

Tmax JTCユーザガイド

Tmax v6.0



Copyright © 2016 TmaxSoft Co., Ltd. All Rights Reserved.

Copyright Notice

Copyright © 2016 TmaxSoft Co., Ltd. All Rights Reserved.

45, Jeongjail-ro, Bundang-gu, Seongnam-si, Gyeonggi-do, 13613, South Korea

Restricted Rights Legend

All TmaxSoft Software (Tmax®) and documents are protected by copyright laws and international convention. TmaxSoft software and documents are made available under the terms of the TmaxSoft License Agreement and this document may only be distributed or copied in accordance with the terms of this agreement. No part of this document may be transmitted, copied, deployed, or reproduced in any form or by any means, electronic, mechanical, or optical, without the prior written consent of TmaxSoft Co., Ltd.

Nothing in this software document and agreement constitutes a transfer of intellectual property rights regardless of whether or not such rights are registered) or any rights to TmaxSoft trademarks, logos, or any other brand features. This document is for information purposes only. The company assumes no direct or indirect responsibilities for the contents of this document, and does not guarantee that the information contained in this document satisfies certain legal or commercial conditions. The information contained in this document is subject to change without prior notice due to product upgrades or updates. The company assumes no liability for any errors in this document.

このソフトウェア(Tmax®)マニュアルの内容とプログラムは、日本国の著作権法および国際条約によって保護されています。マニュアルの内容とプログラムは、TmaxSoft Co., Ltd.との使用許諾契約書の下でのみ使用することができ、マニュアルは使用許諾契約で許可されている範囲を除いては、配布または複製することができません。TmaxSoftの書面による事前の承諾を得ることなく、このマニュアルの全部または一部を電子的または機械的な方法を問わず、転送、複製、配布したり、または二次的著作物を作成する等の行為を一切禁じます。

このソフトウェアのマニュアルとプログラムの使用許諾契約は、いかなる場合においても、マニュアル及びプログラムと関連する知的財産権(登録の有無を問わず)を譲渡するものと解釈されず、TmaxSoftのブランド、ロゴ、商標等の使用権限を与えるものではありません。マニュアルは、情報を提供する目的でのみ提供しており、これに伴う契約上の直接的ないしは間接的な責任を負わず、マニュアルの内容は法律上もしくは商業的な特定の条件が満たされることを保証しません。マニュアルの内容は、製品のアップグレード及び修正により、その内容が予告なく変更されることがあり、内容上の誤りがないことを保証しません。

Trademarks

Tmax®, Tmax WebtoB® and JEUS® are registered trademark of TmaxSoft Co., Ltd. Other products, titles or services may be registered trademarks of their respective companies.

Tmax®, Tmax WebtoB®, JEUS® は、TmaxSoft Co., Ltd.の登録商標です。その他、記載されている会社名、製品名などは、各社の商標または登録商標です。

Open Source Software Notice

Some modules or files of this product are subject to the terms of the following licenses. : openssl-0.9.7.m, zlib-1.1.4, expat-2.0.0, net-snmp, DCE1.0, pthread, google-diff-match-patch, libevent, getopt

Detailed Information related to the license can be found in the following directory :
\${INSTALL_PATH}/license/oss_licenses

この製品の一部ファイルまたはモジュールは、openssl-0.9.7.m、zlib-1.1.4、expat-2.0.0、net-snmp、DCE1.0、pthread、google-diff-match-patch、libevent、getoptライセンスを遵守します。

