() など、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てください (システム提供のサーバの例としては、AUTHSVR、TMQUEUE、TMQFORWARD、TMUSREVT、TMSYSEVT、TMS があります。T_GROUP TA_TMSNAME 属性、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 (6.5 以前) では、アプリケーションの TA_MAXACCESSERS と TA_MAXSERVERS 属性は、ユーザ・ライセンス数をチェックする仕組みと関連付けられていました。特に、あるマシンの TA_MAXACCESSERS とアプリケーションで既に動作しているマシンの TA_MAXACCESSERS の合計が、アプリケーションの TA_MAXSERVERS とユーザ・ライセンス数の合計より大きい場合には、その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 TA_MAXACCESSERS の合計数は、アプリケーションの TA_MAXSERVERS とユーザ・ライセンス数の合計数以下である必要があります。

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 点だけです。すべてのユーザ・ライセンスが使用中の場合、新しいクライアントがアプリケーションに参加することはできません。

制限事項 : この属性を変更しても、アクティブな、あるいはすでに環境設定の済んでいる T_MACHINE オブジェクトには影響はありません。

`TA_MAXCONV:0 <= num < 32,768`

このアプリケーション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。この値が指定されていない場合、T_SERVER クラスに何らかの会話サーバが定義されていれば、64 がデフォルト値になり、そうでない場合は 1 になります。1 サーバ当たりの同時の会話の最大数は、64 です。この属性の T_DOMAIN の値は、マシンごとに T_MACHINE クラスで変更することができます。

制限事項 : この属性を変更しても、アクティブな、あるいはすでに環境設定の済んでいる T_MACHINE オブジェクトには影響はありません。

`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。指定されない場合、デフォルトの 100 が設定されます。この属性の T_DOMAIN の値は、マシンごとに T_MACHINES クラスで変更することができます。

制限事項 : この属性を変更しても、アクティブな、あるいはすでに環境設定の済んでいる T_MACHINE オブジェクトには影響はありません。

`TA_MAXBUFSTYPE:1 <= num < 32,768`

掲示板のバッファのサブ・タイプのテーブルに登録できるバッファのサブ・タイプの最大数。

`TA_MAXBUFTYPE:1 <= num < 32,768`

掲示板のバッファ・タイプのテーブルに登録できるバッファ・タイプの最大数。

TA_MAXDRT:0 <= *num* < 32,768

掲示板のバッファ・タイプのテーブルに登録できるバッファ・タイプの最大数。T_ROUTING クラスの各オブジェクトごとにエントリが 1 つ必要です。実行時にテーブルを拡大できるようにするには、新しい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・テーブルに登録できるサーバ・グループの最大数。

制限事項 : BEA Tuxedo リリース 4.2.2 以前のバージョンを使用しているサイトでは、この属性の値は 100 に固定されます。これらのサイトとの相互運用性を持たせるためには、使用中のサーバ・グループの数が常に 100 以下になるように制限する必要があります。リリース 4.2.2 以前を使用している場所では、定義されたサーバ・グループの数が 100 を超えるアプリケーションに結合することはできません。また、4.2.2 以前のバージョンを使用しているサイトをすでに含むアプリケーションは、100 を超えてサーバ・グループを追加することは認められません。

TA_MAXNETGROUPS:1 <= *num* < 8,192

TUXCONFIG ファイルの NETWORK セクションの中で適合しうるように環境設定されたネットワーク・グループの最大数値を指定します。この値は 1 以上、8192 未満でなければなりません。指定されない場合、デフォルトの 8 が設定されます。

TA_MAXMACHINES:256 <= *num* < 8,191

掲示板のマシン・テーブルに登録できるマシンの最大数。

制限事項 : BEA Tuxedo リリース 4.2.2 では、この属性の値は 256 に固定されます。また、リリース 4.2.2 より古いバージョンでは、この属性の値は 50 に固定されます。リリース 4.2.2 以前のバージョンを使用しているサイトとの相互運用性を持たせるためには、マシン・テーブルの使用中のマシン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 を使用しているサイトでは、定義されたマシンの数が 256 を超えるアプリケーションには結合することはできません。リリース 4.2.2 より古いバージョンを使用しているサイトでは、定義されたマシンの数が 50 を超えるアプリケーションに結合することはできません。また、アクティブなリリース 4.2.2 以前のバージョンを使用しているサイトをすでに含むアプリケーションは、適用される最も小さい最大値を超えてマシンを追加することはできません。

TA_MAXQUEUES:1 <= *num* < 8,192

掲示板のキューのテーブルに登録できるキューの最大数。

制限事項 : リリース 4.2.2 以前のバージョンを使用しているサイトは、TA_MAXQUEUES の設定値が TA_MAXSERVERS の設定値と等しい場合に限り、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
: 掲示板のルーティング基準の範囲テーブルに登録できるルーティング基準の範囲の最大数。TA_RANGES の設定に含まれる個々の範囲ごとに 1 つの項目が必要で、それ以外に T_ROUTING クラスのオブジェクト 1 つにつき 1 つの追加項目が必要です。実行時にテーブルを拡大できるようにするには、新しい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・テーブルに登録できる文字列プール領域の最大数。文字列プールには、TA_RANGES の値で指定された文字列と CARRAY が書き込まれます。実行時の拡大を可能にするためには、項目を余分に割り当てておく必要があります。

TA_MAXSERVERS:1 <= *num* < 8,192
: このアプリケーションの掲示板のサーバ・テーブルに登録できるサーバの最大数。指定されない場合、デフォルトの 50 が設定されます。

  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の例としては、AUTHSVR、TMQUEUE、TMQFORWARD、TMUSREVT、TMSYSEVT、TMS(T_GROUP TA_TMSNAME 属性を参照)、TMS_QM、GWTDOMAIN、および WSL 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 TA_MAXSERVERS の数に入れてください。

TA_MAXSERVICES:1 <= *num* < 32,766
: 掲示板のサービス・テーブルに登録できるサービスの最大数。指定されない場合、デフォルトの 100 が設定されます。

  この属性をセットする場合、システム提供の管理サーバを考慮に入れる必要があります。BEA Tuxedo システムを使用しているサイトを管理するためには、サイト 1 つにつき約 5 つのサービスが必要になります。それ以外の /WS、/Q、/DM といった管理コンポーネントについても、管理サービスを追加し、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。定義可能な最大のグループ ID 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 システムの ATMI 機能 & tpscmt() を参照してください。

制限事項 : 実行時にこの属性が変更されても、アクティブなクライアントやサーバには影響はありません。

TA_LDBAL:{Y | N}

ロード・バランシング機能のオン / オフ（Y/N）を切り替えます。
制限事項 : 実行時にこの属性が変更されても、アクティブなクライアントやサーバには影響はありません。

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ついて使用する通知検知手段のデフォルト設定。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 という値は、ディップ・イン方式の通知検知手段を使用することを示します。これは、ATMI コール中ではシステムはクライアント・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 コール中での検出ポイントは、システムによっては定義されず、ブロック中のシステム・コールがディップ・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のデフォルト設定です。

SIGNAL という値は、シグナル・ベースの通知検出手段を使用することを示します。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キャッチするルーチンをインストールします。

値 THREAD は、THREAD 通知を使用することを示します。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 アプリケーション結合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、マルチスレッディング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 通知を使用することはできず、THREAD が有効になっている場合は DIPIN がデフォルト値になります。

IGNORE という値は、デフォルト設定では通知メッセージがアプリケーションのクライアントに無視されるように指定します。これは、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 : 実行時にこの属性が変更されても、アクティブなクライアントに影響はありません。ネイティブ・クライアント・プロセスのすべてのシグナル処理は、管理システム・プロセスによって行われ、アプリケーション・プロセスが行うのではありません。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動作しているネイティブ・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は、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 システムのライブラリによって、BEA Tuxedo システムの内部テーブルにアクセスするために、アプリケーションのプロセス中で使用されるデフォルト・モードです。FASTPATH は、BEA Tuxedo システムのライブラリが高速アクセス用のプロテクト（保護）されていない共有メモリを利用して BEA Tuxedo システムの内部テーブルにアクセスが可能であることを指定します。PROTECTED は、BEA Tuxedo システムの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にプロテクトされた共有メモリを利用して BEA Tuxedo システムの内部テーブルにアクセスが可能であることを指定します。NO_OVERRIDE を指定すると、アプリケーション・プロセスでは tpinit(3c) または TPINITIALIZE(3cbl) で使用可能なフラグによって選択モードの変更はできないことを示すことができます。

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すると、新しく起動したクライアントと新しく環境設定された T_SERVER オブジェクトのみ影響を受けます。

(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。あるスレッドが BEA Tuxedo コードを実行中、つまりスレッドが掲示板にアタッチされている間に、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からです。BEA Tuxedo システムでは、このような状況を防止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・オプションのリスト（区切りにはカンマを使用）。使用可能なオプションについては、下記で定義します。

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 モードのみ）

NO_XA—XA トランザクション使用不可

NO_AA— 監査および認証のプラグインは呼び出されない

制限事項 : アクティブなアプリケーションでは、ACCSTATS だけを設定または再設定できます。

TA_USIGNAL:{SIGUSR1 | SIGUSR2}
シグナル・ベースの通知に使用されるシグナル（上記の TA_NOTIFY の項参照）。

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。長さゼロの文字列または NONE は、セキュリティ機能がオフであるか、またはオフになることを示します。識別子 APP_PW は、パスワードによるアプリケーションのセキュリティ機能が有効になることを示します（クライアントは、初期化時にアプリケーションのパスワードを渡す必要があります）。この属性の設定には、長さがゼロでない値の TA_PASSWORD 属性が必要になります。識別子 USER_AUTH は、APP_PW とよく似ていますが、さらに、クライアントの初期化時に各ユーザごとの認証も実行されることを示します。識別子 ACL は、USER_AUTH とよく似ていますが、さらに、サービス名、キューの名前、イベントの名前に対するアクセス制御チェックが実行されることを示します。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。識別子 MANDATORY_ACL は、ACL とよく似ていますが、名前に対応する ACL が見つからない場合は、パーミッションは与えられません。

注記 TA_OPTIONS パラメータの NO_AA 値が有効の場合、セキュリティ・パラメータ NONE、APP_PW、および USER_AUTH は正しく動作しますが、認証や監査は実行されません。残りのセキュリティ・モード ACL と MANDATORY_ACL は正しく動作しますが、デフォルトの BEA セキュリティ・メカニズムのみを使用します。

TA_PASSWORD:*string*[0 . . 30]
クリア・テキストのアプリケーション・パスワード。この属性は、TA_SECURITY 属性の値がセットされていない場合は無視されます。システムは、管理者に代わってこの情報を自動的に暗号化します。

TA_AUTHSVC:*string*[0..15]
システムに結合するそれぞれのクライアントに対して、アプリケーションの認証サービスが呼び出されます。TA_SECURITY 属性の値がセットされていない場合、あるいは APP_PW にセットされている場合は、この属性は無視されます。

TA_SCANUNIT:0 <= *num* <= 60 (5 の倍数)
: システムが周期的に実行するスキャンの間隔（単位は秒）。周期的なスキャンは、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ク中の呼び出しを検出するために用いられます。TA_BBLQUERY、TA_BLOCKTIME、TA_DBBLWAIT、および TA_SANITYSCAN は、この値の乗算子です。SET 操作でこの属性値としてゼロが渡されると、属性値はデフォルト値にリセットされます。

TA_BBLQUERY:0 <= *num* < 32,768
: 登録された BBL に関する DBBL 状態チェックの周期を示す TA_SCANUNIT 属性の乗算子。DBBL は、すべての BBL の報告が TA_BBLQUERY で指定した周期でなされるかどうかをチェックします。BBL からの報告がなかった場合は、DBBL はその BBL にメッセージを送り、状態を照会します。応答がない場合、BBL は分断されます。SET 操作でこの属性値としてゼロが渡されると、属性値はデフォルト値にリセットされます。この属性は、TA_SANITYSCAN 属性（下記参照）の値の少なくとも 2 倍以上の値にセットする必要があります。

TA_BLOCKTIME:0 <= *num* < 32,768
: ブロック中の ATMI コールがタイムアウトになる前にブロックする最低時間長を示す、TA_SCANUNIT 属性の乗算子。SET 操作でこの属性値としてゼロが渡されると、属性値はデフォルト値にリセットされます。

TA_DBBLWAIT:0 <= *num* < 32,768
: システムの基本的な正常チェックの周期を示す、TA_SCANUNIT 属性の乗算子。正常チェックには、ローカル・マシンで実行しているそれぞれのクライアント / サーバに対して BBL が実行するクライアント / サーバの妥当性チェックと BBL 状態チェック・イン（MP モードのみ）があります。SET 操作でこの属性値としてゼロが渡されると、属性値はデフォルト値にリセットされます。

TA_SANITYSCAN:0 <= *num* < 32,768
: 基本的な正常チェックの間隔を示す TA_SCANUNIT 属性の乗算子。正常チェックには、ローカル・マシン上で動作しているクライアント / サーバ用の各 BBL が行う、クライアント / サーバのバイアビリティ・チェックのほかに、BBL の状態チェック・インなどがあります (MP モードの場合のみ)。SET 操作でこの属性値としてゼロが渡されると、属性値はデフォルト値にリセットされます。

TA_CURDRT:0 <= *num* < 32,768
: 使用中の掲示板のルーティング・テーブルの現在の項目数を示します。

TA_CURGROUPS:0 <= *num* < 32,768
: 使用中の掲示板のサーバ・グループ・テーブルの現在の項目数。

TA_CURMACHINES:0 <= *num* < 32,768
: 現在環境設定が済んでいるマシンの数。

TA_CURQUEUES:0 <= *num* < 32,768
使用中の掲示板のキュー・テーブルの現在の項目数。

TA_CURRFT:0 <= *num* < 32,768
使用中の掲示板のルーティング基準の範囲テーブルの現在の項目数。

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
使用中の掲示板のサーバ・テーブルの最大項目数。

TA_CURSERVICES:0 <= *num* < 32,768
使用中の掲示板のサービス・テーブルの現在の項目数。

TA_CURSTYPE:0 <= *num* < 32,768
使用中の掲示板のサブ・タイプ・テーブルの現在の項目数。

TA_CURTYPE:0 <= *num* < 32,768
使用中の掲示板のタイプ・テーブルの現在の項目数。

TA_HWDRT:0 <= *num* < 32,768
使用中の掲示板のルーティング・テーブルの最大項目数。

TA_HWGROUPS:0 <= *num* < 32,768
使用中の掲示板のサーバ・グループ・テーブルの最大項目数。

TA_HWMACHINES:0 <= *num* < 32,768
環境設定済みマシンの最大数。

TA_HWQUEUES:0 <= *num* < 32,768
使用中の掲示板のキュー・テーブルの最大項目数。

TA_HWRFT:0 <= *num* < 32,768
使用中の掲示板のルーティング基準の範囲テーブルの最大項目数。

TA_HWRTDATA:0 <= *num* < 32,768
ルーティング・テーブルの文字列プールの最大サイズ。

TA_HWSERVERS:0 <= *num* < 32,768
使用中の掲示板のサーバ・テーブルの最大項目数。

TA_HWSERVICES:0 <= *num* < 32,768
使用中の掲示板のサービス・テーブルの最大項目数。

TA_SEC_PRINCIPAL_NAME:*string*[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、セキュリティのプリンシパル名を識別する文字列。この属性には、最後のヌル文字を除き、最大 511 文字まで指定できます。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。

TA_SEC_PRINCIPAL_NAME は、コンフィギュレーション階層の 4 つのレベル T_DOMAIN クラス、T_MACHINE クラス、T_GROUP クラス、および T_SERVER クラス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 TA_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は、デフォルトでこのドメインの TA_DOMAINID 文字列に設定されます。

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 以降を実行するシステム・プロセスに対して復号化キーを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 だけが指定されている場合には、これ以外の 2 つの属性はそれぞれ、長さゼロの NULL 文字列に設定されます。

TA_SEC_PRINCIPAL_LOCATION:*string*[0..511]

TA_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。この属性には、最後のヌル文字を除き、最大 511 文字まで指定できます。

TA_SEC_PRINCIPAL_LOCATION は、コンフィギュレーション階層の 4 つのレベル T_DOMAIN クラス、T_MACHINE クラス、T_GROUP クラス、および T_SERVER クラスのいずれでも指定できます。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 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。この属性には、最後のヌル文字を除き、最大 511 文字まで指定できます。

TA_SEC_PRINCIPAL_PASSVAR は、コンフィギュレーション階層の 4 つのレベル T_DOMAIN クラス、T_MACHINE クラス、T_GROUP クラス、および T_SERVER クラスのいずれでも指定できます。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)。

初期化処理中、管理者は、TA_SEC_PRINCIPAL_PASSVAR で設定された、各復号化キーのそれぞれのパスワードを入力する必要があります。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。

TA_SIGNATURE_AHEAD: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の範囲 (秒)。指定されていない場合、3600 秒 (1 時間) がデフォルト値になります。この属性は、BEA Tuxedo 7.1 以降のソフトウェアが稼動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

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の範囲 (秒)。指定されていない場合、604800 秒 (1 週間) がデフォルト値になります。この属性は、BEA Tuxedo 7.1 以降のソフトウェアが稼動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。指定されていない場合、N がデフォルト値になります。この属性は、BEA Tuxedo 7.1 以降のソフトウェアが稼動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階層の 4 つのレベル T_DOMAIN クラス、T_MACHINE クラス、T_GROUP クラス、および T_SERVICE クラス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。

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。指定されていない場合、N がデフォルト値になります。この属性は、BEA Tuxedo 7.1 以降のソフトウェアが稼動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階層の 4 つのレベル T_DOMAIN クラス、T_MACHINE クラス、T_GROUP クラス、および T_SERVICE クラスのいずれでも指定できます。特定のレベルで TA_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。

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アクティブでない状態のときにしか調節できません。これは、ATMI インターフェイス・ルーチンを利用してアプリケーションを管理することは不可能なことを意味します。そのため、起動されていないアプリケーションの環境を設定または再設定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、そのアプリケーションに対してマスタ・サイトとして設定されたサイトのみで、アクティブでないアプリケーションのコンフィギュレーション（SET 操作）のためだけに使用することができます。いったんコンフィギュレーションを作成し、アクティブな状態にすると、MIB(5) に記載した標準の ATMI インターフェイスを利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 で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することができます。スコープはグローバルです。

属性表

T_FACTORY **の属性**

属性	使用	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}"	NA
TA_FACTORYID	k	string	R--R--R--	string [1...256]	NA
TA_INTERFACENAME	k	string	R--R--R--	string [1...128]	NA

(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 オブジェクトの構成情報と実行時情報を検索します。

下に示した状態は、GET 要求に対する応答で返される TA_STATE の意味を示します。

ACTive	T_FACTORY オブジェクトが FactoryFinder によって登録されています。

TA_FACTORY
ファクトリに対して登録されている ID。

TA_INTERFACENAME
ファクトリに対する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・リポジトリの 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・グループに関連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、グループの識別、サイト、DTP に関する情報を表します。

属性表

TM_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)	string	rwyr--r--	"*LMID1*[,*LMID2*]"	N/A
TA_STATE(k)	string	rwxr-xr--	GET:"{ACT | INA | MIG}" SET:"{NEW | INV | ACT | RAC |INA | MIG}"	N/A N/A
TA_CURLMID(k)	string	R--R--R--	*LMID*	N/A
TA_ENVFILE	string	rwyr--r--	*string*[0..78]	""
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..78]	""

TM_MIB(5): T_GROUP **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_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

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なければなりません。サーバ・グループ名にはアスタリスク (*)、カンマ (,)、コロン (:) を入れることはできません。

TA_GRPNO:1 <= *num* < 30,000

このサーバ・グループのグループ番号。

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・グループを移行できる先を示します （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 1 つの LMID だけを指定した場合は、プライマリ、セカンダリどちらの LMID とも一致します。アクティブなグループの場所は、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の識別子は、すでに環境設定が済んでいなければなりません。

制限事項 : アクティブなオブジェクトでこの属性を変更する場合、グループのバックアップ LMID の指定だけを変更できます。

TA_STATE:

GET:{ACTive | INActive | MIGrating}

GET 操作は、選択された 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検索します。下に示した状態は、GET 要求に対する応答で返される TA_STATE の意味を示します。

ACTive	定義され、アクティブな状態にある T_GROUP オブジェクト （TMS あるいはアプリケーション・サーバ)。TA_TMSNAME 属性がヌル以外の文字列に設定されたサーバ・グループは、そのグループの TMS がアクティブであればグループ自体もアクティブと見なされます。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	定義され、アクティブでない状態にある T_GROUP オブジェクト。
MIGrating	定義され、現在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際しては ACTive と同等です。

SET:{NEW | INValid | ACTive | ReACtivate | INActive | MIGrating}

SET 操作は、選択された T_GROUP オブジェクトのコンフィギュレーションおよび実行時属性を検索します。次に、SET 要求で TA_STATE に設定できる値の意味を説明します。記述されていない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_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 または 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変わりません。
INValid	D 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な状態に切り替えます。状態の変更は、INActive または MIGrating の状態にある場合にのみ許可されます。この状態の推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が考慮されます（--x--x--x）。 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状態にある場合は、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上で実行され、それ以外の場合は、TMS とアプリケーション・サーバはセカンダリ論理マシン上で実行されます（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は、要求は実行されません。 グループの現在の状態が MIGrating のときには、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 TMS とアプリケーション・サーバの実行マシンとして使用されます。それ以外の場合は、要求は実行されません。サーバ・グループをアクティブな状態にする際に、個々のサーバの状態が必要な場合には、TA_FLAG の TMIB_NOTIFY TA_FLAG 値を使用する必要があります。 正常終了すると、オブジェクトの状態は ACTive になります。
ReACtivate	状態が INActive または MIGrating である場合だけでなく ACTive である場合にもこの状態の変更が許可される点を除けば、ACTive 状態への切り替えとまったく同じです。 サーバ・グループを再度アクティブな状態に戻す際に、個々のサーバの状態が必要な場合には、TA_FLAG の TMIB_NOTIFY TA_FLAG 値を使用する必要があります。

INActive	T_GROUP オブジェクトをアクティブでない状態に切り替えます。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）は、アクティブでない状態に切り替えられます。状態の変更は、ACTive または MIGrating の状態にある場合にのみ許可されます。正常終了すると、オブジェクトの状態は INActive になります。 サーバ・グループを非活性化する際に、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 オブジェクトをアクティブでない状態に切り替え、グループがセカンダリの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。正常終了の場合は、オブジェクトの状態は MIGrating になります。
UnAVailable	グループの全てのアプリケーション・サービスを中断します。(注 : 各アプリケーション・サービスは T_SVCGROUP クラスを通じて中断させることができます。) この状態の値に対する SET 操作は、グループが ACTive な状態の場合にのみ認められます。この操作はグループを ACTive の状態にしますが、全てのアプリケーションの状態は中断されたままの状態です。 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	グループの中にある、中断と記されている全てのアプリケーションを中断ではなくします。この状態の値に対する SET 操作は、グループが ACTive な状態の場合にのみ認められます。このオペレーションはグループを ACTive な状態のままにします。

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、アクティブでないサーバ・グループに対しては返されません。

TA_ENVFILE:*string*[0..78]

このグループ内で実行するサーバの環境設定ファイル。正しくないファイル名を指定すると環境変数に追加されません。*string* の値は環境に置かれます。

起動時、ローカル・サーバは tmboot(1) の環境を継承し、MASTER 上にないリモート・サーバは tlisten(1) の環境を継承します。また、TUXCONFIG、TUXDIR、および APPDIR も、対応する T_GROUP オブジェクトの情報に基づいて環境に置かれます。

PATH は環境で次のように設定されます。

APPDIR:TUXDIR/bin:/bin:/usr/bin:<*path*>

<*path*> は、マシンの環境設定ファイルの最初の 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、サーバの検索パスに使用され、単純なパス名または相対パス名で指定されます。つまり、先頭はスラッシュではありません。

LD_LIBRARY_PATH は環境で次のように設定されます。

APPDIR:TUXDIR/lib:/lib:/usr/lib:<*lib*>

上記の <*lib*> は、マシンの環境ファイルで最初に現れる LD_LIBRARY_PATH= という行 （ただし、このような行が含まれている場合）の値です （それ以降に現れる LD_LIBRARY_PATH= の行は無視されます）。

サーバの初期化時 （tpsvrinit(3c) を呼び出す前）には、サーバは初期化作業の一環としてマシンとサーバの両方の ENVFILE ファイルを読み取り、値をエクスポートします。値がマシンとサーバの両方の ENVFILE ファイルにセットされている場合は、PATH が追加されることを除いて、サーバの ENVFILE ファイルの値がマシンの ENVFILE ファイルの値に優先します。クライアントはマシンの ENVFILE ファイルのみ使用します。マシンとサーバの ENVFILE ファイルを処理する際、<*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下線またはアルファベットだけです。

PATH 行が使用されている場合、PATH は次のように設定されます。

APPDIR:TUXDIR/bin:/bin:/usr/bin:<*path*>

<*path*> は、環境設定ファイルの最初の PATH= 行の値です （これ以降の PATH= 行はすべて無視される）。マシンとサーバの両方の環境設定ファイルに 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= 行が使用されている場合、LD_LIBRARY_PATH は次のように設定されます。

APPDIR:TUXDIR/lib:/lib:/usr/lib:<*lib*>

上記の <*lib*> は、環境設定ファイルで最初に設定されている行はすべて無視される)。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の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影響はありません。

TA_OPENINFO:*string*[0..256]

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・マネージャのインスタンスに依存する情報を示します。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。

TA_TMSNAME 属性の値に TMS 以外のヌルでない文字列が指定されている場合、TA_OPENINFO 属性の値は、リソース・マネージャへの接続を開始する際に必要なリソース・マネージャ固有の情報を示します。それ以外の場合、TA_OPENINFO 属性の値は無視されます。

TA_OPENINFO 属性の値がヌル文字列である場合、このグループに対するリソース・マネージャは、リソースへの接続を開始する際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、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。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 上 #
OPENINFO = "TUXEDO/QM:*qmconfig*:*qspace*"

# Windows 上 #
OPENINFO = "TUXEDO/QM:*qmconfig*;*qspace*"

# AS/400 環境 #
OPENINFO = "TUXEDO/QM:*qmconfig*;*qspace*"

# OpenVMS 環境 #
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・スペースが位置する 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、TA_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。

制限事項 : 実行時にこの属性を変更しても、グループのアクティブなサーバには影響はありません。

TA_CLOSEINFO:*string*[0..256]

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・マネージャの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 文字列は使用されません。

TA_TMSNAME 属性の値に TMS 以外のヌルでない文字列が指定されている場合、TA_CLOSEINFO 属性の値は、リソース・マネージャへの接続を終了する際に必要なリソース・マネージャ固有の情報を示します。それ以外の場合には、TA_CLOSEINFO 属性の値は無視されます。

TA_CLOSEINFO 属性の値がヌル文字列である場合、このグループのリソース・マネージャはリソースへの接続を終了する際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、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。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 : 実行時にこの属性を変更しても、グループのアクティブなサーバには影響はありません。

TA_TMSCOUNT:0 または 2 <= *num* < 11

TA_TMSNAME 属性の値にヌル以外の文字列が指定されている場合、この属性の値は、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・サーバの数を示します。それ以外の場合は、この属性の値は無視されます。

TA_TMSNAME:*string*[0..78]

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。tpbegin() で開始されて tpcommit()/tpabort() で終了する、複数のリソース・マネージャや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、つまり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。

値 "TMS" は、ヌル 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" 以外のブランク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論理マシンとセカンダリ論理マシンに対して TLOGDEVICE を指定する必要があります。