詳細情報については、製品ディレクトリーの\${INSTALL_PATH}/license/oss_licensesに記載されている事項を参照してください。

文書情報

文書名: Tmax JTCユーザガイド

発行日: 2016年8月5日

ソフトウェアバージョン: Tmax v6.0

ガイドバージョン: v2.1.1

目次

このガイドについて	ix
第1章 環境設定	1
1.1. JTCアウトバウンド通信	1
1.1.1. webt.propertiesの設定	1
1.1.2. Startupメソッドの設定	9
1.1.3. Tuxedoの環境設定	10
1.1.4. Tuxedoドメインの環境設定	10
1.1.5. JEUSとTuxedoの起動	11
1.2. JTCインバウンド通信	12
1.2.1. EJBサービスの呼び出し設定	12
1.2.2. Beanサービスの呼び出し設定	14
1.2.3. サービス関数内でのメッセージの作成方法	16
1.3. ログ設定	17
1.3.1. Tuxedoの設定	17
1.3.2. WebTの設定	19
第2章 JTCのクラスと関数	23
2.1. 概要	23
2.2. TuxServiceクラス	23
2.2.1. createCarrayBuffer	24
2.2.2. createFieldBuffer	24
2.2.3. createStringBuffer	25
2.2.4. setAttribute	26
2.2.5. setDefaultCharset	26
2.2.6. setServiceName	27
2.2.7. tpacall	27
2.2.8. tpcall	28
2.2.9. tpconnect	29
2.2.10. tpdiscon	29
2.2.11. tpgetrply	30
2.2.12. tprecv	31
2.2.13. tpsend	31
2.2.14. TuxService	32
2.3. TuxAsyncServiceクラス	32
2.3.1. getXAResource	34
2.3.2. tpacall	34
2.3.3. TuxAsyncService	35
2.4. TuxRemoteServiceManagerクラス	35
2.4.1. TuxedoRemoteServiceManager	37
2.4.2. createStringBuffer	36
2.4.3. createCarrayBuffer	36

2.4.4.	createFiledBuffer	36
2.4.5.	tpcall	37
2.5.	TuxBufferクラス	37
2.6.	TuxStringBuffer/TuxCarrayBufferクラス	37
2.6.1.	getBytes	38
2.6.2.	getString	38
2.6.3.	setBytes	38
2.6.4.	setString	39
2.7.	TuxFieldBufferクラス	39
2.7.1.	createField	39
2.7.2.	getField	40
2.7.3.	getFields	40
第3章	JTCモニタリング	43
3.1.	概要	43
3.2.	環境設定	43
3.3.	JTCモニタリング	44
3.3.1.	JEUSコンソール・ツールを利用したモニタリング	44
3.3.2.	JEUS WebAdminを利用したモニタリング	45
索引	49

図目次

[図 3.1]	webtadmin画面	46
[図 3.2]	JTC Monitoring画面	47
[図 3.3]	JTC Monitoring画面 - リモート・ドメインの詳細情報	48

このガイドについて

対象読者

本書は、Tmax[®](以下、Tmax)のJTC(JEUS-Tuxedo Connector)を利用してプログラムを開発する開発者を対象としています。

前提知識

本書は、Tmaxシステムについての全般的な理解と、Tmaxシステムが提供する各種機能および特性を習得するための基本ガイドです。

本書を理解するためには、以下の事項について熟知している必要があります。

- ミドルウェアおよびUNIXシステムについて
- Tmaxの基本概念について
- JEUSの基本概念について
- Tuxedoの基本概念について

制約事項

本書で触れているJEUSおよびTuxedoについては、本書で詳しく説明しておりません。該当内容についてはJEUSおよびTuxedoのガイドを参照してください。また、実務での具体的な使用方法や管理および運用に関する事項は、各製品のガイドを参照してください。

参考

Tmaxシステムの開発に関する基本的な内容は『Tmax 運用ガイド』や『Tmax アプリケーション開発ガイド』を参照してください。Tmaxで提供するコマンドとC APIについての説明は『Tmax リファレンスガイド』を参照してください。

本書の構成

本書は3つの章から構成されています。

各章の主な内容は以下のとおりです。

- 第1章: 環境設定

JTCを使用するための環境設定とログ設定の方法について説明します。

- 第2章: JTCのクラスと関数

JTCで利用できるクラスと関数について説明します。

- 3章: JTCモニタリング

JTCのモニタリングに必要な設定とモニタリング方法について説明します。

表記上の規則

表記	意味
<AaBbCc123>	プログラム・ソースコードのファイル名、ディレクトリー
<Ctrl>+C	CtrlキーとCキーを同時に押す
[Button]	GUIのボタン、メニュー名
太字	強調
「」、『』（鍵カッコ）	関連文書、あるいはガイド内の他の章および節の表示
「入力項目」	画面UI上の入力項目
<ハイパーリンク>	メール・アカウント、Webサイト
>	メニューの実行順
+----	下位ディレクトリー/ファイル有り
----	下位ディレクトリー/ファイル無し
参考	参照/注意事項
[図 1.1]	図の名称
[表 1.1]	表の名称
AaBbCc123	コマンド、コマンド実行結果の画面出力、サンプル・コード
[]	オプション・パラメータ値
	選択パラメータ値

関連文書

ガイド	説明
Tmax 運用ガイド	Tmaxを利用するための環境設定ファイルとシステム運用方法について説明しています
Tmax アプリケーション開発ガイド	Tmaxアプリケーション・プログラムの開発で使用するAPIの概念と使用方法および例について説明しています
Tmax リファレンスガイド	Tmaxアプリケーションの開発に使用するコマンドおよびクライアントとサーバーの接続、通信に使用する関数の使用方法と例について説明しています

参考文献

製品	ガイド
Tuxedo	Tuxedo Adminsitration Guide

お問合せ先

Korea

TmaxSoft Co., Ltd.
45, Jeongjail-ro, Bundang-gu,
Seongnam-si, Gyeonggi-do, 13613
South Korea
Tel: +82-31-8018-1000
Fax: +82-31-8018-1115
Email: info@tmax.co.kr
Web (Korean): <http://www.tmaxsoft.com>
TechNet: <http://technet.tmaxsoft.com>

USA

TmaxSoft Inc.
101 North Wacker Drive, Suite 2014,
Chicago, IL 60606
U.S.A
Tel: +1-312-525-8330
Email: info@tmaxsoft.com
Web (English): http://www.tmaxsoft.com/us_en/home

Japan

TmaxSoft Japan Co., Ltd.
5F Sanko Bldg, 3-12-16 Mita, Minato-Ku, Tokyo, 108-0073
Japan
Tel: +81-3-5765-2550
Fax: +81-3-5765-2567
Email: info@tmaxsoft.co.jp
Web (Japanese): <http://www.tmaxsoft.co.jp>

China

Beijing TmaxSoft System Software Co., Ltd.
Room103, No.2 Huizhong Building, Seven Street Shangdi,
Haidian District, Beijing, 100085
P.R.China
Tel: +86-10-6298-8827
Email: info@tmaxsoft.com.cn
Web (Chinese): http://www.tmaxsoft.com/cn_en/home_cn_en

Brazil

Tmax Brasil Sistemas e Serviços Ltda.
Av. Copacabana, 177, sala 32~35 Empresarial 18 do Fortel
Alphaville Barueri, Sao Paulo, 06472-001
Brazil
Tel: +55-11-4191-3100
Fax: +55(11) 4191-3705 (extension#112)
Email: info.bra@tmaxsoft.com
Web (Portuguese): http://www.tmaxsoft.com/br_en/home_br_en

Russia

Tmax Rus L.L.C.
Leninsky prospekt, 113/1 (Park Place Moscow),
Office 318e, Moscow, 117198
Russia
Tel: +7(495)970-01-35
Email: info.rus@tmaxsoft.com
Web (Russian): http://www.tmaxsoft.com/ru_ru/home_ru_ru

Singapore

Tmax Singapore Pte. Ltd.
430 Lorong 6, Toa Payoh #10-02,
OrangeTee Building, 319402
Singapore
Tel: +65-6259-7223
Fax: +65-6258-7112
Email: info.sg@tmaxsoft.com
Web (English): http://www.tmaxsoft.com/sg_en/home_sg_en

United Kingdom

TmaxSoft UK Ltd.
215 Knyvett House, Watermans Business Park,
The Causeway, Staines TW18 3BAB
United Kingdom
Tel: +44-1784-895005
Email: info.uk@tmaxsoft.com
Web (English): http://www.tmaxsoft.com/gb_en/home_gb_en

Canada

TmaxSoft Canada, Inc.
2425 Matheson Blvd East, 8th floor,
Unit 824 Mississauga, ON, L4W 5K4
Canada
Tel: +1-905-361-2888
Email: info.canada@tmaxsoft.com
Web (English): http://www.tmaxsoft.com/ca_en/home_ca_en

Australia

TmaxSoft Proprietary Limited
L32, 101 Miller Street, North Sydney 2060
Australia
Tel: +91-9845-330-704
Email: info.aus@tmaxsoft.com
Web (English): http://www.tmaxsoft.com/au_en/home_au_en