一意のサーバ識別子が各 TM サーバに対して自動的に選択され、サーバは何回でも再起動することができます。

TA_SEC_PRINCIPAL_NAME:*string*[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、セキュリティのプリンシパル名を識別する文字列。この属性には、最後のヌル文字を除き、最大 511 文字まで指定できます。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階層の 4 つのレベル T_DOMAIN クラス、T_MACHINE クラス、T_GROUP クラス、および T_SERVER クラス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 TA_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は、デフォルトでこのドメインの TA_DOMAINID 文字列に設定されます。

TA_SEC_PRINCIPAL_NAME のほかにも、TA_SEC_PRINCIPAL_LOCATION と TA_SEC_PRINCIPAL_PASSVAR という属性があることに注意してください。後の 2 つの属性は、アプリケーション起動時に、BEA Tuxedo 7.1 以降を実行するシステム・プロセスに対して復号化キーをオープンする処理で指定します。特定のレベルで TA_SEC_PRINCIPAL_NAME だけが指定されている場合には、これ以外の 2 つの属性はそれぞれ、長さゼロの NULL 文字列に設定されます。

TA_SEC_PRINCIPAL_LOCATION:*string*[0..511]

TA_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。この属性には、最後のヌル文字を除き、最大 511 文字まで指定できます。

TA_SEC_PRINCIPAL_LOCATION は、コンフィギュレーション階層の 4 つのレベル T_DOMAIN クラス、T_MACHINE クラス、T_GROUP クラス、および T_SERVER クラスのいずれでも指定できます。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 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。この属性には、最後のヌル文字を除き、最大 511 文字まで指定できます。

TA_SEC_PRINCIPAL_PASSVAR は、コンフィギュレーション階層の 4 つのレベル T_DOMAIN クラス、T_MACHINE クラス、T_GROUP クラス、および T_SERVER クラスのいずれでも指定できます。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)。

初期化処理中、管理者は、TA_SEC_PRINCIPAL_PASSVAR で設定された、復号化キーそれぞれのパスワードを入力する必要があります。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。

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。指定されていない場合、N がデフォルト値になります。この属性は、BEA Tuxedo 7.1 以降のソフトウェアが稼動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階層の 4 つのレベル T_DOMAIN クラス、T_MACHINE クラス、T_GROUP クラス、および T_SERVICE クラス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。

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。指定されていない場合、N がデフォルト値になります。この属性は、BEA Tuxedo 7.1 以降のソフトウェアが稼動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階層の 4 つのレベル T_DOMAIN クラス、T_MACHINE クラス、T_GROUP クラス、および T_SERVICE クラスのいずれでも指定できます。特定のレベルで TA_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。

制限事項　なし

T_IFQUEUE クラス

概要　T_IFQUEUE MIB クラスは、CORBA 環境における特定のサーバ・キュー (T_QUEUE) に関係し、インターフェイスの実行時属性を表します。このクラスは本質的に読み取り専用であり、インターフェイスが継承する環境設定属性へのアクセス、および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。さらに、このクラスを利用して、管理者はインターフェイスをきめ細かく中断したりアクティブ化することができます。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の呼び出しを処理できるサーバ・プロセスと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

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
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

T_IFQUEUE クラス定義の属性表

(k) - GET キー・フィールド
(l) - ローカル・フィールド
(*) - GET または SET キー。SET 操作で 1 つ以上が必要

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。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 で指定し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 (:) を入れることはできません。

TA_RQADDR: *string*[1..30]
このインターフェイスを提示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、選択された T_IFQUEUE オブジェクトの環境設定情報を検索します。下に示した状態は、GET 要求に対する応答で返される TA_STATE の意味を示します。記述されていない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現在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現在分断されているインターフェイスを表します。

SET: { ACTive | SUSpended}
次に、SET 要求で TA_STATE に設定できる値の意味を説明します。記述されていない状態を設定することはできません。

ACTive	T_IFQUEUE オブジェクトをアクティブ化します。状態の変更は、SUSpended 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 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 (つまり、INActive または INValid から ACTive への状態変更) はサポートされておらず、また非宣言 (つまり、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指定します。この属性に関する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。ロード・バランシングを行うために、インターフェイス・ロードが使用されます。すなわち、追加されている作業ロードが高いキューは新規要求に対して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るようになります。複数のインターフェイス要求がサービス・キューで待ち状態の場合、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の呼び出しを処理するときの時間制限 (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、要求処理時に指定時間制限値を超えると、異常終了されます。この属性を 0 に設定すると、サーバは異常終了されません。

TA_TRANTIME: 0 <= num
: このT_INTERFACEオブジェクトに対して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 (秒)。トランザクション・モード以外の要求が受信され、インターフェイスのT_INTERFACE:TA_AUTOTRAN属性値が "Y" のとき、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示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示する、対応するサーバの数。

TA_TPPOLICY: { method | transaction | process }
: TP フレームワークの非活性化方針。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
: インターフェイスが最初に提示されてから完了したインターフェイスのメソッドの呼び出し数。

TA_NQUEUED: 0 <= num
: 現在このインターフェイスへのキューに登録されている要求の数。

TA_CUROBJECTS: 0 <= num
: 関連するキューに対する、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、関連するマシン上の、このキューに対す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、メモリ内になく、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
: 対応するキューに対し、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、CORBA インターフェイスのドメイン・グループ・レベルおよびサーバ・グループ・レベルでの環境設定および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・グループと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、"") が含まれます。

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・グループ名が含まれます。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のためのコンテナも提供します。

サーバ内でアクティブ化されている各 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、その属性は、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ければ、オブジェクトが動的に作成されます。この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環境設定属性の指定は、すべてオプションであり、システム定義のデフォルトが設定され、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・スケルトンの生成に使用される ICF ファイル経由で識別され、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

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..128]	N/A
TA_SRVGRP	r*	string	ru-r--r--	string[0..30]	N/A

T_INTERFACE クラス定義の属性表

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"[a]
TA_LOAD		long	rwxr-xr--	1 <= num < 32K	50[a]
TA_PRIO		long	rwxr-xr--	1<= num < 101	50[a]
TA_TIMEOUT		long	rwxr-xr--	0 <= num	0[a]
TA_TRANTIME		long	rwxr-xr--	0 <= num	30[a]
TA_FBROUTINGNAME		string	rwyr-yr--	string[1...15]	[b]
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[c]
TA_NQUEUED	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
(r) - オブジェクトを作成するとき必要 (SET TA_STATE NEW)
(*) - GET または SET キー。SET 操作で 1 つ以上が必要。

a. 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。
b. 同じ TA_INTERFACENAME を持つ T_INTERFACE オブジェクトはすべて、TA_FBROUTINGNAME 値が一致していなければなりません。したがって、現在 TA_INTERFACENAME が同じ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が 0 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 (および唯一の有効値) は、既存の一致するオブジェクトに対して現在設定されている TA_FBROUTINGNAME 値になります。
c. 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 に指定されている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 (:) を入れることはできません。この属性に対して明示的に指定された長さ 0 の文字列を使用して、インターフェイスのドメイン・レベルの環境設定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セマンティクス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 値のセマンティクスです。グループおよびドメイン・レベルのオブジェクト間でセマンティクスが異なる場合は、その違いを説明しています。

GET: {ACTive | INActive | SUSpended | PARtitioned}
GET<:/cs> 操作は、選択した T_INTERFACE オブジェクトに対する環境設定情報を検索します。下に示した状態は、GET 要求に対する応答で返される TA_STATE の意味を示します。ここに記述されていない状態は返りません。

ACTive	T_INTERFACE オブジェクトが定義され、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 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、ACTive と同等の状態にある、対応する T_IFQUEUE エントリはありません。
SUSpended	T_INTERFACE オブジェクトが定義され、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際しては ACTive と同等です。
PARtitioned	T_INTERFACE が定義され、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際しては ACTive と同等です。

SET: {NEW | INValid | ACTive | REActivate | SUSpended}
SET 操作は、選択した T_INTERFACE オブジェクトに対する環境設定と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、複数のサーバが現在インターフェイスを提供している場合は、実行時に変更すると複数のサーバに影響することがあります。次に、SET 要求で TA_STATE に設定できる値の意味を説明します。記述されていない状態を設定することはできません。

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があります。その他の環境設定属性の設定は、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化します。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、ドメイン内で現在 SUSpended 状態にある、対応するすべての T_IFQUEUE エントリがアクティブ化されます。グループ・レベルのオブジェクトにこの状態を設定すると、インターフェイスを提供しているグループ内のサーバのみが影響を受けます。状態の変更は、SUSpended 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 ACTive になります。
REActivate	T_INTERFACE オブジェクトを再びアクティブ化します。ドメイン・レベルのオブジェクトにこの状態を設定すると、ドメイン内で現在 SUSpended 状態にある、すべての対応する T_IFQUEUE エントリがアクティブになります。グループ・レベルのオブジェクトにこの状態を設定すると、インターフェイスを提供しているグループ内のサーバのみが影響を受けます。状態の変更は、ACTive または SUSpended 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 ACTive になります。この状態では、各グループ・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することができます。

SUSpended	T_INTERFACE オブジェクトを中断します。ドメイン・レベルのオブジェクトにこの状態を設定すると、ドメイン内で現在 ACTive 状態にある、すべての対応する T_IFQUEUE エントリが中断されます。グループ・レベルのオブジェクトにこの状態を設定すると、インターフェイスを提供しているグループ内のサーバのみが影響を受けます。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 SUSpended になります。

制限事項 : インターフェイスの動的な宣言 (つまり、INActive または INValid から ACTive への状態変更) 、および非宣言 (つまり、ACTive から INActive への状態変更) はサポートされていません。

TA_AUTOTRAN: { Y | N }

トランザクションのコンテキストの範囲外で行われた呼び出しに対して、トランザクションを自動的に開始するかどうかを指定します。
Limitations: この属性を実行時に更新しても、アクティブ同等の T_INTERFACE オブジェクトには反映されず、TA_TXPOLICY が UBBCONFIG ファイル内のこの属性に対して指定された値に上書き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、実行時には無効です。Y に設定された場合と同様に動作します。
never	値 Y は、実行時には無効です。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 は実行時には無効です。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。ロード・バランシングを行うために、インターフェイス・ロードが使用されます。すなわち、追加されている作業ロードが高いキューは新規要求に対して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影響しません。

TA_PRIO: 1 <= *num* <= 101

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るようになります。複数のインターフェイス要求がサービス・キューで待ち状態の場合、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変更しても、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影響しません。

TA_TIMEOUT: 0 <= *num*

このインターフェイスの個々のメソッドの呼び出しを処理するときの時間制限 (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、要求処理時に指定時間制限値を超えると、異常終了されます。この属性を 0 に設定すると、サーバは異常終了されません。
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影響しません。

TA_TRANTIME: 0 <= *num*

この T_INTERFACE オブジェクトに対して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 (秒)。トランザクションモード以外の要求が受信され、インターフェイスの T_INTERFACE:TA_AUTOTRAN 属性値が "Y" のとき、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影響しません。
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、その値は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、警告メッセージが表示されます。

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。名前 FBROUTINGNAME を使用して、将来、ほかのメッセージ・ベースのルーティングのルーティング基準を設定できるようにしておきます。こうしておくと、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 (つまり、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。この属性はドメイン・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 が MIB_LOCAL ビットを設定しない限り)、空白 ("") です。ローカルの場合、同じインターフェイスに対して複数のドメイン・レベルのオブジェクト (マシンごとに 1 つ) が、各オブジェクトに示されているマシンから検索されたローカル値とともに返されます。

TA_NUMSERVERS: 0 <= *num*
: このインターフェイスを提供している対応サーバの数。

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た方針を反映します。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。この属性の設定内容は、TA_AUTOTRAN 属性の効果に影響します。詳細については、TA_AUTOTRAN の説明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、サーバを構築する際に開発者が設定し、サーバの起動時に登録されます。

TA_NCOMPLETED: 0 <= *num*
: 対応する T_IFQUEUE オブジェクトに関して、最初に提供されてから完了されたインターフェイス・メソッドの呼び出し数。ドメイン・レベルのオブジェクトをローカルに照会する (TA_FLAGS MIB_LOCAL ビットを設定) と、マシンごとに 1 つのオブジェクトが、そのマシンの指定されたインターフェイスに対する統計情報とともに返されます。

TA_NQUEUED: 0 <= *num*
: 現在このインターフェイスへのキューに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に照会する (TA_FLAGS MIB_LOCAL ビットを設定) と、マシンごとに 1 つのオブジェクトが、そのマシンの指定されたインターフェイスに対する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 MIB 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することができます。

T_MACHINE クラスの定義

概要

T_MACHINE クラスは、特定のマシンに関す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..64]	N/A
TA_TUXDIR(r)	string	rw-r--r--	*string*[2..78]	N/A
TA_APPDIR(r)	string	rw-r--r--	*string*[2..8]	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..78]	""
TA_PERM	long	rwyr--r--	0001 <= *num* <= 0777	(注 2)
TA_ULOGPFX	string	rwyr--r--	*string*[0..78]	(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)
TA_MAXGTT	long	rw-r--r--	0 <= *num* < 32,768	(注 2)
TA_MAXWSCLIENTS	long	rw-r--r--	0 <= *num* < 32,768	0
TA_MAXACLCACHE	long	rw-r--r--	10 <= *num* <= 32,000	100

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_TLOGDEVICE	string	rw-r--r--	*string*[0..64]	""
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_NADDR	string	rw-r--r--	*string*[0..78]	N/A
TA_NLSADDR	string	rw-r--r--	*string*[0..78]	N/A
TA_FADDR	string	rw-r--r--	*string*[0..78]	""
TA_FRANGE	long	rw-r--r--	1 <= *num* <= 65,535	1
TA_CMPLIMIT	string	rwyr-yr--	"*remote*[,*local*]"	MAXLONG、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
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"

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
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
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

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_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 と TA_PMID は、それぞれこのクラス内で一意の値でなければなりません。SET 操作では、これらのいずれか一方のフィールドだけが使用されます。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された値と同じ値になります。
注 3 デフォルト値は、TA_APPDIR の後に /ULOG が続く文字列です。
注 4 リンク・レベルでの暗号化の値 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:*string*[1..30]
論理マシン識別子。この識別子は、TM_MIB の残りの定義中で、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されます。

TA_PMID:*string*[1..30]
物理マシンの識別子。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..64]

このマシン上での BEA Tuxedo システムのバイナリ形式のコンフィギュレーション・ファイルの場所を示す、ファイルまたはデバイスの絶対パス名。管理者は、このファイル、すなわち TA_TUXCONFIG 属性の値が指す名前のファイルを、マスタ・マシンに 1 つだけ保存する必要があります。このファイルに含まれる情報は、他の 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。環境中でのこの属性の使用方法については、下記の TA_ENVFILE の項を参照してください。

TA_TUXDIR:*string*[2..78]

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。環境中でのこの属性の使用方法については、下記の TA_ENVFILE の項を参照してください。

TA_APPDIR:*string*[2..78]

アプリケーションの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）。最初のディレクトリは、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。環境中でのこの属性の使用方法については、下記の TA_ENVFILE の項を参照してください。

TA_STATE:

GET:"{ACTive | INActive | PARtitioned}"

GET 操作は、選択された 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検索します。下に示した状態は、GET 要求に対する応答で返される TA_STATE の意味を示します。

ACTive	T_MACHINE オブジェクト （管理サーバ、すなわち DBBL、BBL、BRIDGE）が定義され、アクティブな状態にあることを示します。
INActive	T_MACHINE オブジェクトが定義され、アクティブでない状態にあることを示します。
PARtitioned	T_MACHINE オブジェクトが定義され、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るが、現在は接続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際しては ACTive と同等です。

SET:"{NEW | INValid | ACTive | ReACtivate | INActive | ForceINactive | CLEaning}"

SET 操作は、選択された 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。次に、SET 要求で TA_STATE に設定できる値の意味を説明します。記述されていない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_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 または 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な状態に切り替えます。DBBL、BBL、BRIDGE などの必要な管理サーバは指定された場所で開始され、その場所で実行されるように環境設定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）。この状態の推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が考慮されます（--x--x--x）。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 ACTive になります。 マシンをアクティブな状態に切り替え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。

ReACtivate	T_MACHINE オブジェクトをアクティブな状態に切り替えます。DBBL、BBL、BRIDGE などの必要な管理サーバは指定された場所で開始され、その場所で実行されるように環境設定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)。この状態の推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が考慮されます (--x--x--x)。状態の変更は、ACTive または INActive 状態にある場合にのみ許可されます。正常終了すると、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な状態に切り替える際に個々のサーバの状態が必要な場合には、TMIB_NOTIFY TA_FLAG の値を使用する必要があります。
INActive	T_MACHINE オブジェクトをアクティブでない状態に切り替えます。DBBL、BBL、BRIDGE などの必要な管理サーバは指定されたサイトで停止され、そのサイトで実行しているアプリケーション・サーバも停止されます（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、指定されたマシン上の他のアプリケーション・リソースがアクティブでない状態にある場合にのみ許可されます。正常終了すると、オブジェクトの状態は INActive になります。 マシンをアクティブでない状態に切り替える際に個々のサーバの状態が必要な場合には、TMIB_NOTIFY TA_FLAG の値を使用する必要があります。

ForceINactive	T_MACHINE オブジェクトを、接続されたクライアントとは無関係にアクティブでない状態にします。DBBL、BBL、BRIDGE などの必要な管理サーバは指定されたサイトで停止され、そのサイトで実行しているアプリケーション・サーバも停止されます（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。 マシンをアクティブでない状態に切り替える際に個々のサーバの状態が必要な場合には、TMIB_NOTIFY TA_FLAG の値を使用する必要があります。
CLEaning	指定されたマシンおよび指定されたマシンに関係するクリア / スキャニング動作を開始します。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、グローバル掲示板に記録されたそのマシンに関する情報は消去されます。この組合せは、アプリケーションが ACTive な状態にあり、T_MACHINE オブジェクトが ACTive または PARtitioned のいずれかの状態にある場合にのみ許可されます。分断されていないマシンに対する操作が正常終了の場合は、状態は変更されません。分断されたマシンに対する操作が正常終了の場合は、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 への状態の切り替えは、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 クラスを利用してアクティブでない状態に切り替えます。

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に対して使用される、UNIX システムのグループ識別子。tmboot(1)、tmshutdown(1)、あるいは tmadmin(1) といった管理コマンドは、このマシン上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立ち上げられます。

制限事項 : UNIX システム固有の属性です。アプリケーションが動作しているプラットフォームが UNIX ベース以外の場合、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に対して使用される、UNIX システムのグループ識別子。tmboot(1)、tmshutdown(1)、あるいは tmadmin(1) といった管理コマンドは、このマシン上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立ち上げられます。

制限事項 : UNIX システム固有の属性です。アプリケーションが動作しているプラットフォームが UNIX ベース以外の場合、この属性は返されないことがあります。

TA_ENVFILE:*string*[0..78]

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設定ファイル。正しくないファイル名を指定すると環境変数に追加されません。tuxreadenv(3c) で説明されているように、*string* の値は環境内に置かれます。

サーバをブートする際には、ローカル・サーバは tmboot(1) から環境を引き継ぎ、リモート・サーバは （MASTER マシン以外のマシンのサーバ）は tlisten(1) から環境を引き継ぎます。関連する T_MACHINE オブジェクトの情報に基づいてサーバをブートする際には、TUXCONFIG、TUXDIR、および APPDIR も環境中に取り込まれます。PATH は、環境中では次のようにセットされます。

APPDIR:TUXDIR/bin:/bin:/usr/bin:<*path*>

上記の <*path*> は、マシンの環境ファイルで最初に現れる PATH= という行（ただし、このような行が含まれている場合）の値です （それ以降に現れる PATH= の行は無視されます）。この PATH の値は、サーバの検索パスに使用され、単純なパス名または相対パス名で指定されます。つまり、先頭はスラッシュではありません。LD_LIBRARY_PATH は、環境中では次のようにセットされます。

APPDIR:TUXDIR/lib:/lib:/usr/lib:<lib>

上記の <*lib*> は、マシンの環境ファイルで最初に現れる LD_LIBRARY_PATH= という行 （ただし、このような行が含まれている場合）の値です （それ以降に現れる LD_LIBRARY_PATH= の行は無視されます）。

サーバの初期化時 (tpsvrinit() を呼び出す前)、サーバは初期化作業の 1 つとして、マシンとサーバの両方の ENVFILE ファイルを読み取って値をエクスポートします。値がマシンとサーバの両方の ENVFILE ファイルにセットされている場合は、PATH が追加されることを除いて、サーバの ENVFILE ファイルの値がマシンの ENVFILE ファイルの値に優先します。クライアントは、マシンの ENVFILE ファイルだけを処理します。マシンとサーバの両方の ENVFILE ファイルが処理される場合は、<ident>= という形をしていない行はすべて無視されます。<ident> は、アンダースコア記号 (_) またはアルファベットで始まり、アンダースコア記号またはアルファベットだけを使用できます。PATH= という行がと現れると、PATH は次のようにセットされます。

APPDIR:TUXDIR/bin:/bin:/usr/bin:<*path*>

<*path*> は、環境設定ファイルの最初の PATH= 行の値です (これ以降の PATH= 行はすべて無視される)。マシンとサーバの両方の ENVFILE で PATH が存在する場合は、<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> はサーバの環境設定ファイルから読み取ったパスです。LD_LIBRARY_PATH= という行が現れると、LD_LIBRARY_PATH は下記の値にセットされます。

APPDIR:TUXDIR/lib:/lib:/usr/lib:<*lib*>

上記の <*lib*> は、環境設定ファイルで最初に設定されている LD_LIBRARY_PATH= 行の値です (それ以降の LD_LIBRARY_PATH= 行はすべて無視される)。TUXDIR、APPDIR、あるいは TUXCONFIG を再設定しようとしても、値が T_MACHINE の属性値と一致しなければ、再設定は無視され、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の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影響はありません。

TA_PERM:0001 <= *num* <= 0777

このマシン上に作成される共有メモリの掲示板に対する UNIX システムのパーミッション。システムおよびアプリケーション・メッセージのキューに対する UNIX システムのパーミッションのデフォルト設定。

制限事項 : アクティブなオブジェクトの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影響はありません。

UNIX システム固有の属性です。アプリケーションが動作しているプラットフォームが UNIX ベース以外の場合、この属性は返されないことがあります。

TA_ULOGPFX:*string*[0..78]

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 属性の値に文字列 *.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の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影響はありません。

TA_TYPE:*string*[0..15]

マシンのタイプ。マシンを類似のデータ表現を持つクラスにグループ分けするために使用されます。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、どのような文字列の値でも使用でき、その値は比較目的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、あるいはコンパイラによって異なる構造体の表現が生成される場合は、別の TA_TYPE 属性を使用する必要があります。この属性のデフォルト設定は長さがゼロの文字列で、TA_TYPE 属性の値に長さがゼロの文字列を持つ他のすべてのマシンに相当します。

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。指定されていない場合、T_DOMAIN クラスで指定した TA_MAXACCESSERS 値がデフォルト値になります。

BBL、restartsrv、cleanupsrv、tmshutdown()、および 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のサーバ・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てください (システム提供のサーバの例としては、AUTHSVR、TMQUEUE、TMQFORWARD、TMUSREVT、TMSYSEVT、TMS があります。T_GROUP TA_TMSNAME 属性、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、WSL と起動される可能性のあるワークステーション・ハンドラ (WSH) の数の両方が必要です。

BEA Tuxedo リリース 7.1 より前、つまりリリース 6.5 以前では、アプリケーションの TA_MAXACCESSERS および TA_MAXSERVERS (T_DOMAIN TA_MAXSERVERS を参照) 属性は、ユーザ・ライセンス数をチェックする仕組みと関連付けられていました。特に、あるマシンの TA_MAXACCESSERS とアプリケーションで既に動作しているマシンの TA_MAXACCESSERS の合計が、アプリケーションの TA_MAXSERVERS とユーザ・ライセンス数の合計より大きい場合には、その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 TA_MAXACCESSERS の合計数は、アプリケーションの TA_MAXSERVERS とユーザ・ライセンス数の合計数以下で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 点だけです。すべてのユーザ・ライセンスが使用中の場合、新しいクライアントをアプリケーションに結合することはできません。

TA_MAXCONV:0 <= *num* < 32,768
: マシン上のクライアントとサーバ間で同時に行える会話の最大件数。指定されていない場合、T_DOMAIN クラスで指定した TA_MAXCONV 値がデフォルト値になります。1 サーバ当たり同時に行える会話の最大数は、64 です。

TA_MAXGTT:0 <= *num* < 32,768
: このマシンが関与できるグローバルなトランザクションの同時に作成できる最大件数。指定されていない場合、T_DOMAIN クラスで指定した値がデフォルト値になります。

TA_MAXWSCLIENTS:0 <= *num* < 32,768
: (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。TA_MAXWSCLIENTS が指定されていない場合、0 がデフォルト値になります。

  ここで指定する値は、TA_MAXACCESSERS 属性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、TA_MAXWSCLIENTS に対して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によって使用することはできません。この数字を TA_MAXACCESSERS より大きな値にセットした場合は、エラーが発生します。

  TA_MAXWSCLIENTS 属性が使用されるのは、BEA Tuxedo System Workstation 機能を使用する場合だけです。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 提供の代理プロセス、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、この属性を適切な値に設定することで IPC 資源を節約できるようになります。

TA_MAXACLCACHE:10 <= *num* <= 32,000
: TA_SECURITY が ACL または MANDATORY_ACL にセットされている場合に ACL 項目数として適用される、キャッシュのエントリ項目の数。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。

TA_TLOGDEVICE:*string*[0..64]
: このマシンの DTP トランザクション・ログを記憶するための 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 システムのファイル。DTP トランザクション・ログは、デバイスに BEA Tuxedo システム VTOC テーブルとして記憶されます。このデバイスあるいはファイルは、このマシンの TA_TUXCONFIG 属性に対して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。TA_TLOGDEVICE（TA_TLOGDEVICE で指定したデバイス。以下同様）に複数の DTP トランザクション・ログが存在する場合は、それぞれ独自の名前を割り当てる必要があります。TA_TLOGNAME は、DTP トランザクション・ログ・テーブルを作成する TA_TLOGDEVICE に存在する他のどのテーブルとも異なる名前でなければなりません。

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（単位はページ）。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 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の名前。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・バイナリを通じて結合する際には、この名前が必要になります。この属性は、ソケット・ベースの BEA Tuxedo システム・バイナリに対しては必要ありません。

TA_NADDR:*string*[0..78]
: 論理マシン上の BRIDGE プロセスが自分のリスニング・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 のリスニング・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。論理マシンがネットワーク化されたアプリケーションに結合する場合、すなわち、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。

文字列の形式が "0xhex-digits" または "\\xhex-digits" の場合、偶数の有効 16 進数桁を含める必要があります。これらの形式は、指定した文字列の 16 進数表現を含む文字配列に内部で変換されます。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..78]
: この論理マシンによって示された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、上記の 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化されたアプリケーションに結合する場合、すなわち、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..78]
: アウトバウンド接続を確立する前に tmboot、BRIDGE、BSBRIDGE、tmloadcf などのローカル・プロセスが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 アドレスを指定してください。この属性と TA_FRANGE 属性は、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。このパラメータが 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・システムによってバインド先のローカル・ポートが任意に選択されます。

  string の形式が "0xhex-digits" の場合は、有効な偶数桁の 16 進数でなければなりません。これらの形式は、指定した文字列の 16 進数表現を含む文字配列に内部で変換されます。

  TCP/IP アドレスには、以下のいずれかの形式が使用されます。

  - "*//hostname*:*port*"
  - "*//#.#.#.#*:*port*"