India

TmaxSoft Technologies Private Limited
Sobha Alexander Plaza, 3rd Floor,
16/2 Commissariat Road, Bangalore-560025
India
Tel: +91-9845-330-704
Email: info.india@tmaxsoft.com
Web (English): http://www.tmaxsoft.com/in_en/home_in_en

Turkey

TmaxSoft Co., Ltd. Turkey Liaison Office
Windowist Tower. Eski Buyukdere Cad. No:26,
Maslak 34467 Istanbul
Turkey
Tel: +90-544-553-6045
Email: cslee@tmaxsoft.com
Web (English): http://www.tmaxsoft.com/tr_en/home_tr_en

第1章 環境設定

本章では、JTC環境設定とログ設定の方法について説明します。

1.1. JTCアウトバウンド通信

JTCはJEUS-Tuxedo Connectorの略語で、Tuxedoに接続してサービスを利用したり、Tuxedoからサービスを呼び出したりすることができるライブラリーです。WebTライブラリーに含まれて配布されます。

JTC環境設定ファイルにはJEUSサーバーおよびTuxedoサーバーの情報とサービス運用環境を設定します。JTC環境設定ファイルに設定された情報とTuxedoのドメイン・ゲートウェイに設定された情報は一致する必要があります。

JTCに対応するTuxedoの環境設定ファイルには、ドメイン・ゲートウェイを使用するために**SVRGROUPセクション**および**SERVERセクション**に付加的な内容を追加します。ドメインについての詳細情報は、別途のドメイン環境設定ファイルを設定して使用します。

JTCアウトバウンド通信の環境設定は、Tuxedoドメインおよびその他の設定のためのものであり、以下の順序で設定します。

1. webt.propertiesの設定
2. Startupメソッドの設定
3. Tuxedoの環境設定
4. Tuxedoドメインの環境設定
5. JEUSとTuxedoの起動

1.1.1. webt.propertiesの設定

webt.propertiesファイルはログ情報とドメイン情報を設定するファイルであり、次のパスの配下に存在します。

```
$JEUS_HOME/lib/application
```

注

JEUS_HOME/lib/application/webt.propertiesとJEUSMain.xmlで同時にwebt.propertiesを設定すると、Tuxedoは2つのうち1つのみに接続を許容するため、もう一方は接続が切断され続ける現象が発生する

ので注意してください。webt.propertiesの設定に関する内容は『Tmax WebTユーザガイド』を参照してください。

以下は、webt.propertiesファイルの基本形式と設定項目についての説明です。

```
log.level = debug
log.dir = c:/temp
log.file = webt.log
log.bufsize = 1024
defaultCharset = euc-kr

tux.buffer.size = 4096
tux.default.timeout = 20
tux.default.txttimeout = 30

tux.local.name = TDOM1
tux.remote.name.list = TDOM2
tux.default.remote = TDOM2

tux.TDOM2.addr = 192.0.0.1
tux.TDOM2.port = 5000
```

- log.level = [none | debug | info]
 - デフォルト値: info
 - ログレベルを指定します。
- log.dir = string
 - デフォルト値: NULL
 - ログファイルが格納されているディレクトリーを設定します。Windowsの場合、「/」を「\」で表示します。
- log.file = string
 - デフォルト値: webt.log
 - ログファイルの名前を指定します。
- log.bufsize = int
 - デフォルト値: 512
 - ログを残す際に使用されるバッファのサイズを指定します。

- `log.file.date.format = string`
 - デフォルト値: `MMddyyyy`
 - `log.valid.days`が1以上の場合に作成されるログファイル名の形式を指定します。
- `log.valid.days = int`
 - デフォルト値: `-1`
 - ログファイルを使用する場合に何日周期でファイル名をアップデートするかを指定します。
0や負数の値を指定した場合、1つのファイルを使用することになります。
- `tux.local.name = string`
 - デフォルト値: `NULL`
 - JEUSのローカル・ドメインの名前です。Tuxedoのドメイン・ゲートウェイはこの値を使用して相手を区分し認識するため、JTCが使用される各エンジン・コンテナは必ず固有の名前を持っている必要があります。
- `tux.remote.name.list = string`
 - デフォルト値: `NULL`
 - リモート・ドメインとして使用する名前を指定します。複数を指定する場合はコンマ(,)で区分して設定します。この環境に設定された名前は下の設定の<domainName>に対応します。
- `tux.cluster.name.list = string`
 - デフォルト値: `NULL`
 - クラスター・ドメインとして使用する名前を指定します。複数を指定する場合はコンマ(,)で区分して設定します。
- `tux.<clusterName>.remote.name.list = string`
 - デフォルト値: `NULL`
 - 該当クラスターの下で管理されるリモート・ドメインを指定します。クラスターに要求を呼び出した場合、このクラスターに属するリモート・ドメインのうち1つに呼び出されます。複数を指定する場合はコンマ(,)で区分して設定します。