TA_FRANGE:1<= *num* <= 65,535
: アウトバウンド接続を確立する前にローカル・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。TA_FADDR 属性は、範囲のベース・アドレスを指定します。

TA_CMPLIMIT:"*remote*[,*local*]"

リモートのトラフィックおよびオプションのローカルのトラフィックの圧縮が発生するスレッショルド・メッセージ・サイズ。*remote* と *local* には、負の数でない数値、または "MAXLONG" という文字列をセットできます。"MAXLONG" は、マシンに設定された最大の long 値に動的に変換されます。*remote* だけをセットした場合は、*local* はデフォルト設定の MAXLONG になります。

制限事項 : BEA Tuxedo リリース 4.2.2 以前のバージョンを実行しているアクティブなサイト（サイト）では、この属性は T_MACHINE オブジェクトの一部ではなくなります。ただし、そのサイトのバージョンが実行時までわからないため、アクティブでないオブジェクトに対してはこの属性をセットし、使用することができます。BEA Tuxedo リリース 4.2.2 以前のバージョンを使用しているサイトがアクティブな状態になると、設定された値は使用されなくなります。

TA_TMNETLOAD:0 <= *num* < 32,768

このマシンの負荷の調節時に評価されるリモート・サービスに追加されるサービス負荷。

制限事項 : BEA Tuxedo Release 4.2.2 以前のバージョンを実行しているアクティブなサイトでは、この属性は T_MACHINE オブジェクトの一部ではなくなります。ただし、そのサイトのバージョンが実行時までわからないため、アクティブでないオブジェクトに対してはこの属性をセットし、使用することができます。BEA Tuxedo リリース 4.2.2 以前のバージョンを使用しているサイトがアクティブな状態になると、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 pre-ticket なユーザーレベルのセマフォに対するアクセスについて使用されるスピン・カウント。デフォルト設定は、それぞれのマシンで BEA Tuxedo システム のバイナリ中に作成されます。これらのデフォルト設定は、この属性を使用して、調節のために実行時に変更することができます。スピン・カウントは、この属性値をゼロにセットすることにより、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にシステムが使用する TMSPINCOUNT 環境変数もあります。

制限事項 : BEA Tuxedo リリース 4.2.2 以前のバージョンを実行しているアクティブなサイトでは、この属性は T_MACHINE オブジェクトの一部ではなくなります。ただし、そのサイトのバージョンが実行時までわからないため、アクティブでないオブジェクトに対してはこの属性をセットし、使用することができます。BEA Tuxedo リリース 4.2.2 以前のバージョンを使用しているサイトがアクティブな状態になると、設定された値は使用されなくなります。

TA_ROLE:"{MASTER | BACKUP | OTHER}"
: アプリケーションでのこのマシンの役割。MASTER は、このマシンがマスタ・マシンとして使用されることを示し、BACKUP はこのマシンがバックアップ用のマスタ・マシンとして使用されることを示します。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 システムのプロトコルのマイナー・バージョン番号。

TA_RELEASE:1 <= *num*
: このマシンの BEA Tuxedo システムのプロトコルのメジャー・バージョン番号。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ます。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。この最低の暗号化レベルを満たすことができない場合、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・レベルでの暗号化の値 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。

  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・リンクには影響はありません。

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ます。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。デフォルト値は "128" です。

  注記 リンク・レベルでの暗号化の値 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。

  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・リンクには影響はありません。

TA_MAXPENDINGBYTES:100000 <= *num* <= MAXLONG
: メッセージのために割り当てられ得るスペースの量の限界を指定します。そのメッセージは BRIDGE プロセスによって送信されるのを待っています。

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・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、セキュリティのプリンシパル名を識別する文字列。この属性には、最後のヌル文字を除き、最大 511 文字まで指定できます。この属性に指定するプリンシパル名は、このマシン上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階層の 4 つのレベル T_DOMAIN クラス、T_MACHINE クラス、T_GROUP クラス、および T_SERVER クラス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 TA_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は、デフォルトでこのドメインの TA_DOMAINID 文字列に設定されます。

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 以降を実行するシステム・プロセスに対して復号化キーをオープンする処理で指定します。特定のレベルで TA_SEC_PRINCIPAL_NAME だけが指定されている場合、これ以外の 2 つの属性はそれぞれ、長さゼロの NULL 文字列に設定されます。

TA_SEC_PRINCIPAL_LOCATION:*string*[0..511]

TA_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。この属性には、最後のヌル文字を除き、最大 511 文字まで指定できます。

TA_SEC_PRINCIPAL_LOCATION は、コンフィギュレーション階層の 4 つのレベル T_DOMAIN クラス、T_MACHINE クラス、T_GROUP クラス、および T_SERVER クラスのいずれでも指定できます。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 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。この属性には、最後のヌル文字を除き、最大 511 文字まで指定できます。

TA_SEC_PRINCIPAL_PASSVAR は、コンフィギュレーション階層の 4 つのレベル T_DOMAIN クラス、T_MACHINE クラス、T_GROUP クラス、および T_SERVER クラスのいずれでも指定できます。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)。

初期化処理中、管理者は、TA_SEC_PRINCIPAL_PASSVAR で設定された、各復号化キーのそれぞれのパスワードを入力する必要があります。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。

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。指定されていない場合、N がデフォルト値になります。この属性は、BEA Tuxedo 7.1 以降のソフトウェアが稼動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階層の 4 つのレベル T_DOMAIN クラス、T_MACHINE クラス、T_GROUP クラス、および T_SERVICE クラス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。

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。指定されていない場合、N がデフォルト値になります。この属性は、BEA Tuxedo 7.1 以降のソフトウェアが稼動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階層の 4 つのレベル T_DOMAIN クラス、T_MACHINE クラス、T_GROUP クラス、および T_SERVICE クラスのいずれでも指定できます。特定のレベルで TA_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。

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・ハンドラを経由して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

このマシンに現在ログインしている、ネイティブ・クライアントと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件数。

TA_CURGTT:0 <= *num* < 32,768
: このマシンの使用中のトランザクション・テーブルの項目数。

TA_CURRLOAD:0 <= *num*
: 現在このマシンでキューに登録されているサービス負荷。

  制限事項 : T_DOMAIN:TA_LDBAL 属性が "N" にセットされているか、または T_DOMAIN:TA_MODEL 属性が "MP" にセットされている場合は、FML32 NULL 値が返されます。

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・ハンドラを通じてアクセスするクライアントおよびサーバの最大数。

TA_HWCLIENTS:0 <= *num* < 32,768
: このマシンにログインするネイティブ・クライアントと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件数。

TA_HWGTT:0 <= *num* < 32,768
: このマシンの使用中のトランザクション・テーブルの項目の最大数。

TA_HWWSCLIENTS: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() 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]
: マシンのバイナリのソフトウェア・バージョン。バイナリが BEA Tuxedo システムのマスタ・バイナリでない場合は、長さがゼロの文字列。

TA_HWACLCACHE:0 <= *num*
: ACL キャッシュで使用する項目の最大数。

TA_ACLCACHEHITS:0 <= *num*
: 「ヒット」した （すなわちすでにキャッシュに存在した）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*
: アクセス制御違反と判明した 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、正しく処理したサービス負荷の合計。長時間実行中のアプリケーションでは、この値は一巡していることがあるので注意してください。つまり、この属性はロングの最大値を超えると再度ゼロからスタートします。

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総数。長時間実行中のアプリケーションでは、この値は一巡していることがあるので注意してください。つまり、この属性はロングの最大値を超えると再度ゼロからスタートします。

制限事項

SHM モード （T_DOMAIN:TA_MODEL 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_OPTIONS 参照）をセットした MP モード（T_DOMAIN:TA_MODEL 参照）のアプリケーションは、T_DOMAIN:TA_MAXMACHINES 属性で指定された最大数の設定可能な T_MACHINE オブジェクトを持つことができます。このクラスの属性の多くは、サイトでアプリケーションがアクティブでない状態のときにしか調整できません。最低限に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ので、アプリケーションを管理するための ATMI インターフェイス・ルーチンはマスタ・マシン・オブジェクトに対して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の環境設定を行うための手段として tpadmcall() という関数が用意されており、この関数を利用すると、マスタ・マシンのこれらの属性を設定することができます。

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 システムのメッセージ・キューの実行時属性を示します。

属性表

TM_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

TM_MIB(5): T_MSG **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_MSG_STIME	long	R--R--R--	1 <= *num*	N/A
(k) — GET キー・フィールド				

注 1 T_MSG クラスの属性は、すべてローカル属性です。

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。アプリケーションが動作しているプラットフォームが UNIX ベース以外の場合、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、選択された T_MSG オブジェクトの実行時情報を検索します。下に示した状態は、GET 要求に対する応答で返される TA_STATE の意味を示します。

ACTive	T_MSG オブジェクトがアクティブな状態に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、このクラスでは許可されません。

TA_CURTIME:1 <= *num*

T_MSG:TA_LMID で time(2) システム・コールによって返される、1970 年 1 月 1 日 00:00:00 UTC から起算した現在の時間。この属性は、T_MSG:TA_?TIME 属性の値から経過時間を求めるために利用できます。

TA_MSG_CBYTES:1 <= *num*

現在キューに存在するバイト数。

TA_MSG_CTIME:1 <= *num*

キューに関連のある *msqid_ds* 構造体のメンバを変更した最後の msgctl(2) オペレーションを実行した時間。

TA_MSG_LRPID:1 <= *num*
最後にキューからの読み取りを実行したプロセスの識別子。

TA_MSG_LSPID:1 <= *num*
最後にキューに書き込みを実行したプロセスの識別子。

TA_MSG_QBYTES:1 <= *num*
キューに書き込める最大バイト数。

TA_MSG_QNUM:1 <= *num*
現在キューに入っているメッセージの数。

TA_MSG_RTIME:1 <= *num*
最後にキューから読み取ってから経過した時間。

TA_MSG_STIME:1 <= *num*
最後にキューに書き込みを行っ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 バージョン以外の BEA Tuxedo システムではサポートされない場合があります。

T_NETGROUP クラスの定義

概要　T_NETGROUP のクラスは、ネットワーク・グループの属性を表します。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・アドレスによって通信できます。

属性表

TM_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。グループ名は印刷可能な文字から成る文字列で、シャープ記号 (#)、カンマ、コロンや改行復帰を入れることはできません。

TA_NETGRPNO:1 <= *num* <= 8192
ネットワーク・グループに関連するグループ識別子

TA_STATE:

GET:{VALid}
GET 操作は、選択した T_NETGROUP オブジェクトに対するコンフィギュレーション情報を検索します。下に示した状態は、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_NETGROUP オブジェクトが定義され、アクティブではありません。このクラスでは、有効な状態は VALid だけです。ネットグループ にアクティブなものはありません。

SET:{NEW | INValid}
SET 操作は、選択された T_NETGROUP オブジェクトに対するコンフィギュレーション情報を検索します。次に、SET 要求で TA_STATE に設定できる値の意味を説明します。記述されていない状態を設定することはできません。

NEW	アプリケーションに対する 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 オブジェクトを修正します。VALid 状態の時に限り修正ができます。正常終了すると、オブジェクトの状態は変わりません。
INValid	T_NETGROUP オブジェクトをアプリケーションから削除します。状態変更ができるのは、VALid 状態の時と、このネットワーク・グループのオブジェクトをキーとして持つ T_NETMAP クラスにオブジェクトが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・グループに関する優先順位バンドです。バンドの優先順位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。ある優先順位のネットワーク回線がすべて管理者またはネットワーク状態により分断されている場合、次に優先順位の高い回線を使用します。その後、より優先順位の高い回線への接続が再試行されます。

注記 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、別のアルゴリズムを使用して、並列のデータ回線の優先順位を決める可能性があります。

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 のクラスからの 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含まれます。TA_LMID は、多くの TA_NETGROUP グループに含めることができます。1 つの LMID が別の LMID に接続している場合、BRIDGE プロセスは 2 つの LMID が所属するネットワーク・グループのサブセットを決定します。1 対の LMID がいくつかの共通したグループにある時、その 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、ネットワークのデータを同時に送信します。ネットワークのエラーにより最も優先順位が高いグループを使ってデータが送信されない場合は、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 "*failover*" と呼びます。failover の状況では、より優先順位の高いグループへの接続が定期的に試行されます。TA_NETPRIO 値の高い接続が確立すると、優先順位の低い回線にはデータ送信が予定されることはありません。優先順位の低い接続がなくなると、その回線は順番に切断されます。これを "*failback*" と呼びます。

属性表

TM_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-------	*string*[1 . .30]	N/A
TA_STATE	string	RW-------	GET:"VAL" SET:"{NEW | INV}"	N/A N/A
TA_NADDR	string	rw-r--r--	*string*[1 . .78]	""
TA_FADDR	string	rw-r--r--	*string*[0..78]	""
TA_FRANGE	long	rw-r--r--	1 <= *num* <= 65,535	1
TA_MINENCRYPTBITS	string	rwxrwx---	{0|40|56|128}(注 1)	0
TA_MAXENCRYPTBITS	string	rwxrwx---	{0|40|56|128}(注 1)	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 リンク・レベルでの暗号化の値 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*
この属性は、T_NETGROUP のクラスにある関連ネットワーク・グループの名前です。

TA_LMID:*string*
このネットワークのマッピングに対する T_MACHINE のクラス （TM_MIB 中に存在）から継承される論理マシン名です。

TA_STATE:

GET:"{VALid}"

GET 操作は、選択した T_NETMAP オブジェクトに対する環境設定情報を検索します。下に示した状態は、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_NETMAP オブジェクトが定義されます。このクラスでは、有効な状態は VALid だけです。ネットワーク・グループには ACTive 状態のものはありません。

SET:"{NEW | INValid}"

SET 操作は、選択した T_NETMAP オブジェクトに対するコンフィギュレーション情報を更新します。次に、SET 要求で TA_STATE に設定できる値の意味を説明します。リストにない状態は設定されない場合があります。

NEW	アプリケーションに対する 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 オブジェクトを修正します。正常終了すると、オブジェクトの状態は変わりません。
INValid	指定されているネットワークのマッピングを削除します。マッピングの結果、ネットワーク・リンクがアクティブな場合、リンクの接続は切断されます。この切断により、ネットワーク・リンクに関連する T_BRIDGE オブジェクト（TM_MIB 内に存在する）で状態変更が発生する場合があります。

TA_NADDR:*string*

論理マシンに位置する BRIDGE プロセスにより接続指示を受け付けているアドレスとして使用されるネットワーク・アドレスの完全な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 BRIDGE 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、LAN オプションが T_DOMAIN:TA_OPTIONS 属性値に設定されているときには必ず設定します。

string の形式が "0xhex-*digits*" の場合は、有効な偶数桁の 16 進数でなければなりません。これらの形式は、指定した文字列の 16 進数表現を含む文字配列に内部で変換されます。

TCP/IP アドレスの場合には、以下のいずれかの形式が使用されます。

- "*//hostname*:*port*"
- "*//#.#.#.#*:*port*"

TA_FADDR:*string*[0..78]
: アウトバウンド接続を確立する前に tmboot、BRIDGE、BSBRIDGE、tmloadcf などのローカル・プロセスが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 アドレスを指定してください。この属性と TA_FRANGE 属性は、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。このパラメータが NULL または空文字列に設定されている場合は、オペレーティング・システムによってバインド先のローカル・ポートが任意に選択されます。

  string の形式が "0xhex-digits" の場合は、有効な偶数桁の 16 進数でなければなりません。これらの形式は、指定した文字列の 16 進数表現を含む文字配列に内部で変換されます。

  TCP/IP アドレスには、以下のいずれかの形式が使用されます。

  - "*//hostname*:*port*"
  - "*//#.#.#.#*:*port*"

TA_FRANGE:1<= *num* <= 65,535
: アウトバウンド接続を確立する前にローカル・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。TA_FADDR 属性は、範囲のベース・アドレスを指定します。

TA_MINENCRYPTBITS: {0 | 40 | 56 | 128}
: ネットワークのリンク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 は暗号化を行わないことを示します。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・レベルでの暗号化の値 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。

  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・リンクには影響はありません。

TA_MAXENCRYPTBITS: {0 | 40 | 56 | 128}
: リンクを確立する際に許可され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ます。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。デフォルト値は 128 です。

注記 リンク・レベルでの暗号化の値 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。

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・リンクには影響はありません。

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、56 がデフォルト値になります。暗号化の使用が許可されていない場合には、0 ビットがデフォルト値になります。BRIDGE 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、アプリケーションのキューの実行時属性を示します。これらの属性の値は、実行中のアプリケーションでサーバに関連のある割り当てられた BEA Tuxedo システムの要求キューを識別し、特性づけます。また、それぞれのキュー・オブジェクトに関連のあるアプリケーションの作業負荷に関する統計値を記録します。

複数のマシンが存在するアプリケーションで MIB_LOCAL フラグを利用した GET 操作を実行する際には、アクティブなそれぞれのキューに対して複数のオブジェクトが返されます （ローカル属性が収集されるそれぞれの論理マシンごとに 1 つ）。

属性表

TM_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..78]	N/A

TM_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_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				
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

属性の意味

TA_RQADDR:*string*[1..30]
要求キューのシンボリックなアドレス。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・アドレスに関連のある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、選択された 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、直接コンフィギュレーション情報を示すわけではありません。ここで説明した環境設定関連の属性は、関連のある T_SERVER オブジェクトの一部としてセットする必要があります。下に示した状態は、GET 要求に対する応答で返される TA_STATE の意味を示します。

ACTive	この T_QUEUE オブジェクトに関連のある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のあるサーバが現在 MIGrating 状態にあることを示します。この状態については、詳しくは T_SERVER クラスの項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際しては ACTive と同等です。
SUSpended	この T_QUEUE オブジェクトに関連のあるサーバが現在 SUSpended の状態にあることを示します。この状態については、詳しくは T_SERVER クラスの項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際しては ACTive と同等です。
PARtitioned	この T_QUEUE オブジェクトに関連のあるサーバが現在 PARtitioned の状態にあることを示します。この状態については、詳しくは T_SERVER クラスの項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際しては ACTive と同等です。

SET:

SET 操作は、選択された T_QUEUE オブジェクトの実行時の情報を更新します。状態の変更は、T_QUEUE オブジェクトの情報の更新時にのみ許可されます。既存の T_QUEUE オブジェクトの変更は、オブジェクトが ACTive の状態にある場合にのみ許可されます。

TA_GRACE:0 <= *num*

T_QUEUE:TA_MAXGEN の制約が適用される時間（単位は秒）を示します。この属性は、再起動が可能なサーバに対してのみ（すなわち、T_QUEUE:TA_RESTART 属性が "Y" にセットされている場合にのみ）意味を持ちます。この属性の値がゼロのときは、かならずサーバを再起動させる必要があります。

TA_MAXGEN:1 <= *num* < 256
: このキューに関連のある再起動可能なサーバ （T_QUEUE:TA_RESTART == "Y"）に対して指定された期間内 (T_QUEUE:TA_GRACE). に認められている世代（generation）の数。それぞれの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..78]
: このキューに関連のあるアプリケーション・サーバのシステムの再起動と同時に実行される、アプリケーションによって指定されるコマンド。

TA_RESTART:{Y | N}
: このキューに関連のあるサーバが再起動可能 （"Y"）か可能でない （"N"）を示します。

TA_CONV:{Y | N}
: このキューに関連のあるサーバが、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*
: このキューに関連のある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。アプリケーションが動作しているプラットフォームが UNIX ベース以外の場合、この属性は返されないことがあります。

TA_SERVERCNT:1 <= *num* < 8,192
: このキューに関連のある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であったときのキューの長さの合計。この合計値には、すでにアクティブでなくなっているサーバ宛てに送出され、そのサーバによって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、その要求がキューに登録される前にそのつど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" にセットされている場合、あるいは T_DOMAIN:TA_MODEL 属性が "MP" の場合は、TA_TOTNQUEUED は返されません。同じ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 は同じ値を持つことになります。

TA_TOTWKQUEUED:0 <= *num*

このキューがアクティブであった間にこのキューに対して要求された作業負荷の合計。この合計値には、すでにアクティブでなくなっているサーバ宛てに送出され、そのサーバによって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、その要求がキューに登録される前にそのつどその作業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" にセットされている場合、あるいは T_DOMAIN:TA_MODEL 属性が "MP" の場合は、TA_TOTWKQUEUED は返されません。同じ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 は同じ値を持つことになります。

TA_SOURCE:*LMID*

ローカル属性の値が検索される論理マシン。

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たびに加算され、サーバがキューから要求を取り出すたびに減算されます。

制限事項 : T_DOMAIN:TA_LDBAL 属性が "N" にセットされている場合、あるいは T_DOMAIN:TA_MODEL 属性が "MP" の場合は、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 は同じ値を持つことになります。

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作業負荷。T_DOMAIN:TA_MODEL 属性が SHM にセットされ、T_DOMAIN:TA_LDBAL 属性が "Y" にセットされている場合は、この属性はアプリケーション全体を通じてこのキューに割り当てられた作業負荷を示します。一方、TA_MODEL が MP にセットされ、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対して割り当てられた作業負荷を示します。この属性は、負荷の調節を目的として使用されます。新たに起動されるサーバが排除されないようにするために、BBL はこの値をそれぞれのマシンで周期的にゼロにセットします

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示します。これらの属性値は、フィールド名、バッファ・タイプ、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示します。

属性表

TM_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]	"SERVICE2
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。指定された 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。
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して解釈されます。

属性の意味

TA_ROUTINGNAME:*string*[1..15]
ルーティング条件の名前。

TA_ROUTINGTYPE:*type*

ルーティングのタイプを指定します。ATMI 環境で使用される既存の UBBCONFIG ファイルが継続して正常に機能することを保証するため、デフォルトで TYPE=SERVICE に設定されます。CORBA インターフェイス用にファクトリ・ベースの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*[,. . .]]] . . . "

このルーティング項目が適用されるデータ・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 / サブ・タイプの組合せを使用できます。タイプは、FML、FML32、XML、VIEW、VIEW32、X_C_TYPE、および X_COMMON に制限されます。FML、FML32、または XML に対してはサブタイプを指定することはできず、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 (*) といった記号は使用できません。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 / サブタイプのペアが一意であれば、複数のルーティング・エントリに対して同じ基準名を指定することができます。1 つのルーティングに対して複数のバッファ・タイプを指定した場合は、それぞれのバッファ・タイプのルーティング・フィールドのデータ・タイプを同じにする必要があります。

TA_FIELD:*string*[1..30]

ルーティング・フィールド名。TA_TYPE=FACTORY のとき、このファクトリ・ルーティング基準に対応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見なされます。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、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、あるいは 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) と解釈されます。この情報は、メッセージ送信時に、データ依存型ルーティングの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。インデックス付けはサポートされません。したがって、データ依存型ルーティングで XML バッファを処理するとき、BEA Tuxedo システムは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。この情報は、メッセージ送信時に、データ依存型ルーティングの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なければなりません。

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、メッセージ送信時に、データ依存型ルーティングの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なければなりません。

要素名と属性名の組み合わせでは、最大 30 文字まで使用できます。

ルーティング・フィールドの型は、TA_FIELDTYPE 属性によって指定できます。

TA_FIELDTYPE:[char, short, long, float, double, string]
TA_FIELD 属性で指定されるルーティング・フィールドの型。フィールドの型には、char、short、long、float、double、または string を指定できますが、指定できるのは 1 つの型だけです。この属性は、XML バッファのルーティング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、または 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のあるサーバ・グループ。*string* の形式は、カンマで区切って並べられた範囲 / グループ名の組合せのになります。範囲 / グループ名の組合せの形式は下記の通りです。

lower[-upper]:group

lower と *upper* は、符号を持つ数値、またはシングル・クォーテーションに挟まれた文字列です。*lower* は、*upper* 以下でなければなりません。文字列の値にシングル・クォーテーションを使用するためには、シングル・クォーテーションの前に 2 つのバック・スラッシュを入力します（例 : 'O\\'Brien'）。MIN という値を使用すると、マシンの関連フィールドのデータ・タイプに対する最小値を示すことができます。また、MAX という値を使用すると、マシンの関連フィールドのデータ・タイプの最大値を示すことが

できます。たとえば、"MIN--5" は、-5 以下のあらゆる数字を指し、"6-MAX" は、6 以上のあらゆる数字を指します。

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 (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() 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の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、その要求が、必要なサービスを提供するサーバのグループならどこへでも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 以前のバージョンの互換性が失われます。

TA_STATE:

GET:{VALid}

GET 操作は、選択された 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。下に示した状態は、GET 要求に対する応答で返される TA_STATE の意味を示します。記述されていない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。このクラスでは、有効な状態は VALid だけです。ルーティング条件は ACTive になることはなく、コンフィギュレーションを通じ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、パーミッションの決定に際しては INActive と同等です。

SET:{NEW | INValid}

SET 操作は、選択された 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。次に、SET 要求で TA_STATE に設定できる値の意味を説明します。記述されていない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_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。INValid 状態ではこの組み合わせは使用できません。正常終了すると、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_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。デフォルトでは「SERVICE」が指定されます。したがって、データ依存型ルーティングを使用する場合、この属性の指定は省略可能です。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、個別に説明されます。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、アプリケーション中のサーバのコンフィギュレーション情報および実行時情報を示します。これらの属性の値は、環境設定の済んでいるサーバを識別し、特性づけるとともに、それぞれのサーバ・オブジェクトに関する統計値やリソースに関する実行時の情報を提供します。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、このクラスには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には、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。

属性の 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 は、各サーバ・ディスパッチ・コンテキストに固有のものです。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

TM_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
TA_CLOPT	string	rwyr--r--	*string*[0..256]	"-A"
TA_ENVFILE	string	rwyr--r--	*string*[0..78]	""
TA_GRACE	long	rwyr--r--	0 <= *num*	86,400

TM_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_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* < 2,147,483,647	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..78]	""
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)
TA_LMID(k)	string	R--R--R--	*LMID*	N/A
TA_GENERATION	long	R--R--R--	1 <= *num* < 32,768	N/A
TA_PID(k)	long	R--R--R--	1 <= *num*	N/A
TA_RPID	long	R--R--R--	1 <= *num*	N/A

TM_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_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
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

TM_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_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 でこの属性に対して指定した値になります。

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 (:) を入れ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_SERVERNAME:*string*[1..78]
サーバ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・グループ用の 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に最初に現れる PATH= 行の値です。アクティブ・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 オブジェクトからとります。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対応するグループ番号

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と実行時情報を検索します。下に示した状態は、GET 要求に対する応答で返される TA_STATE の意味を示します。

ACTive	T_SERVER オブジェクトは定義されていて、アクティブ状態です。これはサーバがアイドル状態またはビジー状態であることを意味しません。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されているが、非アクティブ状態です。
MIGrating	T_SERVER オブジェクトは定義されていて、現在は、サーバ・グループの 2 次論理マシンへの移行状態です。2 次論理マシンは、T_GROUP:TA_CURLMID 属性に一致しない T_GROUP:TA_LMID 属性にリストされているマシンです。この状態は、パーミッションの決定に際しては ACTive と同等です。
CLEaning	T_SERVER オブジェクトは定義されていて、異常終了によってその後システムが現在クリーンアップしている状態です。実行可能サーバがその TA_GRACE の時間内ではあるが、TA_MAXGEN の起動 / 再起動時間を超えた場合は、その実行可能サーバはこの状態に入ります。この状態は、パーミッションの決定に際しては ACTive と同等です。
REStarting	T_SERVER オブジェクトは定義されていて、異常終了によってシステムが現在再起動している状態です。この状態は、パーミッションの決定に際しては ACTive と同等です。
SUSpended	T_SERVER オブジェクトは定義されていて、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際しては ACTive と同等です。