- `tux.buffer.size = int`
 - デフォルト値: 4096(単位 : バイト)
 - Tuxedoとの通信に使用されるバッファのサイズを設定します。
- `tux.default.timeout = int`
 - デフォルト値: 1000(単位 : 秒)
 - 一般サービスを呼び出す際に使用される応答待機時間です。設定時間を超過した場合、クライアントは例外処理され、以降に受け取った応答メッセージは破棄されます。
- `tux.default.txtimeout = int`
 - デフォルト値: 30(単位 : 秒)
 - トランザクション制御メッセージの応答待機時間です。Tuxedoに設定されているトランザクション・タイムアウト時間よりも大きい値を設定する必要があります。
 - 一般的な設定値は、Tuxedoの環境設定ファイルに設定されているTXTIMEOUT時間+SCANUNIT値です。
- `tux.default.table = string`
 - デフォルト値: NULL
 - Tuxedoのフィールド表ファイルを指定します。
- `tux.default.table.file = string`
 - デフォルト値: NULL
 - Tuxedoのフィールド表テキストファイル(*.f)を指定します。Tuxedo 5以下で使用します。
- `tux.default.table.file.16 = string`
 - デフォルト値: NULL
 - Tuxedoのフィールド表テキストファイル(*.f)を指定します。Tuxedo 6以上で使用します。
- `tux.<domainName>.buffer.size = int`

- デフォルト値: -1 (単位: バイト)
- 該当ドメインの通信に使用されるバッファのサイズを指定します。
- `tux.<domainName>.default.timeout = int`
 - デフォルト値: -1 (単位: 秒)
 - 該当ドメインに関して一般サービスを呼び出す際に使用される応答待機時間を指定します。
- `tux.<domainName>.default.txtimeout = int`
 - デフォルト値: -1 (単位: 秒)
 - 該当ドメインのトランザクション制御メッセージの応答待機時間を指定します。
- `tux.<domainName>.default.readtimeout = int`
 - デフォルト値: -1 (単位: 秒)
 - 該当ドメインのソケット・タイムアウト時間を指定します。
- `tux.<domainName>.thread.min = int`
 - デフォルト値: 1
 - 該当ドメインの基本接続スレッド数を指定します。
- `tux.<domainName>.thread.max = int`
 - デフォルト値: 1
 - 該当ドメインの最大接続スレッド数を指定します。
- `tux.<domainName>.addr[.0] = string`
 - デフォルト値: NULL
 - 該当Tuxedoドメインのメイン・ゲートウェイのIPアドレスを指定します。
- `tux.<domainName>.port[.0] = int`
 - デフォルト値: 0

- 該当Tuxedoドメインのメイン・ゲートウェイのポート番号を指定します。

- `tux.<domainName>.interval[.0] = int`
 - デフォルト値: 20(単位 : 秒)

 - 該当Tuxedoドメインのメイン・ゲートウェイのスケジューリング周期を指定します。

- `tux.<BackUp domain Name>.addr[.x] = string`
 - デフォルト値: NULL

 - 該当Tuxedoドメインのバックアップ・ゲートウェイのIPアドレスを指定します。

- `tux.<BackUp domain Name>.port[.x] = int`
 - デフォルト値: 0

 - 該当Tuxedoドメインのバックアップ・ゲートウェイのポート番号を指定します。

- `tux.<domainName>.interval[.x] = int`
 - デフォルト値: 20(単位 : 秒)