PARtitioned	T_SERVER オブジェクトが定義され、アクティブです。しかし、サーバが動作しているマシンが現在 T_DOMAIN:TA_MASTER サイトから切り離されています。この状態は、パーミッションの決定に際しては ACTive と同等です。
DEAd	T_SERVER オブジェクトが定義され、掲示板でアクティブと識別されています。しかし、現在異常終了により動作していません。この状態は、サーバのローカル BBL が終了を検知し、動作 (REStarting|CLEaning) を行うまで存在します。この状態は、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返されます。この状態は、パーミッションの決定に際しては ACTive と同等です。

SET:{NEW | INValid | ACTive | INActive | DEAd}

SET 操作は、選択した T_SERVER オブジェクトに対するコンフィギュレーション情報と実行時情報を更新します。次に、SET 要求で TA_STATE に設定できる値の意味を説明します。記述されていない状態を設定することはできません。

NEW	アプリケーションに対し T_SERVER 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 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 または 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変わりません。
INValid	アプリケーションに対し 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 をアクティブ化します。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 に設定して再開しなければなりません。）この状態の推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が考慮されます（--x--x--x）。正常終了すると、オブジェクトの状態は ACTive になります。個々のサーバの状態が必要な場合にサーバをアクティブ化するには、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化します。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 INActive になります。個々のサーバの状態が必要な場合にサーバを非アクティブ化するには、TMIB_NOTIFY TA_FLAG 値を使用します。
DEAd	適当なタイムアウト間隔（MIB(5) の TA_MIBTIMEOUT 参照）後になおサーバが動作している場合、サーバに SIGTERM シグナルと SIGKILL シグナルを送信し 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し、再開可能であってもサーバが非アクティブになるので注意して下さい。サーバが長時間かかるサービスを処理中、あるいは SIGTERM シグナルの無効を選択した場合は、SIGKILL が使用でき、システムは異常終了として処理します。状態の変更は、ACTive または SUSpended 状態でのみ可能です。正常終了の場合は、オブジェクトは INActive、CLEaning、または REStarting 状態になります。

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。しかし、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]
: アクティブ時、コマンド行オプションを渡します。詳細については、servopts(5) マニュアル・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、動作中のサーバには影響ありません。

TA_ENVFILE:*string*[0..78]
: サーバ固有環境ファイル。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、動作中のサーバには影響ありません。

TA_GRACE:0 <= *num*
: T_SERVER:TA_MAXGEN 制限値が適用される時間（秒）。この属性は、再開可能なサーバ、すなわち T_SERVER:TA_RESTART 属性を "Y" に設定した場合のみ意味を持ちます。再開サーバが TA_MAXGEN 制限値を超えても、TA_GRACE 時間に達していれば、システムは現在の世代 (すなわち、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。この属性の値がゼロのときは、かならずサーバを再起動させる必要があります。

  要求キューを共有するサーバ（すなわち、T_SERVER:TA_RQADDR の値が同一のサーバ）はこの属性の値も同一に設定しなければなりません。同一の値でない場合、最初にアクティブ化されたサーバがキュー上の全サーバに対応する実行時値を確立します。

  制限事項 : この属性を実行時に変更しても、動作中のサーバに影響はありません。また要求キューを共有しているほかのアクティブ・サーバにも影響はありません。ただし、変更されるのは、選択されたサーバのコンフィギュレーション・パラメータだけです。したがって、管理者がキューを共有する全サーバに対してこの属性を同一値に設定されるよう保証しない限り、アプリケーションの動作は以降のアクティブ化時のブート順序によって異なります。

TA_MAXGEN:1 <= *num* < 256
: 指定された猶予時間内に再開可能なサーバ（T_SERVER:TA_RESTART == "Y"）に対して許される世代数。最初のサーバのアクティブ化が 1 つの世代と数えられ、各再開も 1 つの世代と数えられます。最大世代数を超えた後の処理については前述の TA_GRACE を参照してください。

  要求キューを共有するサーバ（すなわち、T_SERVER:TA_RQADDR の値が同一のサーバ）はこの属性の値も同一に設定しなければなりません。同一の値でない場合、最初にアクティブ化されたサーバがキュー上の全サーバに対応する実行時値を確立します。

制限事項 : この属性を実行時に変更しても、動作中のサーバに影響はありません。また要求キューを共有しているほかのアクティブ・サーバにも影響はありません。ただし、変更されるのは、選択されたサーバのコンフィギュレーション・パラメータだけです。したがって、管理者がキューを共有する全サーバに対してこの属性を同一値に設定されるよう保証しない限り、アプリケーションの動作は以降のアクティブ化時のブート順序によって異なります。

TA_MAX:1 <= *num* < 1,001

ブートされるサーバの最大数。最初に、tmboot() がサーバの T_SERVER:TA_MIN オブジェクトをブートし、その他のオブジェクトは個別の特定サーバ ID を指定）または自動起動（会話型サーバのみ）によって個別に開始されます。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セットの動作中のサーバ（上記の TA_BASESRVID 参照）、およびサーバの環境設定の定義に影響を与えます。

TA_MIN:1 <= *num* < 1,001

tmboot(1) によってブートされる、サーバの最小数。T_SERVER:TA_RQADDR が指定され、TA_MIN が 1 より大きい場合、サーバは MSSQ セットを構成します。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、動作中のサーバには影響ありません。

TA_MINDISPATCHTHREADS:1 <= *num* < 1,000

最初のサーバ起動時に開始されるサーバ・ディスパッチ・スレッドの数を指定します。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ている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 の値の一部としてカウントされません。TA_MINDISPATCHTHREADS <= TA_MAXDISPATCHTHREADS でなければなりません。TA_MINDISPATCHTHREADS が指定されていない場合、0 がデフォルト値になります。

制限事項 : この属性を実行時に変更しても、動作中のサーバには影響ありません。

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ている場合にのみ有効です。

TA_MAXDISPATCHTHREADS > 1 の場合、個別のディスパッチャ・スレッドが使用されますが、この制限にはカウントされません。TA_MINDISPATCHTHREADS <= TA_MAXDISPATCHTHREADS でなければなりません。TA_MAXDISPATCHTHREADS が指定されていない場合、1 がデフォルト値になります。

制限事項 : この属性を実行時に変更しても、動作中のサーバには影響ありません。

TA_THREADSTACKSIZE = *number*

マルチスレッド・サーバで各ディスパッチ・スレッドに対して作成されるスタックのサイズ。このオプションがサーバに対して有効なのは、TA_MAXDISPATCHTHREADS に 1 より大きい値が指定されている場合のみです。

この属性が指定されていない場合、または 0 に設定されている場合、デフォルトのスレッド・スタック・サイズが使用されます。使用されるデフォルト・サイズは、オペレーティング・システムのデフォルト・サイズですが、このサイズがマルチスレッドの BEA Tuxedo アプリケーションにとって十分なサイズであることがわかっている場合に限ります。不十分な場合には BEA Tuxedo でのデフォルト・サイズが使用されます。現在、BEA Tuxedo でのデフォルトのスレッド・スタック・サイズは 1,024,000 です。

スレッド・スタック・サイズを超えると、サーバはコア・ダンプするので注意してください。

制限事項 : この属性を実行時に変更しても、動作中のサーバには影響ありません。

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*

再起動後の、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・スレッドの最大数。管理者が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ことがあるため、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*

再起動後の、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・スレッドの合計数。

TA_RCMD:*string*[0..78]

システムによるアプリケーション・サーバの再開と同時に実行する、アプリケーションが指定するコマンドです。

要求キューを共有するサーバ（すなわち、T_SERVER:TA_RQADDR の値が同一のサーバ）はこの属性の値も同一に設定しなければなりません。同一の値でない場合、最初にアクティブ化されたサーバがキュー上の全サーバに対応する実行時値を確立します。

制限事項 : この属性を実行時に変更しても、動作中のサーバに影響はありません。また要求キューを共有しているほかのアクティブ・サーバにも影響はありません。ただし、変更されるのは、選択されたサーバのコンフィギュレーション・パラメータだけです。したがって、管理者がキューを共有する全サーバに対してこの属性を同一値に設定されるよう保証しない限り、アプリケーションの動作は以降のアクティブ化時のブート順序によって異なります。

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、以下のいずれかの方法を使用する必要があります。

- コマンド・ファイル内またはスクリプト内からリダイレクトまたはパイプします。
- キュー・マネージャ管理プログラム内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
  ```
- バイナリ形式の実行可能ファイルを作成する場合は、Windows の AllocConsole() API 関数を使用して、バイナリ形式の実行可能ファイル内のコンソールを割り当てる必要があります。

TA_RESTART:{Y | N}

再開可能 ("Y") または再開不可能 ("N") に設定します。このサーバ・グループに対してサーバ移行が指定されると（T_DOMAIN:TA_OPTIONS/MIGRATE および代替サイトによる T_GROUP:TA_LMID）、この属性は "Y" に設定しなければなりません。

要求キューを共有するサーバ（すなわち、T_SERVER:TA_RQADDR の値が同一のサーバ）はこの属性の値も同一に設定しなければなりません。同一の値でない場合、最初にアクティブ化されたサーバがキュー上の全サーバに対応する実行時値を確立します。

制限事項 : この属性を実行時に変更しても、動作中のサーバに影響はありません。また要求キューを共有しているほかのアクティブ・サーバにも影響はありません。ただし、変更されるのは、選択されたサーバのコンフィギュレーション・パラメータだけです。したがって、管理者がキューを共有する全サーバに対してこの属性を同一値に設定されるよう保証しない限り、アプリケーションの動作は以降のアクティブ化時のブート順序によって異なります。

TA_SEQUENCE:1 <= *num* < 10,000
: いつ、このサーバを他のサーバに対してブート（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、シーケンス番号の昇順に従って tmboot() でブートされ、降順で tmshutdown() 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() にしか影響を及ぼさず、稼働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を与えます。

TA_SYSTEM_ACCESS:{FASTPATH | PROTECTED}
: BEA Tuxedo システム の内部テーブルにアクセスするために、このサーバプロセスが BEA Tuxedo システム ライブラリで使用するモード。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、動作中のサーバには影響ありません。(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。あるスレッドが BEA Tuxedo コードを実行中、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・コードを実行していることがあるからです。BEA Tuxedo システムでは、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") かを指定します。

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"）。応答受信を期待する 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 システム上でリダイレクトまたはパイプする場合は、TA_RCMD 属性の項に記載されている方法のいずれか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。別の応答キューが割り当てられない場合 (T_SERVER:TA_REPLYQ == "N")、この属性は無視されます。

  注記 Windows 2000 システム上でリダイレクトまたはパイプする場合は、TA_RCMD 属性の項に記載されている方法のいずれか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一の TA_RQADDR 属性値を指定すると、複数サーバ単一キュー (MSSQ) のセットを定義できます。同一の TA_RQADDR オプション属性値を持つサーバは、同一のサーバ・グループでななければなりません。

TA_RQPERM:0001 <= *num* <= 0777
: サーバの要求キューに対する UNIX システムパーミッション。

  制限事項 : UNIX システム固有の属性です。アプリケーションが動作しているプラットフォームが UNIX ベース以外の場合、この属性は返されないことがあります。

TA_LMID:*LMID*
: サーバが稼働している現在の論理マシン。

TA_GENERATION:1 <= *num* < 32,768
: サーバの世代。サーバが最初に tmboot(1) でブートされたり、TM_MIB でアクティブ化されると、その世代は 1 に設定されます。サーバが異常終了し、再開されるごとに、世代が増加します。T_SERVER:TA_MAXGEN を超え、T_SERVER:TA_GRACE に達すると、サーバが再起動され、世代が 1 にリセットされるので注意して下さい。

TA_PID:1 <= *num*
: サーバの UNIX システムのプロセス ID。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には、この属性は一意となりませんので注意して下さい。

  制限事項 : UNIX システム固有の属性です。アプリケーションが動作しているプラットフォームが UNIX ベース以外の場合、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・メッセージ・キュー識別子。別の応答キューが割り当てられない場合 (T_SERVER:TA_REPLYQ == "N")、この属性は T_SERVER:TA_RQID と同一になります。

  制限事項 : UNIX システム固有の属性です。アプリケーションが動作しているプラットフォームが UNIX ベース以外の場合、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・メッセージ・キュー識別子。別の応答キューが割り当てられない場合 (T_SERVER:TA_REPLYQ == "N")、この属性は T_SERVER:TA_RPID と同一になります。

制限事項 : UNIX システム固有の属性です。アプリケーションが動作しているプラットフォームが UNIX ベース以外の場合、この属性は返されないことがあります。

TA_TIMERESTART:1 <= *num*

1970 年 1 月 1 日 00:00:00 UTC を基準とする時間（秒）。サーバが最後に起動または再起動されたときに、T_SERVER:TA_LMID に対して time(2) システム・コールで返される値です。

TA_TIMESTART:1 <= *num*

1970 年 1 月 1 日 00:00:00 UTC を基準とする時間（秒）。サーバが最初に開始されたときに、T_SERVER:TA_LMID に対する time(2) システム・コールで返される値です。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 に達している場合には、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・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、セキュリティのプリンシパル名を識別する文字列。この属性には、最後のヌル文字を除き、最大 511 文字まで指定できます。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 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階層の 4 つのレベル T_DOMAIN クラス、T_MACHINE クラス、T_GROUP クラス、および T_SERVER クラス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 TA_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は、デフォルトでこのドメインの TA_DOMAINID 文字列に設定されます。

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 以降を実行するシステム・プロセスに対して復号化キーをオープンする処理で指定します。特定のレベルで TA_SEC_PRINCIPAL_NAME だけが指定されている場合には、これ以外の 2 つの属性はそれぞれ、長さゼロの NULL 文字列に設定されます。

TA_SEC_PRINCIPAL_LOCATION:*string*[0..511]

TA_SEC_PRINCIPAL_NAME で指定されたプリンシパルの復号化 (秘密) 鍵を収めるファイルまたはデバイスの場所。この属性には、最後のヌル文字を除き、最大 511 文字まで指定できます。

TA_SEC_PRINCIPAL_LOCATION は、コンフィギュレーション階層の 4 つのレベル T_DOMAIN クラス、T_MACHINE クラス、T_GROUP クラス、および T_SERVER クラスのいずれでも指定できます。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 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。この属性には、最後のヌル文字を除き、最大 511 文字まで指定できます。

TA_SEC_PRINCIPAL_PASSVAR は、コンフィギュレーション階層の 4 つのレベル T_DOMAIN クラス、T_MACHINE クラス、T_GROUP クラス、および T_SERVER クラスのいずれでも指定できます。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)。

初期化処理中、管理者は、TA_SEC_PRINCIPAL_PASSVAR で設定された、復号化キーのそれぞれのパスワードを入力する必要があります。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。

TA_NUMCONV:0 <= *num*

このサーバが tpconnect() で起動した会話の数。

TA_NUMDEQUEUE:0 <= *num*

このサーバが tpdequeue() で実行したキューからの取り出し操作の回数。

TA_NUMENQUEUE:0 <= *num*

このサーバが tpenqueue() で実行したキューへの登録操作の回数。

TA_NUMPOST:0 <= *num*

このサーバが tppost() で実行したポストの回数。

TA_NUMREQ:0 <= *num*

このサーバが tpcall() または tpacall() で発行した要求の数。

TA_NUMSUBSCRIBE:0 <= *num*
: このサーバが tpsubscribe() で実行したサブスクリプションの回数。

TA_NUMTRAN:0 <= *num*
: このサーバが最後の起動（または再起動）以降に開始したトランザクション数。

TA_NUMTRANABT:0 <= *num*
: このサーバが最後の起動（または再起動）以降に中途終了したトランザクション数。

TA_NUMTRANCMT:0 <= *num*
: このサーバが最後の起動（または再起動）以降にコミットしたトランザクション数。

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総作業ロード。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作業ロードを示します。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、このフィールドの要素は、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。

  要求元クライアントまたはサーバは、サーバが現在実行しているサービス要求を行ったプロセスです。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意味を持ちます。マルチコンテキスト・サーバの場合は、空文字列がプレースホルダとして返されます。

TA_CLTPID:1 <= *num*
: 要求元クライアントまたはサーバの UNIX システムのプロセス

  また、このフィールドの要素は、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意味を持ちます。マルチコンテキスト・サーバの場合は、0 がプレースホルダとして返されます。

制限事項 : UNIX システム固有の属性です。アプリケーションが動作しているプラットフォームが UNIX ベース以外の場合、この属性は返されないことがあります。

TA_CLTREPLY:{Y | N}
要求元クライアントまたはサーバが応答を期待しているか (Y)、いないか (N) を指定します。

また、このフィールドの要素は、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意味を持ちます。マルチコンテキスト・サーバの場合は、空文字列がプレースホルダとして返されます。

TA_CMTRET:{COMPLETE | LOGGED}
このサーバの TP_COMMIT_CONTROL 特性の設定。

また、このフィールドの要素は、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 関数呼び出しの tpscmt() 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意味を持ちます。マルチコンテキスト・サーバの場合は、空文字列がプレースホルダとして返されます。

TA_CURCONV:0 <= *num*
このサーバが tpconnect() で起動した会話のうち、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_CURREQ:0 <= *num*
このサーバが tpcall() または tpacall() で起動した要求のうち、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存在する場合、そのサービスの名前。

また、このフィールドの要素は、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意味を持ちます。マルチコンテキスト・サーバの場合は、0 がプレースホルダとして返されます。

TA_CURTIME:1 <= *num*

1970 年 1 月 1 日 00:00:00 UTC を基準とする時間（秒）。T_SERVER:TA_LMID の time(2) システム・コールで返された時間です。この属性は、T_SERVER:TA_TIMESTART および T_SERVER:TA_TIMERESTART 属性値からの経過時間を計算するために使用されます。

TA_LASTGRP:1 <= *num* < 30,000

この外部サーバが最後に発行したサービス要求または起動した会話のサーバ・グループ番号 (T_GROUP:TA_GRPNO)。

また、このフィールドの要素は、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意味を持ちます。マルチコンテキスト・サーバの場合は、0 がプレースホルダとして返されます。

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（秒）。

また、このフィールドの要素は、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、タイムアウト処理が行われません。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意味を持ちます。マルチコンテキスト・サーバの場合は、0 がプレースホルダとして返されます。

TA_TIMELEFT:0 <= *num*

タイムアウトになるまでに、このサーバが現在待っている応答を受信することのできる残り時間。

また、このフィールドの要素は、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意味を持ちます。マルチコンテキスト・サーバの場合は、0 がプレースホルダとして返されます。

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、このフィールドの要素は、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。

0 は、サーバが現在トランザクションに使用され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意味を持ちます。マルチコンテキスト・サーバの場合は、0 がプレースホルダとして返されます。

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個々のサーバ・ディスパッチ・コンテキストのコンフィギュレーションおよび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、このクラスの値は T_SERVER クラスの一部として繰り返し使用されます。T_SERVERCTXT クラスの属性は読み取り専用です。

これらの属性の値は、各サーバ・ディスパッチ・コンテキストに関する統計値およびリソースの実行時の情報を提供します。

属性表

TM_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

TM_MIB(5): T_SERVERCTXT **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_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

注 1 T_SERVERCTXT クラスの属性は、すべてローカル属性です。

属性の意味

TA_SRVGRP:*string*[1..30]
サーバ・グループの論理名。サーバ・グループ名にはアスタリスク (*)、カンマ (,)、コロン (:) を入れ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_CONTEXTID:0 <= *num* < 30000
この特定のサーバ・コンテクストの識別子。

TA_CLTLMID:*LMID*
要求元クライアントまたはサーバの論理マシン。要求元クライアントまたはサーバは、サーバが現在実行しているサービス要求を行ったプロセスです。

TA_CLTPID:1 <= *num*
要求元クライアントまたはサーバの UNIX システムのプロセス。

制限事項 : UNIX システム固有の属性です。アプリケーションが動作しているプラットフォームが UNIX ベース以外の場合、この属性は返されないことがあります。

TA_CLTREPLY:{Y | N}
要求元クライアントまたはサーバが応答を期待しているか ("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 の ATMI 関数 tpscmt(3c) を参照してください。

TA_CURCONV:0 <= *num*
このサーバが tpconnect() で起動した会話のうち、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() で起動した要求のうち、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存在する場合、そのサービスの名前。

TA_LASTGRP:1 <= *num* < 30,000
: この外部サーバが最後に発行したサービス要求または起動した会話のサーバ・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（秒）。アクティブなサービスに対して 0 を指定すると、タイムアウト処理が行われません。詳細については、T_SERVICE:TA_SVCTIMEOUT を参照してください。

TA_TIMELEFT:0 <= *num*
: タイムアウトになるまでに、このサーバが現在待っている応答を受信することのできる残り時間。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。0 は、サーバが現在トランザクションに使用され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。属性値は環境設定済みのサービスを識別し、特性付けます。T_SERVICE オブジェクトは T_SVCGRP クラスの一部として特に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起動時コンフィギュレーション属性を指定します。アプリケーション中でアクティブなサービスについての実行時情報は、T_SVCGRP クラスでのみ取得可能です。T_SERVICE クラスへの実行時更新は通常アクティブ T_SVCGRP オブジェクトには反映されません （TA_ROUTINGNAME は例外)。

T_SERVICE クラス、T_SVCGRP クラス共にアプリケーション内のサービス名に対する起動時属性設定を定義します。新規にサービスがアクティブ化（宣言）されると、サーバの初期起動または tpadvertise() 呼び出しをするため、以下の順序でサービス開始時に使用する属性値を決定します。

1. コンフィギュレーションが一致する T_SVCGRP オブジェクトが存在する場合（サービス名とサービス・グループ一致)、このオブジェクトで定義された属性を使用して、宣言されたサービスの初期コンフィギュレーションを行います。
2. 1 に該当しない場合、コンフィギュレーションが一致する T_SERVICE オブジェクトが存在する場合（サービス名とサービス・グループ一致)、このオブジェクトで定義された属性を使用して、宣言されたサービスの初期コンフィギュレーションを行います。

3. 2 に該当しない場合、コンフィギュレーション済みの T_SVCGRP オブジェクトのいずれかの TA_SERVICENAME 属性値が一致する場合、最初に検出されたオブジェクトを使用して、宣言サービスの初期コンフィギュレーションを行います。
4. 上記ケースがいずれも該当しない場合、サービス属性に対するシステム・デフォルト値を使用して、宣言されたサービスの初期コンフィギュレーションを行います。

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、サービス起動時にサーバが宣言したサービスは、コンフィギュレーション値が未指定の場合、確立されたデフォルト・サービス値を使用します（サービス起動時に属性値を識別する方法については、上記を参照）。サーバが提示するサービス名は実行時に作成され（buildserver(1) 参照）、サーバ・オブジェクトにコマンド行オプションが指定されると、こちらが優先されます（T_SERVER:TA_CLOPT および servopts(5) 参照）。

属性表

TM_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_ROUTINGNAME	string	rwxr--r--	*string*[0..15]	""
TA_SIGNATURE_REQUIRED	string	rwxr--r--	"{Y|N}"	"N"
TA_ENCRYPTION_REQUIRED	string	rwxr--r--	"{Y|N}"	"N"

TM_MIB(5): T_SERVICE **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

(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 オブジェクトに対するコンフィギュレーション情報を検索します。下に示した状態は、GET 要求に対する応答で返される TA_STATE の意味を示します。

ACTive	T_SERVICE オブジェクトが定義され、TA_SERVICENAME 値に一致する T_SVCGRP オブジェクトのうち、少なくとも 1 つはアクティブです。
INActive	T_SERVICE が定義され、TA_SERVICENAME 値が一致する T_SVCGRP オブジェクトのうち、どれも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 オブジェクトに対するコンフィギュレーション情報を更新します。次に、SET 要求で TA_STATE に設定できる値の意味を説明します。記述されていない状態を設定することはできません。

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 : 環境設定がされていないサービスが、これらを宣言しているサーバによってまだアクティブな状態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。INValid 状態ではこの組み合わせは使用できません。正常終了すると、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}

要求が既にトランザクションモード以外の場合、このサービスに対してサービス要求メッセージが受信されると、自動的にトランザクションを開始します ("Y")。

制限事項 : この属性を実行時に更新しても、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。ロード・バランシングを行うために、サービスロードが使用されます。すなわち、追加されている作業ロードが高いキューは新規要求に対してあまり選択されません。サービスロードは、T_DOMAIN:TA_LDBAL が "Y" に設定される場合のみ意味があります。

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るようになります。複数のサービス要求がサービス・キューで待ち状態の場合、優先順位の高い要求からサービスが提供されます。

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処理要求の時間制限（秒）。このサービスに対するサーバの処理サービス要求が要求処理時に指定時間制限値を超えると、異常終了されます （kill -9)。この属性を 0 に設定すると、サービスは異常終了されません。

制限事項 : この属性を実行時に更新しても、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 オブジェクトに対して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（秒）。トランザクションモード以外の要求が受信され、サーバの T_SERVICE:TA_AUTOTRAN 属性値が "Y" のとき、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更新しても、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られるデータ・バッファのタイプとサブタイプのリスト。最大 32 のタイプ / サブタイプの組み合わせが可能です。BEA Tuxedo システムで提供されるデータ・バッファのタイプには、FML と FML32 (FML 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) 参照）。サブタイプを持つバッファ・タイプの場合、サブタイプに "*" を指定すると、サービスが対応バッファ・タイプに対するサブタイプをすべて受け付けます。

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) 参照)。TA_BUFTYPE 属性値が ALL の場合、サービスはそのバッファ・タイプ・スイッチにあるすべてのバッファ・タイプを受け入れます。

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。タイプ名およびサブタイプ名にはセミコロン (;)、カンマ (,)、コロン (:)、およびアスタリスク (*) は使用できないので注意して下さい。

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・サービス各インスタンスがサポートしなければならないバッファ・タイプを表します。この属性値はサービス起動時に処理されるため、この属性の更新は、一致するサービス名を持つアクティブな T_SVCGRP オブジェクトがない場合のみ可能です。

TA_ROUTINGNAME:*string*[0..15]

この T_SERVICE オブジェクトには、指定ルーティング判定基準名があります。この属性をアクティブ時に更新すると、対応する T_SVCGRP オブジェクトすべてに反映されます。

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・バッファへのデジタル署名が必要となります。指定されていない場合、N がデフォルト値になります。この属性は、BEA Tuxedo 7.1 以降のソフトウェアが稼動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階層の 4 つのレベル T_DOMAIN クラス、T_MACHINE クラス、T_GROUP クラス、および T_SERVICE クラス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プロセスで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。指定されていない場合、N がデフォルト値になります。この属性は、BEA Tuxedo 7.1 以降のソフトウェアが稼動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階層の 4 つのレベル T_DOMAIN クラス、T_MACHINE クラス、T_GROUP クラス、および T_SERVICE クラスのいずれでも指定できます。特定のレベルで TA_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プロセスで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 / グループのコンフィギュレーションおよび実行時属性を表します。コンフィギュレーションおよび実行時属性値は環境設定済みのサービス / グループを識別し、特性付けます。また、各オブジェクトに対応する統計およびリソースを実行時に追跡します。

T_SERVICE クラス、T_SVCGRP クラス共にアプリケーション内のサービス名に対する起動時属性設定を定義します。新規にサービスがアクティブ化（宣言）されると、サーバの初期起動または tpadvertise() 呼び出しをするため、以下の順序でサービス開始時に使用する属性値を決定します。

1. コンフィギュレーションが一致する T_SVCGRP オブジェクトが存在する場合（サービス名とサービス・グループ一致）、このオブジェクトで定義された属性を使用して、宣言されたサービスの初期コンフィギュレーションを行います。
2. 1 に該当しない場合、コンフィギュレーションが一致する T_SERVICE オブジェクトが存在する場合（サービス名とサービス・グループ一致）、このオブジェクトで定義された属性を使用して、宣言されたサービスの初期コンフィギュレーションを行います。

3. 2 に該当しない場合、環境設定済みの T_SVCGRP オブジェクトのいずれかの TA_SERVICENAME 属性値が一致する場合、最初に検出されたオブジェクトを使用して、宣言サービスの初期コンフィギュレーションを行います。
4. 上記ケースがいずれも該当しない場合、サービス属性に対するシステム・デフォルト値を使用して、宣言されたサービスの初期コンフィギュレーションを行います。

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、サービス起動時にサーバが宣言したサービスは、コンフィギュレーション値が未指定の場合、確立されたデフォルト・サービス値を使用します（サービス起動時に属性値を識別する方法については、上記を参照）。サーバが提示するサービス名は実行時に作成され（buildserver(1) 参照）、サーバ・オブジェクトにコマンド行オプションが指定されると、こちらが優先されます（T_SERVER:TA_CLOPT および servopts(5) 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。サービスを提示するグループ内に複数のサーバがある場合、実行時に特定のサービス名 / グループ名の組み合わせは複数の T_SVCGRP クラスに対応されます。

属性表

TM_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

(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であれば、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、および関連する環境設定レコードに影響がありますが、同一のコンフィギュレーション・レコードから実行時属性を発生させた他のアクティブ・オブジェクトには影響ありません。
注 2 この属性値を指定しないと、TA_SERVICENAME と見なされます。

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。サーバ・グループ名にはアスタリスク (*)、カンマ (,)、コロン (:) を入れることはできません。サービス起動時に使用するサービス属性の検索順序については、上記 T_SVCGRP OVERVIEW セクションで説明されています。

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 オブジェクトに対する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。下に示した状態は、GET 要求に対する応答で返される TA_STATE の意味を示します。

ACTive	T_SVCGRP 属性と TA_SRVID 属性に対して返された値で識別されるサーバ内で、TA_SRVGRP オブジェクトがアクティブです。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 が定義され、アクティブではありません。
SUSpended	T_SVCGRP オブジェクトが定義され、アクティブで、現在中断されています。この状態では、アプリケーションからこのサービスにアクセスすることはできません。この状態は、パーミッションの決定に際しては ACTive と同等です。

PARtitioned	T_SVCGRP オブジェクトが定義され、アクティブで、現在アプリケーションのマスタ・サイトから切り離されています。この状態では、アプリケーションからこのサービスにアクセスすることはできません。この状態は、パーミッションの決定に際しては ACTive と同等です。

SET:{NEW | INValid | ACTive | INActive | SUSpended}

SET 操作は、選択した T_SVCGRP オブジェクトに対するコンフィギュレーション情報と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影響を及ぼします。次に、SET 要求で TA_STATE に設定できる値の意味を説明します。記述されていない状態を設定することはできません。

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 : 環境設定がされていないサービスが、これらを宣言しているサーバによってまだアクティブな状態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、更新できません。
unset	既存の T_SVCGRP オブジェクトを変更します。INValid 状態ではこの組み合わせは使用できません。正常終了すると、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。INActive、SUSpended、INValid 時のみ状態変更が可能です。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。この状態の推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が考慮されます（--x--x--x）。正常終了すると、オブジェクトの状態は ACTive になります。 制限事項 : 予約文字列 "." で始まるサービス名 (TA_SERVICENAME) の場合、状態変更は許可されません。
INActive	T_SVCGRP オブジェクトを非アクティブ化します。状態の変更は、SUSpended 状態でのみ可能です。正常終了の場合は、オブジェクトは INActive（環境設定済みのエントリ）または INValid（環境未設定エントリ）状態のいずれかになります。 制限事項 : 予約文字列 "_" で始まるサービス名 (TA_SERVICENAME) の場合、状態変更は許可されません。
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 : 予約文字列 "_" で始まるサービス名 (TA_SERVICENAME) の場合、状態変更は許可されません。

TA_AUTOTRAN:{Y | N}
: 要求が既にトランザクションモード以外の場合、このサービスに対してサービス要求メッセージが受信されると、自動的にトランザクションを開始します ("Y")。

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。ロード・バランシングを行うために、サービスロードが使用されます。すなわち、追加されている作業ロードが高いキューは新規要求に対してあまり選択されません。

TA_PRIO:1 <= *num* < 101
: この T_SVCGRP オブジェクトには、キューから取り出す優先順位が設定されています。複数のサービス要求がサービス・キューで待ち状態の場合、優先順位の高い要求からサービスが提供されます。

TA_SVCTIMEOUT:0 <= *num*
: このサービス名に対する処理要求の時間制限（秒）。このサービスに対するサーバの処理サービス要求が要求処理時に指定時間制限値を超えると、異常終了されます（kill -9）。この属性を 0 に設定すると、サービスは異常終了されません。

  制限事項 : この属性値は、BEA Tuxedo リリース 4.4.2 以前のサイトでは適用されません。

TA_TRANTIME:0 <= *num*
: この T_SVCGRP オブジェクトに対して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（秒）。トランザクションモード以外の要求が受信され、サーバの T_SVCGRP:TA_AUTOTRAN 属性値が "Y" のとき、トランザクションが自動的に開始されます。

TA_LMID:*LMID*
: このサービスを提示するアクティブなサーバが動作する現在の論理マシン。

TA_RQADDR:*string*[1..30]
: このサービスを提示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
: このサービスを提示するアクティブなサーバの（サーバ・グループ内で）一意なサーバ識別番号。この属性のについては、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 サービス・オブジェクトの関数名は不明となり、文字列 "?" が属性値として返されます。

  制限事項 : この属性は、INActive から ACTive への状態変更時にのみ設定されます。

TA_BUFTYPE:*string*[1..256]
: このサービスで受け付けられる環境設定済みのバッファ・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]
: ルーティング条件の名前。

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*

アクティベイト (宣言) されてから検索された ACTive、または SUSpended オブジェクトに関しての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 属性が "Y" に設定されている場合のみ返されます。

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意味します。SYSTEM サービスはアプリケーション・クライアントやサーバから直接アクセスできません。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 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 サービスのみ検索されます。

現在このサービスへのキューに登録されている要求の数。この属性は追加時に増加し、サーバが要求を削除すると減少します。

制限事項 : この属性は、T_DOMAIN:TA_MODEL 属性が SHM に、T_DOMAIN:TA_LDBAL 属性が "Y" に設定されている場合のみ返されます。

TA_NQUEUED:0 <= *num* < 32,768

現在このサービスへのキューに登録されている要求の数。この属性は追加時に増加し、サーバが要求を削除すると減少します。

制限事項 : この属性は、T_DOMAIN:TA_LDBAL 属性が "Y" に設定されている場合のみ返されます。

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に対する、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

TM_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

TM_MIB(5): T_TLISTEN 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_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}

GET 操作は、選択した T_TLISTEN オブジェクトに対する実行時情報を検索します。下に示した状態は、GET 要求に対する応答で返される TA_STATE の意味を示します。

INActive	T_TLISTEN オブジェクトは非アクティブ。
ACTive	T_TLISTEN オブジェクトはアクティブ。

SET:

SET 操作は、このクラスでは許可されません。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() インターフェイスからは利用不可能です。

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および実行時属性を表します。このクラスでは、ユーザはアプリケーション内のログを操作可能、すなわち、作成、削除、移行などを行うことができます。

属性表

TM_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

TM_MIB(5): T_TLOG **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_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 クラスの属性は、すべてローカル属性です。
注 2 T_TLOG クラスの各オブジェクトで、1 つ以上の TA_GRPNO 属性または 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 発生数です。

属性の意味

TA_LMID:*LMID*

トランザクション・ログ論理マシン識別子。

TA_STATE:

GET:{ACTive | INActive | WARmstart}

GET 操作は、選択した T_TLOG オブジェクトに対する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。下に示した状態は、GET 要求に対する応答で返される TA_STATE の意味を示します。

ACTive	トランザクション・ログが存在し、サイトで準備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、関連するアクティブな T_MACHINE オブジェクトに対応します。
INActive	トランザクション・ログが存在しますが、現在非アクティブです。非アクティブな関連 T_MACHINE オブジェクトに対応し、サイトで tlisten(1) プロセスを動作中の場合のみ返されます。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に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・グループがサイトで開始され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際しては ACTive と同等です。

SET:{WARmstart}

SET 操作は、選択した T_TLOG オブジェクトに対するコンフィギュレーション情報と実行時情報を更新します。次に、SET 要求で TA_STATE に設定できる値の意味を説明します。記述されていない状態を設定することはできません。

unset	T_TLOG オブジェクトを変更します。ACTive 時でのみ可能です。正常終了すると、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。正常終了の場合は、オブジェクトは WARmstart 状態になります。

TA_TLOGCOUNT:1 <= *num*

トランザクション・ログ・データ・レコードをカウント、検索、および追加する数 （TA_TLOGDATA)。状態変更を指定した SET 操作では、この属性は無視されます。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・レコード数を示します。

TA_GRPNO も TA_TLOGDATA も指定しない 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、指定され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。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 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のインデックス （TA_GRPNO および TA_TLOGDATA）。

TA_GRPNO:1 <= *num* < 30,000
: トランザクション・コーディネータ・グループ番号。

TA_TLOGDATA:*string*[1..256]
: フォーマット済みトランザクション・ログ・エントリ。この属性値は直接解釈すべきではありません。むしろ、サーバ・グループ移行の一環で、ログ・レコードの移行の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

TM_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

TM_MIB(5): T_TRANSACTION **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_TRANSACTION クラスの属性は、すべてローカル属性です。
注 2 T_TRANSACTION クラスの各オブジェクトでは、1 つ以上の TA_GRPNO 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 発生数です。

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主にサイトにローカルであり、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]
文字列表現にマップされた tpsuspend() から返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]
文字列表現にマップされた tx_info() から返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 オブジェクトに対する実行時情報を検索します。下に示した状態は、GET 要求に対する応答で返される TA_STATE の意味を示します。同一グローバル・トランザクションに付属する別個のオブジェクト（同等のトランザクション識別子）は状態が異なることがあります。一般的に、調整元サイトで示される状態 (TA_COORDLMID) は本当のトランザクション状態です。例外は調整元以外のサイトがトランザクション状態を ABortonlY に遷移させる条件を通知する場合です。この遷移は最終的に調整元サイトに伝達され、トランザクションがロールバックされます。しかし、この変更は即時には調整元サイトに反映されません。状態はすべて、パーミッションの決定に際しては ACTive と同等です。

PREPrepare	トランザクション処理中に 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、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います。
PostABorT	サーバは xa_end (TPFAIL) を呼び出しているが、TMS はまだ 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・アップするために TMS から通知を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 オブジェクトに対する実行時情報を更新します。次に、SET 要求で TA_STATE に設定できる値の意味を説明します。記述されていない状態を設定することはできません。

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 時にのみ可能で、個別のグループの状態を更新するために用いられます（以降の TA_GSTATE 参照）。正常終了すると、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 を中途終了します。ACTive、ABortonlY、および COMcalled 時のみ状態変更が可能です。正常終了の場合は、オブジェクトは ABorteD 状態になります。

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に残っている時間（秒）。この属性値はトランザクション状態 (TA_STATE) が ACTive の場合のみ返されます。

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数。

TA_GRPINDEX:1 <= *num*

このオブジェクトに対応する最初のグループ固有の属性値 (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、選択された T_TRANSACTION オブジェクトに対する実行時情報を検索します。下に示した状態は、GET 要求に対する応答で返される TA_GSTATE の意味を示します。記述されていない状態は返されません。同一グローバル・トランザクションに付属する別個のオブジェクト (同等のトランザクション識別子) は、グループごとに状態が異なる場合があります。一般的に、グループのサイトで示される状態は、グループのトランザクションへの本当の参加状態を示しています。例外は、調整元の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 に設定する場合です。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が、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されたグループでアクティブです。
ABorteD	トランザクションは、ロールバックされるように識別され、指定されたグループに対して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・マネージャで読み取り操作のみが実行されたため、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	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、コミットされる用意ができています。
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、または一致しない場合があります。
HABort	これは、トランザクションの最終結果と一致する場合、または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 フェーズが完了しました。

SET: {HCOmmit|HABort}
SET 操作は、選択した T_TRANSACTION オブジェクト内の発行要求の最初のグループに対する実行時情報を更新します。下に示した状態は、SET 要求で TA_GSTATE に設定できる値の意味を示します。記述されていない状態を設定することはできません。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のみ可能です。

HCOmmit	グループの作業を、指定され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 で TA_STATE が REAdy、かつ指定したグループが調整元サイト上にない場合のみ可能です。正常に終了すると、オブジェクトは HCOmmit 状態になります。
HABort	グループの作業を、指定され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 TA_STATE が REAdy、かつ指定したグループが調整元サイト上にない場合のみ可能です。正常に終了すると、オブジェクトは HCOmmit 状態になります。

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

TM_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)

TM_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_MMDDYY(k)	long	R--R--R--	*mmddyy*	Current date
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
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 クラスの属性は、すべてローカル属性です。
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。戻りレコードを指定マシン上で動作するプロセスが作成した戻りレコードに制限するために使用するのではありません。複数のマシンがネットワーク接続によるファイル・システム経由でログ・ファイルを共有するような場合、「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。この機能条件では、TA_PMID を正規表現のキー・フィールドとして使用した方が便利です。

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]
物理マシンの識別子。

TA_MMDDYY:*mmddyy*
アクセス対象に決定されたユーザ・ログ・ファイルの日付。

TA_STATE:

GET:{ACTive}
GET 操作は、選択した T_ULOG オブジェクトに対する実行時情報を検索します。下に示した状態は、GET 要求に対する応答で返される TA_STATE の意味を示します。

ACTive	返されたオブジェクトが既存のユーザ・ログ・ファイルを指定論理マシンに反映させます。

SET:
SET 操作は、このクラスでは許可されません。

TA_ULOGTIME:*hhmmss*
このオブジェクトで表現されるユーザ・ログ・メッセージの時刻。この属性の値は、時間を 10,000 倍して分を 100 倍した値に加算し、最後に秒を加算して計算されます。キー・フィールドとして実行時、この属性はメッセージに対するアクセスする時間範囲の開始時間を表します。

TA_ENDTIME:*hhmmss*
このユーザ・ログ・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・ファイル内に返却 / 要求されるユーザ・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・ファイルと同様のユーザ・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。トランザクションに対応しないメッセージはこの属性の値として長さが 0 の文字列を検索します。

TA_XID:*string*[1..78]
: tx_info() から返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。トランザクションに対応しないメッセージはこの属性の値として長さが 0 の文字列を検索します。

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 ID。

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
: 特定のアプリケーション関連の識別子。

TA_SEVERITY:*string*[1..30]
: メッセージのレベル（ある場合）。

TA_ULOGCAT:*string*[1..30]
: メッセージ・カタログ中のメッセージを使用した場合には、そのカタログの名前を示します。

TA_ULOGMSGNUM:1 <= *num*
: メッセージ・カタログ中のメッセージを使用した場合には、カタログのメッセージ番号を示します。

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 の場合にのみ行われます。

このクラスの検索には指示が必要です。すなわち TA_LMID を指定しなければなりません。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・レコードを利用できません。

このクラスに対する検索が完全に行われなかった場合、常に 1 の値の TA_MORE が返されます。この値は発行要求に対する詳細情報が利用可能であることを示します。

TM_MIB(5) に関する追加情報

診断

一般に、TM_MIB(5) とのインターフェイス時、2 種類のエラーがユーザに返されます。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()、tpdequeue()) で、いずれかが各関数で定義されたエラーを返します。エラーは該当するマニュアル・ページの記述に従って解釈します。

しかし、要求を満足させることができるシステムサービスに要求が正常に送られ、システムサービスが要求処理に障害があると判断した場合、アプリケーション・レベルのサービス障害の形で異常終了が返されます。このような場合、tpcall() および 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に示す 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) 経由でシステムに転送された要求に対する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仮定します)。

サービス障害が管理要求処理時に発生した場合、FML32 フィールドの TA_STATUS に障害のテキストによる説明が設定され、FML32 フィールドの TA_ERROR に、以下に示すように障害原因が設定されます。エラー・コードはすべてマイナス値です。

[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) マニュアル・ページに記載されています。これらのエラー・コードは、ここで定義する TM_MIB(5) 固有のエラー・コードと排他関係にあります。

以下の診断コードは TA_ERROR に戻され、管理要求が正常に完了したことを示します。これらのコードはプラスの値です。

[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) マニュアル・ページに記載されています。これらのリターン・コードは、ここで定義する TM_MIB(5) 固有のリターン・コードと排他関係にあります。

相互運用性

このマニュアル・ページに定義されたヘッダ・ファイルとフィールド・テーブルは BEA Tuxedo リリース 6.1 以上で使用できます。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。

リリースの異なるサイト (いずれも BEA Tuxedo リリース 6.1 以降) が相互運用を行う場合、旧サイトに関する情報は、そのリリースの MIB リファレンス・ページで定義するとおりにアクセスおよび更新できます。この情報は、新しいリリースで利用可能な情報のサブセットになります。

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、さらにこのマニュアル・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・プラットフォームおよびワークステーション・プラットフォームで利用可能です。

使用例

次のセクションでは、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の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を行うコードについて説明します。ローカル環境に対して妥当な値が必要な場合は変数名を使用します。たとえば、TUXCONFIG は 2 つの要素を持つ文字ポインタの配列で、各要素でマシン上の 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ためには、フィールド・テーブル tpadm が必要です。そのために、以下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

以下のヘッダ・ファイルがあります。