 - 該当Tuxedoドメインのバックアップ・ゲートウェイのスケジューリング周期を指定します。

- `tux.<domainName>.svc = string`
 - デフォルト値: NULL

 - 該当Tuxedoドメインで呼び出すTuxedoサービス名を指定します。複数を指定する場合はコンマ(,)で区分して設定します。

- `tux.<clusterName>.svc = string`
 - デフォルト値: NULL

 - クラスター・ドメインが設定されている場合、リモート・ドメインにサービスを指定する代わりに、クラスター名に該当するサービスを指定します。このように設定された場合、該当クラスターの下にあるリモート・ドメインを使用して、指定されたサービスを呼び出すことができます。複数を指定する場合はコンマ(,)で区分して設定します。

- tux.<BackUp domain Name>.notx = boolean
 - デフォルト値: false
 - 該当Tuxedoドメインのnot-transactionモードの設定可否を指定します。

- tux.<BackUp domain Name>.backup = string
 - デフォルト値: NULL
 - 該当Tuxedoドメインのバックアップ・ドメインを指定します。ここに設定されたゲートウェイはメインのドメイン・アドレスと異なる必要があり、上のバックアップ・ゲートウェイ機能と同時に設定してはいけません。

- tux.mbean = boolean
 - デフォルト値: false
 - JTC AdminのためのMBeanを登録するかどうかを指定します。JTC Admin機能は、JEUS v6.0 Fix#7以上でサポートします。

以下は、各状況別の設定例のファイルです。

- Mirrorを指定する場合

```
tux.local.name=LJEUS1
tux.remote.name.list=TJEUS1

#### Main
tux.TJEUS1.addr=160.61.1.121
tux.TJEUS1.port=40001
tux.TJEUS1.retry=0
tux.TJEUS1.interval=10

#### Mirror 1
tux.TJEUS1.addr.1=160.61.1.122
tux.TJEUS1.port.1=40001
tux.TJEUS1.retry.1=0
tux.TJEUS1.interval.1=10

#### Mirror 2
tux.TJEUS1.addr.2=160.61.1.131
tux.TJEUS1.port.2=40001
tux.TJEUS1.retry.2=0
tux.TJEUS1.interval.2=10
```

- Backupを指定する場合

```
tux.local.name=TDOM1
tux.remote.name.list=TDOM2,TDOM3
tux.default.remote=TDOM2

tux.TDOM2.addr=192.0.0.1
tux.TDOM2.port=5000
tux.TDOM2.backup=TDOM3

tux.TDOM3.addr=192.0.0.2
tux.TDOM3.port=5000
```

- クラスターを使用する場合

```
tux.cluster.name.list=RCDOM1,RCDOM2
tux.RCDOM1.remote.name.list=TDOM2,TDOM3
tux.RCDOM2.remote.name.list=TDOM2

tux.RCDOM1.svc=TOUPPER,TOUPPER2,FDLTOUPPER,FDLTOUPPER2
tux.RCDOM2.svc=TOUPPER,FDLTOUPPER

tux.TDOM2.addr=192.0.0.1
tux.TDOM2.port=5000

tux.TDOM3.addr=192.0.0.2
tux.TDOM3.port=5000
```

- POJOを使用する場合(インバウンド)

```
tux.beans.service.list=TOUPPER,TOLOWER
tux.beans.TOUPPER.class=tmax.qa.sample.JtcBean1
tux.beans.TOLOWER.class=tmax.qa.sample.JtcBean2
```

- EJBを使用する場合(インバウンド)

```
tux.ejbs.service.list= GSVC01
tux.ejbs.GSVC01.export=echotest
tux.ejbs.GSVC01.method=setStringEchoTux
```

参考

WebTの環境設定に関する詳細は『Tmax WebTユーザガイド』を参照してください。

1.1.2. Startupメソッドの設定

JTCを以下のようにエンジン環境設定ファイルのStartupクラスに登録します。1つのエンジン・コンテナには必ず1つのJTCのみを起動します。

以下は、JEUSMain.xmlでStartupメソッドを設定する部分です。

```
<lifecycle-invocation>
  <class-name>tmax.webt.TuxBootstrapper</class-name>
  <invocation>
    <invocation-method>
      <method-name>init</method-name>
    </invocation-method>
  </invocation>
</lifecycle-invocation>
```