```
#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 を使用するときは、次のようにライブラリを手動でリンクする必要があります。-L$ {TUXDIR} /lib -ltmib -lqm

初期コンフィギュレーション

以下のコードは FML32 バッファを作成し、入力します。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・コードを解釈しています。ここで示す要求はアプリケーションの初期コンフィギュレーションを作成します。

```
/* 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

以下のコードは、上記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ます。以下に示す要求は先に確立したコンフィギュレーションに別のマシンを追加します。

```
/* 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_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、上記の例を参照する */
```

2 つ目のマシンのバックアップ・マスタの作成

既存バッファを再利用して新規にコンフィギュレーション済みの 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);

/* TM_MIB(5) T_DOMAIN 属性の変更を設定 */
Fchg32(ibuf, TA_MASTER, 0, "LMID1,LMID2", 0);

tpadmcall(...); /* 詳細なエラー処理については、上記の例を参照する */
```

2 つのサーバ・グループの追加

バッファを再利用して 2 つの要求を作成します。1 つは 1 つのサーバ・グループ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する要求です。もう一方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だけなので注意して下さい。

```
/* 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);

/* 最初のグループを定義する TM_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、上記の例を参照する */

/* 2 つ目のグループを定義する TM_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、上記の例を参照する */
```

各グループに対して 1 つのサーバを追加

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);

/* 最初のサーバを定義する TM_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、上記の例を参照する */

/* 2 つ目のサーバを定義する TM_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、上記の例を参照する */
```

ルーティング基準の追加

ルーティング基準定義を追加します。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により、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 TM_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、上記の例を参照する */
```

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 TM_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、上記の例を参照する */
```

マスタ・サイト Admin の起動

T_DOMAIN クラス・オブジェクトの状態を ACTIVE に設定してマスタ・サイト管理プロセス (DBBL、BBL、BRIDGE) を起動します。

```
/* 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、上記の例を参照する */
```

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、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 */
 }

 /* 型付きバッファとしてバッファを再初期化 */
 Finit32(ibuf, Fsizeof32(ibuf));
 Finit32(obuf, Fsizeof32(obuf));
```

残りのアプリケーションの起動

アプリケーションの残り部分を起動します。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、各サーバをブート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。この例は tpcall() からのエラー・リターンの処理を示します。

```
/* 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_MACHINE", 0);
 Fchg32(ibuf, TA_STATE, 0, "RAC", 0);

 /* マシンを識別する TM_MIB(5) 属性を設定 */
 Fchg32(ibuf, TA_LMID, 0, "LMID1", 0);

 /* /AdminAPI を呼び出し、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状態の問い合わせを作成します。

```
/* 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 TM_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、上記の例を参照する */
```

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 フラグが使用可能です。

```
/* 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、上記の例を参照する */

/* 次は、マスタ・マシンのアプリケーション・サーバ */
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、上記の例を参照する */

/* アクティブ・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、上記の例を参照する */
```

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

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す。これは、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`
NameManager サービスをスレーブとして指定します。デフォルト値です。

`–M`
NameManager サービスをマスタとして指定します。

`-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 サービスは BEA Tuxedo 固有のサービスです。このサービスは、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します。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・ロールとしてブートできます。-M オプションを指定しない場合は、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 つです。

マスタの 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設定できます。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も設定できます。これらの設定はいずれも、FactoryFinder ドメイン・コンフィギュレーション・ファイル (factory_finder.ini) で指定できます。

factory_finder.ini ファイルの場所は、マスタの 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は、マスタの NameManager は初期化に失敗します。-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 1 つまたは複数の TMFFNAME プロセスをブートすることができます。信頼性を高めるには、できれば異なるマシンで 2 つ以上の NameManager サービスを設定する必要があります。

相互運用性　TMFFNAME サーバは、BEA Tuxedo バージョン 4.0 以上のソフトウェアで動作します。

注意　1 つ以下の NameManager サービスしかアプリケーションの UBBCONFIG (TMFFNAME -N) に設定されていない場合、サーバはブート中に終了し、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 ファイルで設定されていないマスタの NameManager サービスが、スレーブの NameManager サービスの開始時に実行されていると、サーバはブート中に終了し、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することはできません。

TMSYSEVT サーバがアプリケーションの UBBCONFIG ファイルで設定されていない場合に、このサーバが NameManager サービスの開始時に実行されていないと、サーバはブート中に終了し、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 ファイルで設定されていない場合に、FactoryFinder サービスが開始されると、サーバはブート中に終了し、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 Reference』の factory_finder.ini、TMSYSEVT(5)、userlog(3)、UBBCONFIG(5)、および『C++ Programming Reference』の TP フレームワークに関する章

TMIFRSVR

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・リポジトリの 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、サーバはブートできません。

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　『BEA Tuxedo Reference』の ir2idl(1)、UBBCONFIG(5)、および 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す。このサーバは、後で処理するために tpenqueue() を使用して格納されたメッセージを転送します。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することができます。

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、サーバ用にすで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、サーバに関連付けられます。カスタマイズに利用できる、追加コマンド行オプションの一覧を次に示します。

-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 文字以内に制限された文字列です。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のために、tpbegin() で使用されるトランザクション・タイムアウト値を示す場合に使用します。指定されない場合、省略時値は 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示すために、使用します。値が 0 の場合は、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、省略時値は 30 秒です。

-e
: キュー上にメッセージがなければサーバを終了させる場合に使用します。このオプションは、キューに関連付けられたしきい値コマンドと併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 サーバを開始したり停止したりすることができます。

-d

サービスを異常終了 (TPESVCFAIL) 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、再試行しないようトランザクションのロールバック後にキューから削除させ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。

-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。このフラグは、リソース・マネージャと関連のないサーバ・グループへの転送を可能にします。

-f *delay*

サーバに、tpcall を使う代わりにサービスにメッセージを転送させ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、メッセージ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は、優先順位は、コンフィギュレーション・ファイルにおいて定義されているサービスの優先順位か、または省略時値 (50) になります。

メッセージは、1 つのトランザクション内で、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 を伴って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際、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方針が使用されます。デフォルトの配信方針が決まるのは、メッセージへの応答がキューに登録されるときです。つまり、元のメッセージがキューに登録されて、メッセージへの応答がキューに登録される間に、応答キューのデフォルトの配信方針が変更された場合、最終的に応答がキューに登録されたときに配信方針が有効になります。

サービスが異常終了すると、そのキューについて構成されている再試行制限回数の範囲内で、トランザクションはロールバックされ、メッセージは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* 秒の間中断されます。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 オプションが指定されている場合は、サービスが TPESVCFAIL で異常終了し、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、そして、応答メッセージ (および関連する tpurcode) 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条件によっては、TMQFORWARD がキューからメッセージを取り出したり、メッセージを転送したりできないことがあり、このような構成条件では、サーバはブートに失敗します。このために注意すべき構成条件として、次のものがあります。

- SRVGRP では、TMSNAME に TMS_QM が設定されていなければなりません。
- OPENINFO に、関連するデバイスおよびキューの名前を示す値が設定されていなければなりません。
- SERVER エントリは、MSSQ の一部ではあってはなりません。
- REPLYQ には N を設定しなければなりません。
- -q オプションは、コマンド行オプション中で指定しなければなりません。
- サーバは、サービスを宣言してはなりません (たとえば、-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

提供時の TMQFORWARD では、BEA Tuxedo システム に提供されている標準バッファ・タイプの処理に対応します。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・スイッチとともに使用して、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。『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・スイッチおよび必要なサポートされる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。次の例を利用して、詳細を説明します。

```
buildserver -v -o TMQFORWARD -r TUXEDO/QM -f
${TUXDIR}/lib/TMQFORWARD.o -f apptypsw.o
```

この buildserver のオプションは次のとおりです。

-v
: buildserver が冗長モードで動作することを指定します。これは、cc コマンドを標準出力に表示します。

-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、TMQFORWARD を使用することを推奨します。このコマンドのアプリケーション固有バージョンは、$APPDIR にインストールすることができ、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ブートされます。

-r TUXEDO/QM
: このサーバに関連付けられているリソース・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 オプションの第 1 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・エディット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。ここに指定するファイルは、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、複数回指定できます。

サービスを宣言する -s オプション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 # Window NT 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

このサーバは、キューへのメッセージの登録、およびキューからのメッセージの取り出しを、それぞれ tpenqueue() および tpdequeue() を呼び出すプログラムの代わりに行います。アプリケーション管理者は、このサーバを SERVERS セクションでアプリケーション・サーバとして指定することにより、アプリケーションがキューに対するメッセージの登録および取り出しを行えるようにします。

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、サーバ用にすで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、サーバに関連付けられます。カスタマイズに利用できる、追加コマンド行オプションの一覧を次に示します。

-t *timeout*
: トランザクション・モードにないキュー操作で使用するタイムアウトを示すために使用します。「トランザクション・モードにないキュー操作」とは、たとえば、tpenqueue() または tpdequeue() が、トランザクション・モードにない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る場合のことです。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。この場合、この値に基づいて、操作がタイムアウトしてエラーが要求者に返されるためです。指定されない場合、省略時値は 30 秒です。

TMQUEUE サーバは、アプリケーションの一部としてブートされ、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条件によっては、TMQUEUE がキューにメッセージを登録したり、キューからメッセージを取り出したりできないことがあり、このようなコンフィギュレーション条件では、TMQUEUE はブート時に失敗します。SRVGRP では、TMSNAME に TMS_QM が設定されていなければならず、OPENINFO に、関連するデバイスおよびキュー・スペースの名前を示す値が設定されていなければなりません。

メッセージ要求時のキューの名前

関数 tpenqueue() および tpdequeue() 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。省略時には、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だけで十分ですが、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という場合もあります。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の -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へ、メッセージの宛先が指定されます (なお、キュー・スペースは、OPENINFO 文字列に基づいて、GROUP につき 1 つしか存在できないことに注意してください)。

アプリケーション・バッファ・タイプの処理

提供時の TMQUEUE では、BEA Tuxedo システムに提供されている標準バッファ・タイプの処理に対応します。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。『BEA Tuxedo /Q コンポーネント』を参照してください。

buildserver で説明されているカスタマイズは、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・スイッチおよび必要なサポートされる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。次の例を利用して、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
```

この buildserver のオプションは次のとおりです。

-v
: buildserver が冗長モードで動作することを指定します。これは、cc コマンドを標準出力に表示します。

-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、TMQUEUE を使用することを推奨します。

-s *qspacename,qspacename* :TMQUEUE
: サーバのブート時に宣言できるサービス名を指定します (servopts(5) を参照)。このサーバでは、サービス名は、要求を出す先となるキュー・スペースの名前の別名として使用されます。カンマの間に空白は使用できません。関数名 TMQUEUE の前にはコロンを付けます。-s オプションは、複数回使用できます。

-r TUXEDO/QM
: このサーバに関連付けられているリソース・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 オプションの第 1 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・エディット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。ここに指定するファイルは、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、複数回指定できます。

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 Release 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・スペース名の 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・スペースが利用できます。2 つの TMQUEUE サーバはどちら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、システム・エラーや潜在的なエラー条件に関するイベント通知を処理するための BEA Tuxedo システムが提供するサーバです。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動作をスタートさせることができます。

フィルタや通知に関するルールは、制御ファイル *control-file* に記憶されます。このファイルは、デフォルトでは ${APPDIR}/tmsysevt.dat という名前になります。*control-file* の構文は EVENT_MIB(5) で定義されます。特に、EVENT_MIB のクラス属性は、通知に関するルールの範囲内でサブスクリプションをアクティブな状態にするように設定できます。

1 つまたは複数の二次的な TMSYSEVT プロセス (すなわちセカンダリ・サーバ) を起動し、使用効率を上げることができます。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ライン・オプションの -S を利用して起動する必要があります。

EVENT_MIB(5) のコンフィギュレーションを更新する際には、プライマリの 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、各自プライマリ・サーバの制御ファイルの内容が更新されていないかどうかをポーリングを通じてチェックし、必要に応じて自分の (ローカルの) の制御ファイルの内容を更新します。ポーリングの間隔は -p オプションでセットできますが、デフォルトの設定では 30 秒間隔になります。

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

一次 TMSYSEVT サーバを別のマシンに移行するためには、システムの管理者は現在の制御ファイルのコピーを渡す必要があります。二次の各 TMSYSEVT サーバでは、自動的に新しいコピーが保持されます。

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セットすることができます。

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)、TMUSREVT(5)

tmtrace(5)

名前

tmtrace— 実行時トレーシング機能

機能説明

実行時トレーシング機能を使用すると、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。

実行時トレーシングは、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 メッセージの到達、またはトランザクションの開始があります。

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、*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は、*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ー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、およびトリガーは、*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されます。

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 トリガーおよび undye トリガー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、次のとおりです。

`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 コールが処理される際の `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 インターフェイスに対する黙示的コールのトレース・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コール、および管理を目的として呼び出される内部コール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 インターフェイスに対するすべてのコール (明示的なコールと黙示的なコールの両方) 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・マネージャとの間のインターフェイス) に対するすべてのコールのトレース・ポイント。

`trace`
: メッセージのダイ設定を含む、トレーシング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`*] の構文で指定する文字列です。ここで、*`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。

ヌル文字列も正規のトレース指定です。ヌル文字列は、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 および `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、フィルタ指定は次の構文を使用します。

[{ + | - }] [*`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 * を使用して、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取り出すことを指定します。+ 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。

空のフィルタの場合は、選択するカテゴリがないという意味で、トレーシングを使用不可にしています。

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ー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・レコードが送信され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、レシーバ指定は次の構文を使用します。

[*/ regular-expression /*] *receiver*

ここで、オプションの正規式を使用して、フィルタを通るトレース・ポイントのサブセットを選択することもできます。正規式は、トレース・レコードと一致します。空のレシーバ指定も正当であり、この場合、トレース・レコードは発行できません。

現在、*receiver* の唯一有効な値は次のとおりです。

`ulog`
: トレース・レコードをユーザ・ログへ発行します。

トリガー指定

トリガーは、トレース・レコードが発行された後に実行されるオプションのアクションです。多くて 1 つのアクションが、フィルタを渡す各トレース・レコードのために実行されます。

トレース指定の 3 つめの部分であり、オプションのトリガー指定は次の構文を使用します。

[/ regular-expression /] action

ここで、オプションの正規式を使用して、フィルタを通るトレース・ポイントのサブセットのためだけに実行するように、トリガーを制限できます。正規式は、トレース・レコードと一致します。

使用可能なアクションは、次のとおりです。

`abort`
: `abort()` を呼び出し、プロセスを終了させます

`ulog(`*message*`)`
: ユーザ・ログに *message* を書き込みます。

`system(`*`command`*`)`
: `system`(3) を利用して *`command`* を実行します (これは Windows クライアントに対して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`*`)`
: トレース指定を指定の値 *`trace-spec`* に設定し直します。

`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を含みます。

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することはできません。

使用例

クライアントをトレースするには、そのクライアントのために、アプリケーション・サーバによって作成されたすべての ATMI 呼び出しをトレースするほかに、クライアントの環境に次のように設定しエクスポートします。`TMTRACE=on` この指定は、クライアント内のすべてのトレース・ポイントを記録し、メッセージのダイ設定をオンにします。クライアントのため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、自動的にすべての ATMI トレース・ポイントを記録します。

前述の例で、クライアントからのサービス要求をトレースして、クライアントからの出力のトレースを、それらの要求についての最低限の情報に制限するには、クライアントの環境に次のように設定し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
```

この指定は、クライアントで行われるすべての tpacall 呼び出しを記録し、メッセージのダイ設定をオンにします。クライアントのため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、自動的にすべての ATMI トレース・ポイントを記録します。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、クライアントのため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、参加しているすべてのマシン上にあるサーバ *ENVFILE* 内に次のように設定します。

```
TMTRACE=atmi:/tpservice/ulog
```

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シングを使用可能にするには、すべてのクライアントの環境および参加しているすべてのマシン上の、マシン *ENVFILE* 内に次のように設定しエクスポートします。

```
TMTRACE=*:ulog:dye
```

この設定では、BBL および 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、ブート後に ATMI のトレーシングをオンにするには、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* 内に 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 である、実行中のサーバ・プロセスを、tpreturn を実行したときに中止するには、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、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する、BEA Tuxedo システム提供のサーバです。

このファイルは、デフォルトでは ${APPDIR}/tmusrevt.dat という名前になります。*control-file* の構文は EVENT_MIB(5) で定義されます。特に、EVENT_MIB のクラス属性は、通知に関するルールの範囲内でサブスクリプションをアクティブな状態にするように設定できます。

1 つまたは複数の二次的な TMUSREVT プロセス (すなわちセカンダリ・サーバ) を起動し、使用効率を上げることができます。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ライン・オプションの -S を利用して起動する必要があります。

EVENT_MIB(5) のコンフィギュレーションを更新する際には、プライマリの 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、各自プライマリ・サーバの制御ファイルの内容が更新されていないかどうかをポーリングを通じてチェックし、必要に応じて自分の (ローカルの) の制御ファイルの内容を更新します。ポーリングの間隔は -p オプションでセットできますが、デフォルトの設定では 30 秒間隔になります。

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

プライマリの TMUSREVT サーバを別のマシンに移行する際には、システムの管理者は制御ファイルの現在のコピーを提供する必要があります。セカンダリの各 TMUSREVT サーバは、自動的に最新のコピーを保持します。

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 サーバの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セットすることができます。

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)、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* に拡張されます。*lvalue* には、複数の BEA Tuxedo システム ライブラリ・ルーチンが、正のエラー番号 (>0) を設定します。tperrno はオブジェクトの識別子である必要はなく、機能呼び出しの結果、修正可能な *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・ルーチンのマニュアル・ページには、各ルーチンのエラー条件およびそのコンテキストにおけるエラーの意味をリストしています。リスト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はありません。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 の設定を指定している場合です。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。

各エラーの一般的な意味を次に示します。

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、あるいは、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
: ブロッキング条件が存在しており、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
: シグナルが受信されましたが、TPSIGRSTRT は、指定されませんでした。

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判断により、トランザクションの代わりに行われる処理の一部はコミットされ、一部は中途終了 (アボート) されました。

TPEINVAL
: 無効な引数がありました。

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コネクション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、すで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、またはそれが正しいサービス型でないために *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。その理由は、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、または正しいアプリケーションのパスワードを提供されていないためです。

TPEPROTO
: ライブラリ・ルーチンは、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 からのクライアントです。

TPERMERR
: リソース・マネージャは、オープンまたはクローズに失敗しました。

TPESVCERR
: サービス・ルーチンは、tpreturn() または tpforward() のいずれかにおいてエラーを見つけました (たとえば、不正な引数が渡されました)。

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・レベルの問題です。

TPESYSTEM
: BEA Tuxedo システム のエラーが発生しました。

TPETIME
: タイムアウトが発生しました。

TPETRAN
: 呼び出し側がトランザクション・モードになりません。

使用方法

ルーチンには、エラーの戻り値がないものもあります。tperrno にゼロを設定するルーチンはないため、アプリケーションは、tperrno にゼロを設定し、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 を検査することができます。

関連項目

個々の BEA Tuxedo ライブラリ・ルーチンのエラー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

下記は、tpurcode の使用例です。

アプリケーション・サービスで rcode により myval の値を返す場合、次のようになります。

```
.
.
.
tpreturn(TPSUCCESS, myval, rqst->data, 0L, 0);
.
.
.
```

クライアント・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" の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 ファイルのための変数
- ファイル・システム、および TLOG 変数
- /WS 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
: 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) を参照のこと。

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合わせる必要があります。

NLSPATH
: メッセージ・カタログのパス名の指定。指定がない場合デフォルトのパス名を使用する。nlpaths(5) を参照してください。

PAGER

qmadmin(1)、tmadmin(1) でページング出力をする時に使うページング・コマンドを指定する。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)) は無効になる。

PATH

実行形式のために検索するパス名を含む。

SHELL

BEA Tuxedo システム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、その端末のタイプを指定します。

TMPDIR

一時ファイルが書き込まれるディレクトリのパス名。また、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( ) が呼び出されると、TMPDIR 変数は BEA Tuxedo システムで無視されます。

BEA Tuxedo リリース 6.5 以前のバージョンでは、サービス・キューがいっぱいになりメッセージ・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。このコードは、BEA Tuxedo リリース 7.1 以降でも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 (この変数が設定されている場合) によって指定されるディレクトリのパス名になり、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 をセットする必要があります。

これらの変数についての詳細は、UNIX システムのマニュアル environ(5) の項目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し、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
: システム・クライアント用のパスワードを指定するのに使用され、セキュリティが稼動している状態で、このシステム・クライアントはアプリケーションパスワードを求める。変数のパスワードを設定することによって、そのパスワードは、手動によるエントリではなく、スクリプトから供給される。

ENVFILE
: tmloadcf(1) で使用される変数。通例、他の BEA Tuxedo システムの環境変数の設定を含む。この変数はシステムにより自動的に設定される。

TLOGDEVICE
: トランザクション・ログ用パス名。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 と必ず同一であること。

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。

ULOGPFX
: 中央イベント・ログのファイル名の前に付ける接頭辞。デフォルトは ULOG。

TUXDIR
: BEA Tuxedo システムのソフトウェアがインストールされる基本ディレクトリの指定。

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 ファイルのための変数

以下の環境変数は、FML および VIEWS で使用されます。

FIELDTBLS
: カンマで区切られる、フィールド・テーブル・ファイルの一覧

VIEWFILES
: カンマで区切られる、バイナリ・VIEW ファイルの一覧

FLDTBLDIR
: コロンで区切られる、FIELDTBLS ファイルを検索するディレクトリの一覧

VIEWDIR
: コロンで区切られる、VIEWFILES ファイルを検索するディレクトリの一覧

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

下記の変数は、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。

FSMAXUPDATE
: 更新リストのサイズを設定、および更新回数の最大値を設定。

FSMSGREP
: メッセージを繰り返す間隔の設定。

FSOFFSET
: 汎用デバイス・リスト内のオフセットの指定。

/WS 変数

以下の変数は、/WS クライアント・マシンで使用します。

WSALLOWPRE71
: BEA Tuxedo 7.1 以降のソフトウェアが稼動するワークステーション・マシンに対して、リリース 7.1 より前の BEA Tuxedo アプリケーションとの相互運用を認めるかどうかを決定します。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
: /WS 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

以下の環境変数が BEA Tuxedo/Q では使用されます。

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。

この変数の詳細については、『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の指定。

ALTCFLAGS
: COBOL のコンパイラに渡すフラグ。

注記 Windows システムでは、ALTCC と ALTCFLAGS 環境変数は使用されません。これらの変数を設定すると、予想外の結果が生じることがあります。最初 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の指定。

COBOPT
: COBOL コンパイラ用のコマンド行引数を含む。

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) の数の指定。デフォルトは「3」。

PMID
: MP モードでの物理マシン ID の指定に使用可能。MP モードで、物理マシン ID を指定するために使用できます。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) ログ用パス名の設定。

TMCMPLIMIT
: メッセージを圧縮すべきかどうかの指定、およびローカル・メッセージとリモート・メッセージのしきい値の設定。変数の構文は、

  TMCMPLIMIT=[remote_threshold[,local_threshold]]

  しきい値は、0 から MAXLONG の範囲内の数字。この数字はデータ圧縮をするメッセージの最少サイズをバイトで設定する。

TMCMPPRFM
: プロセスの圧縮レベルの設定。整数 1 から 9 が有効値。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

TMNETLOAD
: ネットワーク上のロード・バランシングの確立。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。この変数によりローカル・サービスが強制的に使用される。

TMNOTHREADS
: マルチスレッド化された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、マルチスレッド化された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。有効な値は 0 ～ 32,767 の整数です。この変数に設定した値は、UBBCONFIG ファイルに指定される値より優先されます。

UIMMEDSIGS
: 信号の遅れを無視するために、この変数に "Y" を設定する。

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 システムが提供するバッファ・タイプの記述

形式

省略時のバッファ・タイプ・スイッチ

```
/*
 * 以下の定義は
 * $TUXDIR/lib/tmtypesw.c に指定されています。
 */

#include "tmtypes.h"

/*
 * バッファ・タイプ・スイッチの初期設定
 */

struct tmtype_sw_t tm_typesw[] = {
{
      "CARRAY",     /* type */
      "*",          /* subtype */
      0             /* dfltsize */},
} ,
{
      "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 */},
{
      "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 */
},
{
        "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 */
},
{
        /* 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 */
},
{
    /* 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 */
},
{
        "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 */
},
{
        "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 */
},
{
     "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 */
},
{
""
}
};

struct tmtype_sw_t _TM_FAR *
_TMDLLENTRY
_tmtypeswaddr(void)
{
  return(tm_typesw);
}
```

機能説明

次の表は、BEA Tuxedo システムが提供する 10 種類のバッファ・タイプの一覧です。

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
XML	XML ドキュメント用のバッファ
VIEW	C 構造体または FML VIEW
X_OCTET	CARRAY に等しいもので、XATMI 互換性用に提供されています。
X_C_TYPE	VIEW に等しいもので、XATMI 互換性用に提供されています。
X_ COMMON	VIEW に等しいもので、XATMI 互換性用に提供されています。
FML32	32 ビット識別子および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

すべての VIEW、X_C_TYPE、および X_COMMON バッファは同じルーチンのセットで処理され、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ように注意してください。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 内に配布され、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もつシステムでは、$TUXDIR/lib の下に libbuft.so.50 という新しいインスタンスを作成およびインストールします。WSH などの BEA Tuxedo システム・プロセスを含む、すべてのプロセスは、再コンパイルしなくても、新しいタイプ・スイッチへのアクセス権を自動的にもつことになります。Windows のワークステーションでは、バッファ・タイプのスイッチのための共有オブジェクトは、WBUFT.DLL となります。これは、$TUXDIR¥bin に格納されている必要があります。

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) など特殊なコマンドを使用して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。また、アプリケーション独自のバッファ・タイプを定義するときに、アプリケーションが使用できる BEA Tuxedo システム提供の省略時のルーチン (たとえば _finit()) についての説明も記載されています。

システムが提供する _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 – タイプ・スイッチのインストレーション
$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を識別するスイッチ

形式　　バッファ・タイプの構造

```
/*
 * 以下の定義は、$TUXDIR/include/tmtypes.h にあります。
 */
#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のサイズ */
   /* 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));
   /* 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, 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ます */
   void (_TMDLLENTRY *reserved[9]) _((void));
};
```

```
/*
 * アプリケーション・タイプ・スイッチ・ポインタ
 * テーブル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それぞれ、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、tm_typesw 配列内に対応するエントリが必要です。BEA Tuxedo が提供するバッファ・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には、$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しなければならないルーチンのセマンティクスは、buffer(3c) に説明されています。

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・スイッチの具体例

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環境設定するには、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。UBBCONFIG ファイル全体に関する追加情報については、500 ページ「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入れ、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り、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 (あるいはその両方) の集まりへのアクセスを提供するインターフェイス (および関連するソフトウェア) 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() によって開始され、tpcommit() または tpabort() によって終了されます。

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もつ必要があります。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していません。

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、ホスト固有のマニュアル・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・ファイルのフォーマット

UBBCONFIG ファイルは、9 つの指定セクションから構成されます。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、RESOURCES、MACHINES、GROUPS、NETGROUPS、NETWORK、SERVERS、SERVICES、INTERFACES、および ROUTING です。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。等号記号 (=) の両側で、空白類 (スペースやタブ文字) を使用することができます。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、セクションごとに説明します。

予約されているワード DEFAULT: で始まる行には、その行があるセクションのその行の後のすべての行に適用されるパラメータ指定が含まれています。省略時の指定は 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。これらの行の形式は次の通りです。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、ほかの DEFAULT: 行によって再設定されるか、またはセクションが終わるまで有効です。これらの値は、DEFAULT: 行以外の行のオプション・パラメータによって上書きされる場合もあります。DEFAULT: 行以外の場合は、省略時設定に戻ります。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 (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 などのように BEA Tuxedo システムで既に認識されている文字列値) の場合、一般には標準 C 規則が適用されます。標準 C の *identifier* では、先頭にアルファベットまたは下線を使用し、英数字または下線以外の文字を使用することはできません。識別子に使用できる最大文字数は 30 文字です (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文字列です。ユーザ定義の文字列では、最後のヌル文字を除き、最大 78 文字まで使用できます。この規則には、以下のような例外事項があります。

- CLOPT、BUFTYPE、OPENINFO、および CLOSEINFO パラメータでは、256 文字まで使用できます。
- SEC_PRINCIPAL_NAME、SEC_PRINCIPAL_LOCATION、および SEC_PRINCIPAL_PASSVAR パラメータでは、最後のヌルを除き 511 文字まで使用できます。
- RANGES パラメータでは、2048 文字まで使用できます。ただし、Domains の場合は 1024 文字までしか使用できません。

ROUTING 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
"\"" は二重引用符に変換
"\n" は復帰改行に変換
"\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。これにより警告メッセージを生成します。

"#" はコメントを示します。復帰改行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

サーバ数やサービス領域に存在できるサービス数などのシステム規模の指定を行うセクション。このセクションの行は次のような形式をとります。RESOURCES セクションの行の形式は、*KEYWORD value* です。*KEYWORD* はパラメータの名前で、*value* はそれに対応する値です。有効な *KEYWORDs* には以下のものがあります。

IPCKEY *numeric_value*

BEA Tuxedo システムの掲示板における IPC キー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「指名」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・キューを指します。また、このキーは、マルチプロセッサ全体の掲示板などの資源の名前を取り出す基準としても使用されます。IPCKEY は、32,768 より大きく、262,143 未満の値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は、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ブート時に、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 はこの代替位置でブートされ、その位置にある代替 TUXCONFIG ファイルが使用されます。LMID の値は両方とも、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において、MASTER と BACKUP は常に全てのマシンよりも上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." の間には強制はされていません。

DOMAINID *string_value*

ドメイン ID 文字列を指定します。このパラメータの指定がない場合は、"" 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) 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 となる値がとられます。このパラメータの *RESOURCES* セクションの値は、プロセッサごとに MACHINES セクションで変更することができます。

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 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 がとられます。このパラメータの RESOURCES セクションの値は、プロセッサごとに MACHINES セクションで変更することができます。

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 (すなわち、0600 のような 8 進数値) で、プロセスに対する読み取り書き込みパーミッションを指定するために使用します。このパラメータの指定がないと、IPC 構造体に対するパーミッションは 0666 (1 人のユーザ、1 つのグループ、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) という省略時設定となります。値は 0001 以上 0777 以下の範囲で指定できます。このパラメータの RESOURCES セクションの値は、プロセッサごとに MACHINES セクションで変更することができます。

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。指定されていない場合、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 セクションで変更することができます。

BBL、restartsrv、cleanupsrv、tmshutdown()、tmadmin() など、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てください。システム提供のサーバの例としては、AUTHSVR、TMQUEUE、TMQFORWARD、TMUSREVT、TMSYSEVT、TMS 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 と 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。特に、あるマシンの MAXACCESSERS とアプリケーションで既に動作しているマシン (複数のマシンが動作している場合もある) の MAXACCESSERS の合計が、アプリケーションの MAXSERVERS とユーザ・ライセンス数の合計より大きい場合、その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 の合計数は、アプリケーションの MAXSERVERS とユーザ・ライセンス数の合計数以下である必要があります。

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 点だけです。すべてのユーザ・ライセンスが使用中の場合、新しいクライアントがアプリケーションに参加することはできません。

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、8192 未満でなければなりません。指定されない場合、デフォルトの 50 が設定されます。

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。システム提供サーバの例としては、AUTHSVR、TMQUEUE、TMQFORWARD、TMUSREVT、TMSYSEVT、TMS (GROUPS セクションの TMSNAME パラメータを参照)、TMS_QM、GWTDOMAIN、および WSL があります。

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 の数に入れてください。

MAXSERVICES *numeric_value*

掲示板のサービス・テーブルに合う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。指定されない場合、デフォルトの 100 が設定されます。

MAXGROUPS *numeric_value*

掲示板のグループ・テーブルに合う構成サーバ・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。指定されない場合、デフォルトの 100 が設定されます。

MAXNETGROUPS *numeric_value*
: TUXCONFIG ファイルの NETWORK セクションの中で適合しうるように環境設定されたネットワーク・グループの最大数値を指定します。この値は 1 以上、8192 未満でなければなりません。指定されない場合、デフォルトの 8 が設定されます。

MAXMACHINES *numeric_value*
: 掲示板のマシン・テーブルに合う構成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。この値の指定がない場合は、省略時の値の 256 がとられます。

MAXQUEUES *numeric_value*
: 掲示板のキュー・テーブルに合う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。この値の指定がない場合は、値は MAXSERVERS の設定値に設定されます。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 の設定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。定義可能な最大のグループ ID は、TA_MAXACLGROUPS - 1 です。この値は、1 以上で 16,384 以下でなければなりません。指定されていない場合、16,384 がデフォルト値になります。

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)。

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 の指定がない場合は、省略時設定の Y がとられます。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。

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 の場合は、TP_COMMIT_CONTROL 特性は、TP_CMT_LOGGED に初期設定され、*value* が COMPLETE の場合は、TP_COMMIT_CONTROL 特性は、TP_CMT_COMPLETE に初期設定されます。CMTRET の指定がない場合は、省略時設定の COMPLETE がとられます。この特性の設定の詳細については、BEA Tuxedo システム ATMI 関数である tpscmt() を参照してください。

OPTIONS {[LAN | MIGRATE | NO_XA | NO_AA],*}

使用するオプションを指定します。2 つのオプションを指定する場合は、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。

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獲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 は、BEA Tuxedo システム・ライブラリが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利用して内部テーブルにアクセスできることを指定します。PROTECTED は、内部テーブルを共有メモリを介して BEA Tuxedo システム・ライブラリがアクセスできるときに、それらのテーブルの共有メモリを BEA Tuxedo システム・ライブラリの外部からはアクセスできないようにすることを指定します。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利用してアプリケーション・プロセスが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、または NO_OVERRIDE だけを伴って指定する場合、省略時のモードは FASTPATH となります。

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。あるスレッドが BEA Tuxedo コードを実行中、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・コードを実行することができるためです。BEA Tuxedo システムでは、このような状況を防止することはできません。

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。指定されていない場合、このパラメータのデフォルト値は NONE になります。"APP_PW" という値は、アプリケーションのパスワード・セキュリティ機能を使用することを意味します (クライアントは初期化時にアプリケーション・パスワードを渡す必要があります)。"APP_PW" をセットすると、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は与えられません。

AUTHSVC *string_value*
システムに結合している各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。このパラメータは、SECURITY の識別コードが "USER_AUTH"、"ACL"、または "MANDATORY_ACL" のいずれかにセットされていることを必要とします (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 文字以下とします。SECURITY レベルが "USER_AUTH" のときは、デフォルトのサービス名は (指定しなかった場合は) "AUTHSVC" となります。SECURITY レベルが ACL または MANDATORY_ACL の場合、サービス名が指定されていないときには、..AUTHSVC がデフォルトのサービス名になります。

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 AUTHSVC として宣言され、SECURITY が ACL または MANDATORY_ACL に設定されているときには ..AUTHSVC として宣言されます。AUTHSVC と ..AUTHSVC は、同じ認証サービスを指します。

文字列値の 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。

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。指定されない場合、デフォルトの 100 が設定されます。このパラメータの RESOURCES セクションの値は、マシンごとに MACHINES セクションで変更することができます。

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。指定されていない場合、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。1 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 セクションで変更することができます。

MAXBUFTYPE *numeric_value*

掲示板のバッファ・タイプ・テーブルに収めることができるバッファ・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。指定されない場合、デフォルトの 16 が設定されます。

MAXBUFSTYPE *numeric_value*

掲示板のバッファ・サブタイプ・テーブルに収めることのできるバッファ・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。指定がなければ、省略時の値 32 がとられます。

MAXDRT *numeric_value*

構成データ依存基準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。この値の指定がない場合、省略時の値は ROUTING セクションの構成に入力された値です。

MAXRFT *numeric_value*

データ依存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。この値の指定がない場合、省略時の値は ROUTING セクションの構成に入力された値です。

MAXRTDATA *numeric_value*

データ依存ルーティング範囲文字列の最大文字列プール・サイズを指定します。この値は 0 以上、32,761 未満でなければなりません。この値の指定がない場合、省略時の値は ROUTING セクションの構成に入力された値です。

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時間間隔 (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します。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・タイムアウト値の単位に影響します。SANITYSCAN、BBLQUERY、DBBLWAIT、BLOCKTIME の各パラメータの値は、この単位の倍数で、一定の時間で動作するように設定されているシステム内部のその他の操作に対するものです。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。省略時設定は 10 秒です。

SANITYSCAN *numeric_value*

この設定値を SCANUNIT に乗じた値が、システムの正常性チェックを行う間隔となります。値 SCANUNIT 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、サーバにも行われます。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 に対して送信して、BBL が OK であることを示します。

DBBLWAIT *numeric_value*

DBBL がタイムアウト前にそのすべての 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 はその BBL に要求を転送するたびに、リクエスタへの応答の前にそれらの全 BBL から肯定応答が返されるまで待機します。このオプションは、動作不能の、あるいは異常な BBL を適宜通知するために使用することができます。DBBLWAIT は 0 より大きくなければなりません。このパラメータの指定がない場合は、(SCANUNIT * DBBLWAIT) が SCANUNIT より大きくなるか、または 20 秒となるように省略時の値が設定されます。

BBLQUERY *numeric_value*

すべての BBL の DBBL により実行される状態チェック間の間隔を SCANUNIT の乗数で指定します。DBBL は、すべての BBL が、BBLQUERY サイクル内に報告したことを、チェックして確認します。BBL からの報告がなかった場合は、DBBL はその BBL にメッセージを送り、状態を照会します。応答がない場合、BBL は分断されます。BBLQUERY 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 秒になるように省略時の値が設定されます。