上のように設定した場合、自動的にアプリケーション配下のwebt.propertiesファイルを環境設定ファイルに選択します。

多数のエンジン・コンテナで使用される場合、各コンテナは別途の環境ファイルを必要とします。つまり、以下のように各エンジン・コンテナに対して別途の設定を持つコネクションが必要です。

```
<lifecycle-invocation>
  <class-name>tmax.webt.TuxBootstrapper</class-name>
  <invocation>
    <invocation-method>
      <method-name>init</method-name>
      <method-params>
        <method-param> java.lang.String </method-param>
      </method-params>
    </invocation-method>
    <invocation-argument> config-filename </invocation-argument>
  </invocation>
</lifecycle-invocation>
```

JTCは基本的にTuxedoドメイン・ゲートウェイと1:1ソケット通信を行うため、JEUSのクラスタリング機能を使用したTuxedoサービスの共有は不可能です。

注

上記のような設定により、JTCのための環境ファイルを別途指定した状態で、別のJEUS_HOME/lib/application/webt.propertiesが存在したり、あるいはJEUSMain.xmlの<node>下の<command-option>ノードで-Dwebt.properties=<config-filename>のようにwebt.propertiesファイルを別途設定している場合には、JTCと環境ファイルが重複するため、設定が正しく適用されないことがあります。

webt.propertiesファイルは常に1つのコンテナで単独で使用する必要があることに注意してください。

1.1.3. Tuxedoの環境設定

SERVERSセクションに、ドメイン管理サーバー、ゲートウェイ管理サーバー、ドメイン・ゲートウェイ・サーバーを登録します。Tuxedo 7.2以上の場合、CLOPTオプションに[-t]を必ず追加します。

以下は、SERVERSセクションに前述のサーバーを登録する例です。

```
*SERVERS

DMADM          SRVGRP = GROUP1 SRVID = 1
                CLOPT = "-At -r -o /user/tux/log/DMADM.out"
GWADM          SRVGRP = GROUP2 SRVID = 1
                CLOPT = "-At -r -o /user/tux/log/GWADM.out"
GWTDOMAIN      SRVGRP = GROUP2 SRVID = 2
                CLOPT = "-At -r -o /user/tux/log/GWTDOMAIN.out"
```

1.1.4. Tuxedoドメインの環境設定

Tuxedoドメインの環境設定ファイルでは、Tuxedoローカル・ドメインとJTCがインストールされたJEUSリモート・ドメイン、Tuxedoで呼び出すことになるJEUSのサービス名を設定します。

以下は、Tuxedoドメインの環境ファイルを設定する部分です。

```
*DM_LOCAL_DOMAINS
TDOM1      GWGRP = GROUP2
           TYPE = TDOMAIN
           DOMAINID = "TDOM1"
           BLOCKTIME = 20
           CONNECTION_POLICY = INCOMING_ONLY
           DMTLOGDEV = "/user/tux/log/TLOG"
           AUDITLOG = "/user/tux/log/ALOG"
           DMTLOGNAME = "DMTLOG_TDOM1"

*DM_REMOTE_DOMAINS
TDOM2      TYPE = TDOMAIN
           DOMAINID = "TDOM2"

*DM_TDOMAIN
TDOM1      NWADDR = "//211.56.251.175:5000"

*DM_REMOTE_SERVICES
TOLOWER    RDOM = "TDOM2"
JEUSXA     RDOM = "TDOM2"
```

上の項目のうちCONNECTION_POLICYとDMTLOGDEVでは以下の事項に注意してください。

項目	注意事項
CONNECTION_POLICY	INCOMMING_ONLYを必ず使用します
DMTLOGDEV	グローバル・トランザクションを使用するためにDMTLOGDEVにTLOGデバイス・ファイルを設定します

JTCは単方向の接続のみをサポートするため、JTCに該当するドメインのリモート・ドメインのアドレス値は意味がありません。したがって該当項目は設定しなくても構いません。Tuxedoを初めて起動する際、アドレス値が設定されていないというメッセージがULOGに残ることがありますが、無視しても構いません。

Tuxedoの環境設定ファイルでローカル・ドメインに該当するドメインのアドレス値は、JTCに設定されるリモート・ドメインのアドレス値と一致する必要があります。JTCのローカル・ドメインの名前は、Tuxedoの環境設定ファイルのリモート・ドメインの名前と一致する必要があります。