BLOCKTIME *numeric_value*

ブロッキング呼び出し (たとえば、応答の受信) の後、タイムアウトするまでの時間を SCANUNIT の乗数で設定します。BLOCKTIME 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は、(SCANUNIT * BLOCKTIME) が約 60 秒になるように省略時の値が設定されます。

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省略時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することができます。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 という値は、ディップ・イン方式の通知検知手段を使用することを示します。これは、ATMI コール中ではシステムはクライアント・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 コール中での検出ポイントは、システムによっては定義されず、ブロック中のシステム・コールがディップ・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のデフォルト設定です。

SIGNAL という値は、シグナル・ベースの通知検出手段を使用することを示します。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キャッチ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、管理システム・プロセスによって行われ、アプリケーション・プロセスが行うのではありません。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動作しているネイティブ・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は、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 は、THREAD 通知検出を使用することを指定します。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 通知を使用することはできません。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 という値は、デフォルト設定では通知メッセージがアプリケーションのクライアントに無視されるように指定します。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。

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な値は SIGUSR1 および SIGUSR2 で、省略時設定は 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 を指定することができます。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。

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、セキュリティの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き、最大 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 以降を実行するシステム・プロセスに対して復号化キーをオープンする処理で指定します。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き、最大 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き、最大 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、復号化キーのそれぞれのパスワードを入力する必要があります。パスワードの入力を求めるプロンプトは、tmloadcf(1) によって表示されます。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の範囲 (秒) を指定します。指定されていない場合、3600 秒 (1 時間) がデフォルト値になります。このパラメータは、BEA Tuxedo 7.1 以降のソフトウェアが稼動す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の範囲 (秒) を指定します。指定されていない場合、604800 秒 (1 週間) がデフォルト値になります。このパラメータは、BEA Tuxedo 7.1 以降のソフトウェアが稼動す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。指定されていない場合、N がデフォルト値になります。このパラメータは、BEA Tuxedo 7.1 以降のソフトウェアが稼動す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。指定されていない場合、N がデフォルト値になります。このパラメータは、BEA Tuxedo 7.1 以降のソフトウェアが稼動す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

コンフィギュレーションの物理マシンの論理名を指定する MACHINES セクション。また、このセクションは指定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、アプリケーションが使用する各物理プロセッサに対する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。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 NT 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ますが、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 セクション内で一意である必要があります。

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入れることはできません。また 30 文字までで指定する必要があります。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で LMID が必要です。

以下のパラメータは必須です。

TUXCONFIG = *string_value*

このマシン上のバイナリ TUXCONFIG ファイルがあるファイルまたはデバイスの絶対パス名。この文字列値の最大長さは 64 文字です。管理者が保守する必要がある TUXCONFIG ファイルは、MASTER マシン上で TUXCONFIG によりポイントされる 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、システムのブート時に自動的に MASTER マシンと同期されます。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は、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。

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*

このマシン上の BEA Tuxedo システム・ソフトウェアがあるディレクトリの絶対パス名。このパラメータは各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は、より新しいリリースの 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*

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・ディレクトリです。この絶対パス名は、他の絶対パス名のコロンで分けられたリストが後につくことが任意にあります。SECURITY が設定されているコンフィギュレーションにおいて、それぞれのアプリケーションは明確な 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。

以下は、オプション・パラメータです。

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。この値の有効範囲は 0 から 2147483647 までです。このパラメータの指定がない場合は、RESOURCES セクションで指定した値が省略時の値となります。

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。この値の有効範囲は 0 から 2147483647 までです。このパラメータの指定がない場合は、RESOURCES セクションで指定した値が省略時の値となります。

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、プロセスの読み取り書き込みパーミッションを通常の UNIX システム方式 (すなわち、0600 などの 8 進数で) で指定する場合に使用します。値は 0001 以上 0777 以下の範囲で指定できます。このパラメータの指定がない場合は、RESOURCES セクションで指定した値が省略時の値となります。

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。指定されていない場合、RESOURCES セクションで指定される MAXACCESSERS の値がデフォルト値になります。

  BBL、restartsrv、cleanupsrv、tmshutdown()、tmadmin() など、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てください。システム提供のサーバの例としては、AUTHSVR、TMQUEUE、TMQFORWARD、TMUSREVT、TMSYSEVT、TMS 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) の両方をこの値に含める必要があります。

BEA Tuxedo リリース 7.1 より前、つまりリリース 6.5 以前では、アプリケーションの MAXACCESSERS と 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と関連付けられていました。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 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 点だけです。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。指定されていない場合、値は 0 以上で 32,768 未満でなければなりません。この値の指定がない場合、デフォルトの値は 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 に対して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によって使用することはできません。この値を MAXACCESSERS の値よりも大きい値に設定した場合は、エラーになります。

MAXWSCLIENTS パラメータが使用されるのは、BEA Tuxedo System Workstation 機能を使用する場合だけです。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) 資源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中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 以上で 32,000 以下でなければなりません。デフォルト値は 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。指定されていない場合、RESOURCES セクションで指定される MAXCONV の値がデフォルト値になります。1 サーバ当たりの同時の会話の最大数は、64 です。

MAXPENDINGBYTES = *number*
: bridge 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 以上 32,768 未満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、RESOURCES セクションで指定した値が省略時の値となります。

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、15 文字以下の任意の文字列値に設定することができます。2 つのマシンが同じ TYPE 値をもつ場合は、それらのマシン間でデータを送信する場合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することができます。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使用される場合または、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する必要があります。このパラメータを指定しない場合は、値を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が省略時の値となります。

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メッセージ・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。この値の指定がない場合、このパラメータの省略時の値は "MAXLONG、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の計算時に、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。この値の指定がない場合、省略時の値は 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*

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。このパラメータの指定がない場合は、このマシンは TLOG をもたないとみなされます。この文字列値の最大長さは 64 文字です。

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、数値オフセット (デバイス先頭からの) 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。このパラメータの指定がない場合は、"TLOG" が省略時設定となります。1 つの TLOGDEVICE に複数の TLOG がある場合は、それぞれが一意の名前をもってい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とも異なる名前をもっていなければなりません。このパラメータは 30 文字以下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 以下でなければなりません。指定されない場合、デフォルトの 100 ページが設定されます。

ULOGPFX = *string_value*

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。指定マシンの ULOGPFX の値を使用して、こ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 を格納する BEA Tuxedo ファイル・システムの開始点までの、数値オフセット (デバイス先頭からの) 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。省略時のオフセットは 0 です。TUXOFFSET の値は、ゼロでなければ、マシン上ですべてのサーバがブートする環境に置かれます。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*

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* の先頭には下線 (_) または英文字を指定し、全体は下線または英数字のみで構成します。*value* では、${*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 (\) を使用すると、バックスラッシュ自体およびドル記号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、拡張された *value* が環境に組み込まれます。

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 上にないもの) 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 の値より優先されます。

PATH および 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 および LD_LIBRARY_PATH を使用して、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が検索されます。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、セキュリティの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き、最大 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、このマシン上で実行される 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 以降を実行するシステム・プロセスに対して復号化キーをオープンする処理に関係するパラメータです。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き、最大 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き、最大 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、各復号化キーのそれぞれのパスワードを入力する必要があります。パスワードの入力を求めるプロンプトは、tmloadcf(1) によって表示されます。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となります。指定されていない場合、N がデフォルト値になります。このパラメータは、BEA Tuxedo 7.1 以降のソフトウェアが稼動す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。指定されていない場合、N がデフォルト値になります。このパラメータは、BEA Tuxedo 7.1 以降のソフトウェアが稼動す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 以上で 32,768 未満でなければなりません。この値が指定されていない場合は、デフォルトで 500 が設定されます。値を 0 に設定した場合は、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 を有効な値としては使用できません。

*GROUPS セクション

サーバ・グループを記述するセクション。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 を作成した後で、tmconfig、wtmconfig(1) を使用して追加することができます。サーバ・グループ・エントリは、マシン上のサーバまたはサービス (あるいはその両方) の集合の論理名です。この論理名は、SERVERS セクションの SRVGRP パラメータの値として使用され、1 つのサーバをこのグループのメンバとして識別します。SRVGRP はまた、サービスの特定のインスタンスをそのグループ内でのオカレンスで識別するために、SERVICES セクションでも使用されます。他の GROUPS パラメータはこのグループを特定のリソース・マネージャ・インスタンス (たとえば、従業員データベース) と関連付けます。GROUPS セクション内の行の形式は、次の通りです。

GROUPNAME required_parameters [*optional_parameters*]

ここで、*GROUPNAME* は、グループの論理名 (*string_value*) です。グループ名は、GROUPS セクションのすべてのグループ名と MACHINES セクションの LMID 値を通して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。このパラメータは 30 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。

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 モードの省略時の値) 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 つまで指定することができます。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。この値は 0 より大きく、30000 未満でなければなりません。また、GROUPS セクションのすべてのエントリを通して一意である必要があります。

以下は、オプション・パラメータです。

TMSNAME = *string_value*
: 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() で開始されて tpcommit()/ tpabort() で終了される、複数のリソース・マネージャや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。一意のサーバ識別子が各 TM サーバに対して自動的に選択され、サーバは何回でも再起動することができます。

ENVFILE = *string_value*
: グループ内のすべてのサーバを指定ファイルの環境で実行することを指定します正しくないファイル名を指定すると環境変数に追加されません。行は *ident=value* の形式であり、*ident* に下線またはアルファベット以外の文字を使用することはできません。*value* 内で、${*env*} という形式の文字列は、既に環境内にある変数を使用してファイルが処理されるときに展開されます。前方参照はサポートされておらず、値が設定されていない場合は、変数は空の文字列に置き換えられます。ダラー記号とバックスラッシュ (\) をエスケープするには、バックスラッシュを使用します。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、拡張された *value* が環境で使用されます。

  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、省略時の値は TMSNAME が指定されているグループの場合は 3 です。このパラメータが指定されていて、その値が 0 以外である場合は最小値は 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、セキュリティの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き、最大 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 つ以上のシステム・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 以降を実行するシステム・プロセスに対して復号化キーをオープンする処理に関係するパラメータです。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き、最大 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き、最大 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、各復号化キーのそれぞれのパスワードを入力する必要があります。パスワードの入力を求めるプロンプトは、tmloadcf(1) によって表示されます。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。指定されていない場合、N がデフォルト値になります。このパラメータは、BEA Tuxedo 7.1 以降のソフトウェアが稼動す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。指定されていない場合、N がデフォルト値になります。このパラメータは、BEA Tuxedo 7.1 以降のソフトウェアが稼動す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*

このグループに対してリソース・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場合、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 パラメータは TMS 以外の値に設定されているが、OPENINFO 文字列にヌル文字列 ("") が設定されている場合、またはこのパラメータが指定されていない場合、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 文字列の形式は基にするリソース・マネージャの供給元の要件に応じて決まります。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、*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。次に示す例は、Oracle リソース・マネージャをオープンする際に必要な情報のタイプを示す OPENINFO 文字列です。

```
OPENINFO="Oracle_XA:
Oracle_XA+Acc=P/Scott/*****+SesTm=30+LogDit=/tmp"
```

Oracle_XA は、Oracle XA インターフェイスの公開名です。OPENINFO 文字列の連続する 5 つのアスタリスク (*) は、パスワードの暗号化に関係しています。パスワードの暗号化については、この後で説明します。

OPENINFO 文字列のリソース・マネージャに渡されるパスワードは、クリア・テキストまたは暗号化形式のいずれの方法で格納することもできます。パスワードを暗号化するには、まず、パスワードを必要とする位置で OPENINFO 文字列に 5 つ以上のアスタリスクを連続して入力します。次に、tmloadcf(1) を実行して UBBCONFIG ファイルをロードします。tmloadcf() がアスタリスクを検出すると、パスワードの作成を求めるプロンプトが表示されます。次の例を参照してください。

```
tmloadcf -y /usr5/apps/bankapp/myubbconfig
Password for OPENINFO (SRVGRP=BANKB3):
password
```

tmloadcf(1) は、パスワードを暗号化形式で TUXCONFIG ファイルに格納します。tmunloadcf() を使用して TUXCONFIG ファイルから UBBCONFIG ファイルを再生成すると、パスワードは @@ 付きの暗号化形式で区切り文字として UBBCONFIG ファイルに出力されます。次の例を参照してください。

```
OPENINFO="Oracle_XA:
Oracle_XA+Acc=P/Scott/@@A0986F7733D4@@+SesTm=30+LogDit=/tmp"
```

tmunloadcf() によって生成された UBBCONFIG ファイル内で、tmloadcf() が暗号化されたパスワードを検出しても、パスワードの作成を求めるプロンプトは表示されません。

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場合、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 パラメータは TMS 以外の値に設定されているが、CLOSEINFO 文字列にヌル文字列 ("") が設定されている場合、またはこのパラメータが指定されていない場合、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、基にするリソース・マネージャの供給元の要件に応じて決まります。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・グループについて説明します。ネットワークの複数のグループには、複数のペア・マシンがあることがあります。グループ間のエレメントの通信方法を決定するために、2 つの通信ノードが優先度メカニズムを使用します。

LMID はそれぞれ、デフォルトのネットワーク・グループである DEFAULTNET のメンバーでなければなりません。リリース 6.4（NETGROUPS が使用可能）以前にリリースされた BEA Tuxedo 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・グループ番号（NETGRPNO）は、0 (ゼロ) で、変更しません。しかし、DEFAULTNET のデフォルト優先順位は、変更されることがあります。

このセクションのエントリの一般的な形式は次の通りです。

NETGROUP required_parameters [*optional_parameters*]

ここで、*NETGROUP* は、ネットワーク・グループ名のことです。*NETGROUP* が DEFAULTNET と同じ場合、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、フェイル・バック時に使用できるように管理者が割り当てる必要があります。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。

以下は、オプション・パラメータです。

NETPRIO = *numeric_value*

ここで、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、優先順位のバンドを超えて並列通信が行われます。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の状況により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、ゼロ以上 8192 未満でなければなりません。指定されない場合、デフォルトの 100 が設定されます。これが唯一の変更可能な DEFAULTNET のパラメータであることに注意してください。

注記 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。

NETWORK セクション

LAN 環境のネットワーク・コンフィギュレーションを記述する NETWORK セクション。BRIDGE サーバが位置している各プロセッサごとに、ネットワーク・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通りです。

LMID required_parameters [*optional_parameters*]

ここで、*LMID* は、BRIDGE プロセスが位置している論理マシンです。*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*

このパラメータは、*LMID* の待機アドレスとして *LMID* に置かれている BRIDGE プロセスが使用する完全ネットワーク・アドレスを指定します。BRIDGE のリスニング・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 "0xhex-digits" または "\\xhex-digits" の場合、偶数の有効 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。

"//host.name:port_number"
"//#.#.#.#:port_number"

最初の形式では、ドメインはローカル・ネーム解決機能（通常 DNS）を使って、*hostname* のアドレスを見つけます。*hostname* はローカル・マシンでなければならず、ローカル・ネーム解決機能は、*hostname* を解決してローカル・マシンのアドレスを取得しなければなりません。2 列目の "*#.#.#.#*" 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 の範囲の 10 進数で、指定された文字列の 16 進の表現です。

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基本トランスポート・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。

以下は、オプション・パラメータです。

BRIDGE = *string_value*

このパラメータは、*LMID* 上に置かれた BRIDGE プロセスがネットワークにアクセスするのに使用するデバイス名を指定します。6.4 以上のバージョンでは、BRIDGE パラメーターが求められことはありません。以前のバージョンでは TLI に基づいたネットワークのため、デバイスのための絶対パス名が BRIDGE の値として要求されました。ネットワーク・トランスポートのエンドポイントのファイル・パスは /dev/*provider_name* のフォームをとります。STARLAN の場合、/dev/starlan になります。

NLSADDR = *string_value*

このパラメータは、*LMID* が識別するノード上でネットワークにサービスを提供する tlisten(1) プロセス用のネットワーク・アドレスです。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。アドレスが "0x" や "\\x" をつけた 16 進形式である場合は、有効な 16 進数の偶数を含まなければなりません。TCP/IP アドレスは "*#.#.#.#:port*" の形式にできます。NLSADDR が、警告を表示する MASTER LMID 以外のどのエントリにもない場合、tmloadcf(1) はエラーを表示します。ただし、NLSADDR が MASTER LMID にない場合には、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*

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。このパラメータと FRANGE パラメータは、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 アドレスを指定してください。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。このパラメータが設定されていない場合は、空文字列がデフォルト値になり、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 に設定され、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 65,535 です。デフォルト値は 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ます。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。

MAXENCRYPTBITS = {0 | 40 | 56 | 128}

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 は暗号化を行わないことを示します。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。デフォルト値は "128" です。

注記 リンク・レベル暗号化の値の 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。

NETGROUP = *string_value*

string_value はネットワーク・エントリーと関連したネットワーク・グループです。指定されない場合、デフォルトは DEFAULTNET とされます。NETGROUP パラメーターが DEFAULTNET に設定されなければ、ファイルの NETGROUPS セクションにあるグループ名として以前に現れたものを持たなければなりません。以前のバージョンと相互操作を可能にするために、DEFAULTNET の NETGROUP を伴う全てのネットワーク・エントリーは TM_MIB の T_MACHINE クラスで、そして他の NETGROUP と関連する NETWORK エントリーは TM_MIB の T_NETMAP クラスで説明されています。

*SERVERS セクション

システム内で起動されるサーバの初期条件についての情報を提供するセクション。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。ホスト独自のマニュアル・ページには、UBBCONFIG サーバ・テーブル・エントリが特定の環境に適用されるかどうか、また適用される場合は、対応する構文が明示されていなければなりません。

AOUT required_parameters [*optional_parameters*]

ここで、*AOUT* は tmboot(1) により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(1) 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、/usr/bin、それから <path> の順番に (<path> はマシン環境ファイル上の始めの PATH= 行です。) *AOUT* が検索されます。APPDIR と TUXDIR の値は、TUXCONFIG ファイルの適切なマシン・エントリからとられます。詳細は MACHINES セクションの ENVFILE を参照してください。

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、サーバ・グループ・パラメータを指定する必要があります。

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。

省略可能パラメータは、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) が使用します。サーバは実行されると、コンフィギュレーション・ファイルからそのエントリを読み取り、実行時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。

省略可能なブート・パラメータには以下のものがあります。

CLOPT = *string_value*
: ブート時に *AOUT* に渡される servopts(5) オプションを指定します。none を指定すると、省略時設定 -A がとられます。*string_value* の長さは 256 文字まで指定できます。

SEQUENCE = *number*
: このサーバを、他のサーバに関連していつブートまたはシャットダウンするかを指定します。SEQUENCE パラメータの指定がない場合は、サーバは SERVERS セクションと同じ順序でブートされます (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、シーケンス番号をもつサーバがすべて昇順にブートされ、次に、シーケンス番号をも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ブートされます。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 によりブートされるサーバのオカレンスの最小数を指定します。RQADDR が指定されていて、MIN が 1 よりも大きい場合は、そのサーバは MSSQ セットを形成します。そのサーバのサーバ識別子は、最高 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 の値の範囲は 0 から 1000 です。指定されていない場合、1 がデフォルト値になります。

MAX = *number*
: ブートできるサーバのオカレンスの最大数を指定します。始めは、tmboot が MIN サーバをブートます。以降は、次に -i オプションで、関連するサーバ識別子を指定することにより、最大、MAX 個まで同じサーバをブートすることができます。MAX の値の範囲は MIN から 1000 です。このパラメータの指定がない場合、省略時の値は MIN と同じになります。

省略可能な実行時パラメータには次のようなものがあります。

ENVFILE = *string_value*
: 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できるサーバ・グループに関連付けられている場合は、その両方のマシンの同じ位置に ENVFILE が必要です。

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。このファイル環境を変更するために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。

CONV = {Y | N}
: サーバが会話型サーバであるかどうかを示します。接続は会話型サーバとの間でのみ確立でき、(tpacall() または tpcall() を介した) RPC 型要求は、非会話型サーバとの間にのみ確立できます。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。このパラメータは 30 文字以下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、同じキュー名をもつ RQADDR が指定されている場合は、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 の指定がない場合で、PERM が RESOURCES セクションに指定されている場合は、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は、応答を受けようとするサーバの REPLYQ は Y に設定してください。

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 の指定がない場合は、省略時の値 0666 が使用されます。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は、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*

AOUT が再起動できる場合は、このパラメータは、*AOUT* が異常終了したときに実行され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 との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。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 により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。この値は 0 より大きく、256 より小さくなければなりません。このパラメータの指定がないと、省略時の値 1 がとられます (つまり、サーバは 1 度は起動することができるが、再起動することはできないということです)。

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高 MAXGEN 回再起動できることを示します。この値は、0 以上 2147483648 未満でなければなりません。0 の場合は、*AOUT* は何回でも再起動できます。GRACE の指定がない場合は、省略時設定 86,400 秒 (24 時間) がとられます。

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。サーバの移行が指定されている場合は、RESTART を Y に設定する必要があります 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・ライブラリがその内部テーブルへのアクセス権を獲得するために使用する省略時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 は、ライブラリが、共有メモリを介して内部テーブルを迅速にアクセスできるように指定します。PROTECTED は、内部テーブルを共有メモリを介して BEA Tuxedo システム・ライブラリがアクセスできるときに、それらのテーブルの共有メモリを BEA Tuxedo システム・ライブラリの外部からはアクセスできないようにすることを指定します。NO_OVERRIDE を指定して (単独で、または FASTPATH あるいは PROTECTED とともに)、アプリケーション・プロセスが選択されたモードを変更することはできないことを示すことができます。SYSTEM_ACCESS の指定がない場合、その省略時設定は RESOURCES セクションの SYSTEM_ACCESS キーワードの設定により決まります。

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。あるスレッドが BEA Tuxedo コードを実行中、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・コードを実行していることがあるからです。BEA Tuxedo システムでは、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ている場合にのみ有効です。

MAXDISPATCHTHREADS が 1 より大きい場合 (MAXDISPATCHTHREADS>1) は、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。このパラメータが指定されていない場合、1 がデフォルト値になります。

MINDISPATCHTHREADS = *number*

最初のサーバ起動時に開始されるサーバ・ディスパッチ・スレッドの数を指定します。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ている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*

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、オペレーティング・システムのデフォルト値が使用されます。このオプションがサーバに対して有効なのは、MAXDISPATCHTHREADS に 1 より大きな値が指定されている場合のみです。

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、セキュリティの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き、最大 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、このサーバ上で実行される 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 以降を実行するシステム・プロセスに対して復号化キーをオープンする処理に関係するパラメータです。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き、最大 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き、最大 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、各復号化キーのそれぞれのパスワードを入力する必要があります。パスワードの入力を求めるプロンプトは、tmloadcf(1) によって表示されます。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 以上で 32,768 未満の値を指定します。そうでない場合は、文字列の値として DEFAULT が使用できます。その場合、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。値が指定されていない場合は、文字列 DEFAULT が有効な値として使用されます。0 を指定した場合は、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。この属性に指定できる値は最大 32,767 です。

SERVICES セクション

アプリケーションが使用するサービスに関する情報を提供するセクション。
SERVICES セクション内の行の形式は次の通りです。

SVCNM [*optional_parameters*]

ここで、*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。

このセクションには必須パラメータはありません。パラメータが必要ない場合は、サービスをリストする必要はありません。以下は、オプション・パラメータです。

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 以下の値を指定することができます。指定されない場合、デフォルトの 50 が設定されます。数値が高くなるほどロード・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 が優先度の最高値)。省略時設定は 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もつことが可能になります。このパラメータは 30 文字以下でなければなりません。

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

このサービスが受け入れるデータ・バッファのタイプおよびサブタイプ。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することができます。BEA Tuxedo システムで提供されるデータ・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) 参照）。サブタイプをもつ TYPE では、サブタイプに "*" を指定して、該当サービスに関連するタイプのすべてのサブタイプを受け入れさせることができます。

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) 参照)。BUFTYPE パラメータが ALL に設定されている場合は、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入れます。BUFTYPE パラメータを省略することと、このパラメータを ALL に設定するのとは同じことです。サービス名は同じで異なる SRVGRP パラメータをも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、どこで終わるのか解りにくくなります)。

有効な BUFTYPE 指定には次のようなものがあります。

BUFTYPE=FML は、サービスが FML バッファをとることを示します。
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。このパラメータが指定されていない場合、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 パラメータをもつ複数のエントリがある場合は、ROUTING パラメータはこれらすべてのエントリに対し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 の場合は、サービスにタイムアウトを適用しないことを示します。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 信号で終了します。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。指定されていない場合、N がデフォルト値になります。このパラメータは、BEA Tuxedo 7.1 以降のソフトウェアが稼動す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。指定されていない場合、N がデフォルト値になります。このパラメータは、BEA Tuxedo 7.1 以降のソフトウェアが稼動す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 アプリケーションのみのものです。

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*
: このサービス用に自動的に開始されるトランザクションに対して、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 2147483648 未満でなければなりません。デフォルトは 30 秒です。0 を指定すると、マシンの最大タイムアウト値に設定されます。

INTERFACES セクション

このセクションは、アプリケーションで使用する CORBA インターフェイスに対して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提供します。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。

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	■ 使用するインターフェイスの名前 ■ 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それに関連するパラメータの詳細については、後述の「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。

以下のオプション・パラメータを使用できます。

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させ、呼び出しから復帰したときにトランザクションを終了させるかどうかを指定します。AUTOTRAN パラメータは、トランザクション方針が `optional` のインターフェイスにのみ適用されます。それ以外の場合は無視されます。
デフォルトは `N` です。

  トランザクション方針は、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。トランザクション方針は、実行時に対応する `T_IFQUEUE MIB` オブジェクトのトランザクション方針の属性として使用されます。

システム管理者は AUTOTRAN 値を設定する前に、プログラマがインターフェイスに割り当てているトランザクション方針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 が Y に設定されている場合は、TRANTIME パラメータを設定する必要もあります。

FACTORYROUTING = *criteria-name*

このインターフェイスへのオブジェクト・リファレンスを作成する際にルーティング基準を使用する場合は、このパラメータが必要です。ルーティング基準は、UBBCONFIG ファイルの ROUTING セクションで指定されます。

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。この数字は、このアプリケーションで使用されるほかの CORBA インターフェイスの LOAD の数値を基準に指定します。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、ルーティングされた CORBA インターフェイスのロード・ファクタ分 (LOAD) だけ増加します。

PRIO = *number*

CORBA インターフェイスのメソッドすべてに対して、キューから取り出される際の優先順位を指定します。この値は 0 より大きく 100 以下でなければなりません。最も高い優先順位は 100 です。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は、TRANTIME パラメータを設定する必要があります。この値は 0 〜 2,147,483,647 (2^{31-1}) つまり、約 70 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。デフォルト値は 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。タイムアウトが発生すると、インターフェイスのメソッドを処理するサーバは、SIGKILL シグナルと共に終了します。実行に最も時間がかかるメソッドには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提供します。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) を参照してください。ROUTING セクションの行の形式は次の通りです。

CRITERION_NAME required_parameters

ここで、*CRITERION_NAME* は、サービス・エントリで指定されたルーティング・エントリの (*string_value*) 名です。*CRITERION_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。このパラメータは 30 文字以下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 VIEW 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、メッセージ送信時に、データ依存型ルーティングの関連するフィールド値を得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、フィールドの値は 8,191 以下の数値でなければなりません。

エレメントの内容またはエレメント属性に基づいて XML ドキュメントのルーティングを行うには、次の構文で FIELD パラメータの値を定義する必要があります。

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

FIELD の値は、ルーティングの要素または属性名を指定します。この要素は、要素のタイプ (または名前)、あるいは XML ドキュメントまたはデータグラムの要素の属性名として見なされます。この情報は、メッセージ送信時に、データ依存型ルーティングの要素の内容または属性値を識別するために使用されます。要素名と属性名の組み合わせで使用できる文字数は 30 文字以内です。インデックス付けはサポートされていないため、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とき、指定されたエレメント・タイプの最初のオカレンスだけを認識します。

XML では、属性名で使用される文字セットを厳密に定義します。属性名は、単一の文字、下線、またはコロンの後に 1 つ以上の名前文字を続けた文字列でなければなりません。要素名と属性名では共に大文字 / 小文字を区別します。

XML については、http://www.w3c.org/XML の World Wide Web Consortium の Web サイトで詳しく解説されています。

FIELDTYPE = *type*

FIELD パラメータで指定されるルーティング・フィールドの型を示します。このパラメータは、XML バッファのルーティングでのみ使用されます。*type* には、CHAR、SHORT、LONG、FLOAT、DOUBLE、または STRING のいずれかを設定できます。ルーティング・フィールドのデフォルトの型は 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 1,024 文字までしか使用できません。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 (後述の「EXAMPLE」を参照)。

範囲は、単一値 (符号付き数値または文字列を一重引用符で囲んだもの) である場合と、"lower - upper" の形式をとる場合があります (ここで lower と upper は一重引用符で囲んだ符号付き数値または文字列です)。"lower" の値は、"upper" の値より小さくなければなりません。一重引用符を文字列値の中に入れる場合は (O'Brien など)、その前に "¥" を 2 つ置く必要があります ('O¥¥'Brien')。MIN という値を使用すると、マシンの関連フィールドのデータ・タイプに対する最小値を示すことができます。また、MAX という値を使用すると、マシンの関連フィールドのデータ・タイプの最大値を示すことができます。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、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 (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 および long 整数値は数字列で、その前に任意にプラス記号やマイナス記号を付けることができます。C コンパイラまたは atof() により受け入れられる浮動小数点数は、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とります。

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することができます。

ルーティングに使用する際は、要素の内容に文字参照、エンティティ参照、または 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 の属性値は、そ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*[, . . .]]] . . ."

これは、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。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することができます。タイプは、FML、FML32、XML、VIEW、VIEW32、X_C_TYPE、または X_COMMON のいずれかでなければなりません。FML、FML32、または XML タイプには、サブタイプを指定できません。VIEW、VIEW32、X_C_TYPE、および X_COMMON タイプにはサブタイプが必要です。"*" 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 (*) といった記号は使用できません。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 / サブタイプのペアが一意であれば、複数のルーティング・エントリに対して同じ基準名を指定することができます。このパラメータは必須です。1 つのルーティングに対して複数のバッファ・タイプを指定した場合は、それぞれのバッファ・タイプのルーティング・フィールドのデータ・タイプを同じにする必要があります。

ルーティング・エントリの例を次に示します。

```
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) や、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。

XML の要素の CODE に基づくルーティング・エントリの例を次に示します。

```
PRODUCT FIELD="ORDER/CODE" RANGES="'AAA' - 'FFF':DBG1, 'GGG-ZZZ':DBG2"
BUFTYPE="XML"
```

ここでは、CODE は、ルート・エレメント ORDER の子エレメントです。

ORDERNO 属性に基づくルーティング・エントリは、以下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・エレメント ORDER の XML 子エレメントである 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・ファイルが検出されるようにするためには、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。
*RESOURCE
IPCKEY      80952 # 周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

相互運用性のあるアプリケーションでは、その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 システム間で互いに相互運用する場合、PMID (マシン ADDRESS)、LMID、TLOGNAME、グループ名、RQADDR、サービス名、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

たとえば、BRIDGE を実行しているローカル・マシンは TCP/IP アドレス指定機能を使用していて、アドレスは 155.2.193.18 であり、backus.company.com と名付けられているとします。さらに BRIDGE が要求を受け取るポート番号は 2334 だとします。そして、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仮定します。-l オプションで指定されるアドレスは、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
```

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。その後に IP アドレス 155.2.193.1 の各エレメントは 16 進数に変換されています。つまり 155 が 9B になり、2 が 02 になるように次々に変換されています。

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は、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) を形成する際の元になります。viewc、viewc32(1) を参照してください。

BEA Tuxedo システムでは、バイナリ・ファイル .V は以下の 2 通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 ファイルを実行時に解釈して、FML バッファと C 構造体との間のマッピングを行います。
- 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で .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 つの部分からなっています。

- 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 文字までで、有効な C 識別子でなければなりません。つまり、先頭が下線またはアルファベットで、英数字または下線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" で始め、その後に VIEW 記述の名前を続け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" がある行は、VIEW 記述の最後に置かなければなりません。先頭に # が付いている行は注釈として扱われ、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
```

VIEW 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
: メンバの型を示します。int、short、long、char、float、double、string、carray、または dec_t のいずれかを指定します。ダッシュ (-) 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。これは有効な *fbname* とすることはできません。

fbname
: フィールド化バッファのフィールドの名前です。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 の場合は、このフィールドには、ダッシュ (-) などの特別の意味をもたない引数値を入れる必要があります。

count
: 割り当てるエレメントの数、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を示します。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、後で説明します。FML バッファにマップしない VIEW の場合は、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）を含むことができ、またはダッシュ (-) プレース・ホルダ値を含まなければなりません。

size

type 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。dec_t 型では、size は 2 つの数値をカンマで区切ります。1 つめの数値は 10 進数値のバイト数 (0 より大きく、10 未満でなければならない) で、2 つめの数値は小数点の右側の桁数です (0 より大きく、バイト数に 2 をかけた数値から 1 を引いた値未満でなければならない)。その他のフィールド・タイプでは、(-) を指定すると VIEW コンパイラがサイズを計算します。

null

ユーザ指定のヌル値を示します。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、後で説明します。

フラグ・オプション

VIEW 記述のメンバ記述の `flag` 要素として指定できるオプションのリストを以下に示します (フラグ・オプションは FML 対応の VIEW にのみ適用されます)。`L` および `C` オプションは、FML ベースではない VIEW 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 を関連す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、転送される配列要素数を示すために ACM が使用されます。たとえば、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 が配列の大きさを超える場合は、省略時の設定としてその配列の大きさがとられます。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 のタイプは short として宣言され、その名前は "C_*cname*" の形で生成されます (ここで cname は、ACM が宣言されている *cname* エントリ)。たとえば、`parts` という名前のメンバの ACM は、以下のように宣言されます。

```
short C_parts;
```

生成された ACM 名が、先頭に "C_" 接頭辞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。このオプションをもつ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の場合にのみ有効です。

L

このオプションはタイプが文字配列または文字列であるメンバ記述にのみ使用し、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。文字列フィールドまたは文字配列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には、このオプションは無意味です。L オプションにより、タイプが文字配列または文字列である構造体メンバの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へデータを転送する場合は、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越える場合は、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。したがって、ALM が構造体メンバの配列の大きさを超える場合は、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、そのフィールドが文字配列タイプであれば、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。文字列の場合は、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。文字配列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 は符号なし short となるように定義され、"L_*cname*" という名前が付きます (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は、次のように ALM を宣言します。

```
unsigned short L_cname;
```

一方、オカレンス数が 1 より大きい場合、つまり N の場合、ALM は次のように宣言され、

```
unsigned short L_cname[N];
```

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と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 メンバと ALM (前述の C 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。

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、どの値を文字列および文字配列タイプの構造体メンバのヌル値として解釈されるようにするかを決めるために使用します (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)。P オプションを指定しない場合は、構造体メンバの値がユーザ指定の NULL 値と等しいと (ただし、後続の NULL 文字は考慮に入れない)、構造体メンバが NULL 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 NULL 値と構造体メンバの値が等しく、しかも最後の文字が完全な長さまで及んでいると (ただし、後続の NULL 文字は考慮に入れない)、構造体メンバが NULL 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 のタイプが文字配列 [25] で、それに対し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は、TEST は、最初の 5 文字が、それぞれ a、b、c、d、e であればヌルとみなされます。P オプションが設定されている場合は、TEST は、最初の 4 文字がそれぞれ 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 (このオプションでは、FML ベースでない VIEW では無視されます)。このオプションによるメンバのマッピングが行えるのは、フィールド化バッファから構造体へのデータの転送時にのみです。

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。省略時のヌル値が提供されていますが、独自のヌル値を定義することもできます。

省略時のヌル値は、数値タイプの場合はすべて 0 であり、char タイプでは "\\0"、文字列および文字配列タイプでは "" 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、\\|\\|、\\|'、\\|"\| です。

文字列、文字配列、char のヌル値は二重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入れません。

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 型である場合には、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。
- メンバのタイプが文字列である場合。この場合はその値は必ずヌル値と同じ文字列です。
- メンバのタイプが文字配列で、ヌル値が長さ N である場合。この場合は文字配列の最初の 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省略時値の最大サイズは 2660 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 ファイルをカンマで区切って示した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`
: VIEW オブジェクト・ファイルがある可能性のあるディレクトリをコロンで区切って示したリストを指定します。`VIEWDIR` の設定がない場合は、カレント・ディレクトリとなる値がとられます。

`VIEW32` では、環境変数 `VIEWFILES32` および `VIEWDIR32` が使用されます。

使用例

```
# FML ベース 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用の管理情報ベース

形式

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

機能説明　　BEA Tuxedo システムの MIB は、/WS 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。

管理要求をフォーマットしたり管理応答の意味を解釈するためには、WS_MIB は共通 MIB のマニュアル・ページ MIB(5) とともに使用する必要があります。このマニュアルに記載されたクラスや属性を利用して MIB(5) の手順に従ってフォーマットした要求は、アクティブなアプリケーション中に存在するさまざまな ATMI インターフェイスを利用して管理サービスを要求するために使用できます。WS_MIB(5) のすべてのクラス定義に関連する追加情報については、524 ページ「WS_MIB(5) に関する追加情報」を参照してください。

WS_MIB(5) は、以下のクラスから構成されます。

WS_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、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た属性の詳細な説明

属性表
: クラスの各属性に関する名前、タイプ、パーミッション、値、およびデフォルト値を示す表。属性表の形式を以下に示します。

属性の意味
: 各属性の意味を説明します。

制限事項
: このクラスにアクセスしたり、このクラスを解釈する場合の制限事項。

属性表の形式

既に述べたように、本 MIB に含まれる各クラスは、以下に 4 つの部分に分けて定義されます。その内の 1 つが属性表です。属性表は、クラス内の属性と、管理者、オペレータ、一般ユーザが属性を使用してどのようにアプリケーションとのインターフェイスをとるかを示した 1 ページの参照ガイドです。属性表の各属性の記述は、5 つの項目 (名前、タイプ、パーミッション、値、デフォルト値) から成ります。各項目については MIB(5) で説明します。

TA_FLAG 値

MIB(5) は、共通およびコンポーネント MIB 固有のフラグ値の両方が入った long 値フィールドである共通 TA_FLAGS を定義しています。この際、WS_MIB(5) 固有のフラグ値は定義されません。

FML32 フィールド・テーブル

このマニュアル・ページに記述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で区切ったリストに、またフィールド・テーブル名 tpadm は FIELDTBLS 環境変数で指定されるカンマで区切ったリストに、アプリケーションで指定しなければなりません。

制限事項

この MIB のヘッダ・ファイルやフィールド・テーブルには、BEA Tuxedo リリース 6.0 以降のサイト (ネイティブ版および /WS 版の両方) で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 /WS 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 クラスにリンクされます。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

WS_MIB(5): T_WSH **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_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

WS_MIB(5): T_WSH **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_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	R--R--R--	1 <= *num*	N/A
TA_NADDR	string	R--R--R--	*string*[1..78]	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
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

WS_MIB(5): T_WSH 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 操作のために 1 つ以上のキーが必要

注 1 T_WSH クラスの属性は、すべてローカル属性です。

属性の意味

TA_CLIENTID:*string*[1..78]
この WHS のクライアント識別子。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_WSHCLIENTID:*string*[1..78]
この WHS の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。このフィールドを使用して、WSH をその対応する /WS クライアントの T_CLIENT オブジェクトにリンクすることができます。このフィールド値は、T_CLIENT クラスの TA_CLIENTID 属性の値とつねに同じです。

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 クライアントに対する状態。TM_MIB(5) 内の T_CLIENT クラスに対して定義されたすべての状態は、このマニュアル・ページに示されたとおりに返されるかセットされる可能性があります。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点に留意する必要があります。ただし、DEAD への状態変化は T_WSH クラスを介して許され、結果として、不成功に終わるターゲット WSH がすべての接続を処理することになります。

TA_LMID:*LMID*
WSH が稼働している現在の論理マシン。

TA_PID:1 = *num*
WSH クライアントに対するネイティブ・オペレーティングシステム・プロセス識別子。プロセス識別子の重複を許す異なるマシンにクライアントが存在している可能性があるので、これは一意の属性ではないかもしれません。

TA_NADDR:*string*[1..78]
: 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*" の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 以外の値を設定すると、指定された時間量（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ることを示しています。

TA_ACTIVE:{Y | N}
: "Y" の値は、WSH がその関連付けられたワークステーション・クライアントの 1 つの代わりに作業を現在実行していることを示しています。"N" の値は、その関連付けられたワークステーション・クライアントの代わりに実行する処理を現在待っていることを示しています。

TA_TOTACTTIME:0 <= *num*
: WSH が処理開始後アクティブ状態にあった時間（秒数）。

TA_TOTIDLTIME:0 <= *num*
: WSH が処理開始後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 WSH プロセスのセット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 がフロー制御に遭遇した回数。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に関してのみ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に関してのみ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に関してのみ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・メッセージの数。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に関してのみ考慮する必要があります。

TA_SENTBYT:0 <= *num*
: この WSH が、その現在と過去のワークステーション・クライアントのすべてに送出したバイト数。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に関してのみ考慮する必要があります。

TA_SENTNUM:0 <= *num*
: この WSH が、その現在と過去のワークステーション・クライアントのすべてに送出した BEA Tuxedo システム・メッセージの数。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に関してのみ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内でここに記載されていないものは、T_CLIENT クラスを通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、/WS 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・プロセスの構成属性と実行時の属性を表します。これらの属性値は、アプリケーション内の WSL T_SERVER オブジェクトの /WS 固有のコンフィギュレーション属性を識別し、特徴を示します。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_WSH にリンクされます。

属性表

WS_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. .78]	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	注 1 を参照してください。
TA_MULTIPLEX	long	rw-r--r--	1 <= *num* < 32,767	10
TA_MINENCRYPTBITS	string	rwxrwx---	{0 | 40 | 56 | 128} 注 2 を参照してください。	0
TA_MAXENCRYPTBITS	string	rwxrwx---	{0 | 40 | 56 | 128} 注 2 を参照してください。	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

WS_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_CMPLIMIT	string	rwxr-xr--	*threshold*	MAXLONG
TA_CLOPT	string	rwxr--r--	*string*[0..128]	"-A"
TA_ENVFILE	string	rwxr--r--	*string*[0..78]	""
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..78]	""
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
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 操作のために 1 つ以上のキーが必要

注 1 オブジェクト作成時にこの属性の値が指定しないと、0 が割り当てられます。この属性に 0 の値を指定すると、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る ことを示しています。MIB_LOCAL フラグをセットして 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留意してください。

注 2 リンク・レベル暗号化の値の 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。

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 (:) を入れ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 WSL サーバの状態。TM_MIB(5) 内の 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、またはこのマニュアル・ページに記載されたように設定される可能性があります。

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*
: WSL サーバのオペレーティング・システムのプロセス ID。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には、この属性は一意となりませんので注意して下さい。

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..78]
: WSL プロセスがそのリスニング・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 用のリスニング・アドレスを使って、アプリケーションに参加しているワークステーション・クライアント・プロセスが WSL プロセスにコンタクトします。文字列の形式が "0xhex-digits" または " ¥¥xhex-digits" の場合、偶数の有効 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。また、*string* の値も以下のいずれかの形式で表されます。

  //host.name: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ネーム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 進数を表しています。*Port_number* は 0 から 65535 の間の数字または名前です。

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基本トランスポート・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 プロセスの周知のアドレス・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周知のネットワーク・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。このアドレスは、TA_NADDR とほぼ同じ形式です。異なる点は、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、ポート番号を同じ長さの文字 M で置き換えている点です。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、周知のネットワーク・アドレスは 0x00021111dddddddd です。アドレス・テンプレートが "//" で始まるときは、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、ネットワーク・アドレスを組み合わせるためにアドレス・テンプレートへとコピーされます。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称です。これに対するデフォルト値は、システム提供のワークステーション・ハンドラに相当する 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することが可能です。詳細については、カスタマイズの項と buildwsh(1) のマニュアル・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 と共に使用可能になる必要があるハンドラの最小数。WSL は起動されるとすぐに、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
: WSL と関連して常時使用可能とされるべきハンドラの最大数。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、最小数のハンドラに対する設定値より大きいか等しくなければなりません。

TA_MULTIPLEX:1 <= *num* < 32,767
: 任意の 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ます。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。

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ます。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。デフォルト値は "128" です。

  注記 リンク・レベル暗号化の値の 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。

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、利用できる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、利用できる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前にアイドルでいられる最大時間（分）。35,204,650 の値の場合、クライアントはタイムアウトにされることなく無制限にアイドル状態にあることができます。0 の値の場合、クライアントが非アクティブ化の時間が 1 秒を超えたら終了できます。

TA_MAXINITTIME:1 <= *num* < 32,767
: ワークステーション・クライアントが、WSL によってタイムアウトにされる前に WSH を介して初期化プロセスを完了することができる最小時間（秒数）。

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・サイズです。しきい値は、文字列 "MAXLONG" または負以外の数値です。"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) マニュアル・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（つまり、ダブルダッシュ "--" の後のオプション）は設定されないことがあり、返されません。

TA_ENVFILE:*string*[0..78]
: 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 : 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 制限が適用される期間（秒数）。この属性は再起動可能な WSL サーバにしか意味がありません。つまり、T_WSL:TA_RESTART 属性が "Y" に設定されている場合にだけ意味があります。再開サーバが TA_MAXGEN 制限値を超えても、TA_GRACE 時間に達していれば、システムは現在の世代 (すなわち、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" を指定すれば、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。

  この値を低く設定しすぎると、切断される可能性が非常に高くなることに気をつけてください。

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"）に対して許された世代数。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 を参照してください。

TA_NETTIMEOUT:0 <= *num* <= MAXLONG
: TA_NETTIMEOUT 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。デフォルトの値は 0 で、この値はタイムアウトがないことを示します。

  この値を低く設定しすぎると、切断される可能性が非常に高くなることに気をつけてください。

TA_RCMD:*string*[0..78]
: システムによるアプリケーション・サーバの再開と同時に実行する、アプリケーションが指定するコマンドです。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で実行可能な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 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
: いつ、このサーバを他のサーバに対してブート （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従って tmboot() で起動され、降順で tmshutdown() 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() だけに影響を与え、稼働中のサーバが 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_CURHANDLERS:0 <= *num*
: この WSL に関連付けられた現在アクティブなハンドラの数。

TA_HWHANDLERS:0 <= *num*
: ある一定の時点にこの WSL に関連付けられた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 ワークステーション・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留意してください。

TA_SUSPENDED:"{NEW | ALL | NONE}"
: "NEW" の値は、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。"ALL" の値は、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。"NONE" の値は、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、/WS 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 クラスの特殊なものを表しているので、対応する T_SERVER オブジェクト内に重複されているある属性を表します。T_SERVER クラスに含まれていてリストされていない属性はそのクラスを介してアクセスする必要があり、T_WSL クラスを介して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

一般に、WS_MIB(5) とのインターフェイス時、2 つの一般的なタイプのエラーがユーザに返されます。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()) で、いずれかが各関数で定義されたエラーを返します。エラーは該当するマニュアル・ページの記述に従って解釈します。

しかし、要求を満足させることができるシステムサービスに要求が正常に送られ、システムサービスが要求処理に障害があると判断した場合、アプリケーション・レベルのサービス障害の形で異常終了が返されます。このような場合、tpcall() および 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) 経由でシステムに転送された要求に対する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仮定します)。

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 フィールにエラーの内容について説明したテキストがセットされ、TA_ERROR という FM32 フィールドにはエラーの原因 (下記参照) を示す値がセットされます。以下のエラー・コードは、いずれも負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) マニュアル・ページに記載されています。これらのエラー・コードは、ここで定義する WS_MIB(5) 固有のエラー・コードと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 に戻され、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) マニュアル・ページに記載されています。これらのリターン・コードは、ここで定義する WS_MIB(5) 固有のリターン・コードと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・ページに定義されたヘッダ・ファイルとフィールド・テーブルは、BEA Tuxedo リリース 5.0 以降で使用できます。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、このマニュアル・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。リリースの異なるサイト（両方ともリリース 6.0 以降）が相互運用を行う場合、旧サイトに関する情報は、そのリリースの MIB マニュアル・ページで定義する通り、アクセスおよび更新することができます。この情報は新しい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、さらにこのマニュアル・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・プラットフォームおよびワークステーション・プラットフォームで利用可能です。

使用例

以下の例は、TM_MIB(5) と WS_MIB(5) を組み合わせて /WS グループを順序立てて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ためには、フィールド・テーブル tpadm が必要です。そのために、以下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

以下のヘッダ・ファイルがあります。

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

WS グループを SUSpended 状態にする

以下のコードは /WS グループの状態を SUSpended に設定します。これにより、/WS 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・クライアントをサスペンドします。このコードと、その後のコードは、一緒に作業している /WS グループを識別するローカル変数の ta_srvgrp と ta_srvid がすで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を割り当て */ 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・リストの検索

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するだけです。与えられた T_WSL オブジェクト・キー・フィールド ta_srvgrp と ta_srvid に関連付けられたすべての T_WSH オブジェクトを検索します。TA_FILTER 属性を設定して検索を限定することによって効率を高めます。

```
/* 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 オブジェクトの検索

取り出された TA_WSHCLIENTID 値を使って、この /WS 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。

T_CLIENT オブジェクトへの通知

取り出された TA_CLIENTID 値を使って、ログオフする必要があるこの /WS グループ内のワークステーション・クライアントに通知します。

```
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、中断、通知 */
for (i=0; i cltcnt ;i++) {
  p = Ffind32(cltids, TA_CLIENTID, i, NULL);

  /* ログオフするクライアントを通知 */
  tpconvert(p, (char *)ci, TPCONVCLTID);
  tpnotify(ci, notptr, 0, 0);
}
```

残りの T_CLIENT オブジェクトの非アクティブ化

取り出された TA_CLIENTID 値を使って、この /WS グループ内の残りのワークステーション・クライアントを非アクティブ化します。すでに非アクティブ化されているワークステーション・クライアントは SET 時にエラーを返すことに留意してください。

```
/* 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);

/* 関係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 オブジェクトの非アクティブ化

次に、T_WSL オブジェクトを非アクティブ化します。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・クライアントによるネイティブ・サービスへのアクセスを可能にする、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 セクションにおいて、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のアプリケーションへの結合を許可します。ワークステーション・リスナおよびワークステーション・ハンドラの処理を指定するときは、コマンド行のオプションを使用します。

サーバ関連のパラメータすなわち、位置、サーバ・グループ、サーバ ID 等のパラメータは、既に定義されたサーバ関連のコンフィギュレーション・ファイルを利用して、ワークステーション・リスナと関連付けられます。ワークステーション・リスナ専用のコマンド行の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 がブートされる度に、ある一つの周知のネットワーク・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することが可能になります。WSH は、WSL によって、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、必要に応じて動的に始動・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。

下記の WSL 専用コマンド行のオプションを使用することが可能です。これらのコマンド行のオプションは、パラメータ CLOPT の二重ダッシュ (--) の後で指定することができます。

-n *netaddr*

WSL プロセスがそのリスニング・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。必須パラメータはこれだけです。

WSL 用のリスニング・アドレスを使って、アプリケーションに参加しているワークステーション・クライアント・プロセスが WSL プロセスにコンタクトします。*netaddr* (1 〜 78 文字まで含めることができる) の形式が *0xhex-digits* または ¥¥*xhex-digits* の場合には、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。また、アドレスは、以下のいずれかの形式で表されます。

//host.name:port_number
//#.#.#.#:port_number

文字列 *#.#.#.#* はドットで区切った 10 進数の形式です。各 # は 0 から 255 までの 10 進数を表します。*port_number* は 0 から 65,535 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基本トランスポート・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。このパラメータは、指定しても、指定しなくとも構いません。なぜなら、sockets 等のネットワーク・プロバイダはデバイス名を必要としないからです。

[-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称です。これに対するデフォルト値は、システム提供のワークステーション・ハンドラに相当する 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することができます。詳細については、buildwsh(1) のマニュアル・ページを参照して下さい。

[-t *timeout-factor*]
このオプションは、-I オプションによって置き換えられ、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ていますが、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。このパラメータの省略時の値は、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。有効な範囲は、1 から 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 (たとえば、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 を指定しない場合は、タイムアウトはありません。

[-m *minh*]
任意の時間に、この WSL と共に使用可能になる必要があるハンドラの最小数。WSL は、起動されるとすぐにこれと同数の WSH を始動し、管理者が WSL をシャットダウンするまで WSH の数がこの最小数を下回らないように維持します。このパラメータのデフォルト値は 0 です。有効な範囲は、0 から 255 です。

[-M *maxh*]
WSL と関連して常時使用可能とされるべきハンドラの最大数。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。このパラメータの省略時の値は、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な値の範囲は、1 から 4096 です。また、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され得る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・クライアントを処理する必要性に応じて、新たなハンドラが始動されることが確実となります。この値は、1 以上 4096 以下でなければなりません。このパラメータの省略時の値は 10 です。

[-p *minwshport*]
[-P *maxwshport*]

この対のコマンド行オプションを使用して、このリスナ・サーバに対応する WSH が利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 から 65535 までの数字でなければなりません。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基本トランスポート・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*]

WSL によるタイムアウトが発生する前にワークステーション・クライアントが WSH を通じて初期化プロセスを終了するために認められている時間 (単位は秒) を指定します。このパラメータのディフォルト値は 60 です。有効な設定範囲は、1 から 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部分的にコントロールし、他は -k オプションでコントロールします。複数の WSL/WSH の組を容認します。-k オプションは、-c オプションと同じように機能します。

[-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 を指定すれば、クライアントとハンドラの両方に対してこのオプションをオフにできます。

[-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 は暗号化を行わないことを示します。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。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・レベル暗号化の値の 40 ビットは、後方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許可され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ます。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。デフォルト値は "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。

[-H *external-netaddr*]
WSH プロセスの周知のアドレス・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周知のネットワーク・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。このパラメータの形式は、ほぼ -n オプションと同じです。異なる点は、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、ポート番号を同じ長さの文字 M で置き換えている点です。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、周知のネットワーク・アドレスは 0x00021111dddddddd です。アドレス・テンプレートが "//" で始まるときは、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、ネットワーク・アドレスを組み合わせるためにアドレス・テンプレートへとコピーされます。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*]

ネットワーク・タイムアウト・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設定できます。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。0 (ゼロ) は、タイムアウトがないことを示します。これが省略時の設定です。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。

たとえば、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どんなコンフィギュレーションも、ブート時に WSL を異常終了させます。

移植性

WSL は、サポートされているすべてのサーバ・プラットフォーム上で動作する、BEA Tuxedo システム提供のサーバとしてサポートされています。

相互運用性

WSL は、相互運用性があ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 以降のノード上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』