トランザクションを使用するためのTLOGDEVファイルは以下のように作成します。

```
/user/tux>tmadmin -c

tmadmin - Copyright (c) 1996 BEA Systems, Inc.
Portions * Copyright 1986-1997 RSA Data Security, Inc.
All Rights Reserved.
Distributed under license by BEA Systems, Inc.
TUXEDO is a registered trademark.

> crdl -z <tlogfilepath> [-b <blocksize>] [-o <configoffset>]
   crdl -z /user/tux/log/TLOG
```

1.1.5. JEUSとTuxedoの起動

アプリケーションの開発および環境設定が完了すると、JEUSとTuxedoを起動します。

以下は、環境設定ファイルをTuxedoシステムに適用し、起動する手順です。

```
tmloadcf -y <TuxedoConfigurationFile>
dmloadcf -y <TuxedoDomainEnvironmentFile>
tmboot -y
```

以下は、JEUSシステムを起動するコマンドです。

```
jeus -xml -U <adminuser> -P <password>
```

1.2. JTCインバウンド通信

JTCはTuxedoからJEUSへの通信をサポートします。TuxedoのTDomain GatewayとJTCの通信により、JEUSのEJBや一般クラス(Bean)を呼び出すことができます。

実際の動作は、Tuxedoのクライアント、サーバー・アプリケーションでtpcallでTDomain Gatewayに登録したサービス呼び出すと、当該TDomain Gatewayを通じてJTCにメッセージが転送され、JTCはwebt.propertiesに設定されたtpcallの引数のうち、サービス名に該当するEJBや一般クラスを呼び出します。

1.2.1. EJBサービスの呼び出し設定

JTCインバウンド通信でEJBサービス呼び出すには、以下のような設定が必要です。

1. webt.propertiesの設定
2. Tuxedoの環境設定
3. Tuxedoドメインの環境設定
4. JEUSとTuxedoの起動

注

JTCでEJBを呼び出すには、EJBを「shared type」にデプロイする必要があります。

webt.propertiesの設定

以下は、webt.propertiesファイルの基本形式と設定項目についての説明です。

```
tux.ejbs.service.list = <ServiceName>
tux.ejbs.<ServiceName>.export = <EJB module>
tux.ejbs.<ServiceName>.method = <method>
```

- tux.ejbs.service.list = string
 - デフォルト値 : NULL
 - EJBサービス呼び出すためにTuxedoで認識するすべてのサービス名をコンマ(,)で区切って記述します。

サービス名はTuxedoのクライアントとサーバーでtpcallする場合に使用するサービス名です。当該サービス名が呼び出されると、exportに設定したEJBのメソッドが呼び出されます。メソッドは、引数と戻り値がtmax.jtc.io.TuxBufferを使用する必要があります。

- tux.ejbs.<ServiceName>.export = string

- デフォルト値 : NULL
- 当該サービスを呼び出すときに使用されるEJBモジュール名を設定します。
- `tux.ejbs.<ServiceName>.method = string`
 - デフォルト値 : NULL
 - サービスに該当するEJBモジュールのメソッド名を設定します。

以下は、`webt.properties`の設定例です。

```
tux.ejbs.service.list = GSVC01
tux.ejbs.GSVC01.export = EchoTest
tux.ejbs.GSVC01.method = setStringEchoTux
```

TuxedoでGSVC01サービスを呼び出して、JEUSのEchoTest EJBのsetStringEchoTux関数を呼び出します。

Tuxedoの環境設定

SERVICEセクションに以下のようにサービスを登録します。

```
*SERVICE
GSVC01
```

Tuxedoドメインの環境設定

Tuxedoドメインの環境設定ファイルのDM_REMOTE_SERVICESセクションに、Tuxedoで呼び出すJEUSのサービス名を以下のように設定します。

```
*DM_REMOTE_SERVICES
GSVC01
    RNAME = "GSVC01"
    RDOM = "TUXGW1"

*DM_TDOMAIN
TUXGW1 NWADDR = "//192.168.37.100:9521"
```

JEUSとTuxedoの起動

アプリケーションの開発および環境設定が完了すると、JEUSとTuxedoを起動します。起動する方法は、JTCアウトバウンド通信の方法と同じです。詳細は「[1.1.5. JEUSとTuxedoの起動](#)」を参照してください。

1.2.2. Beanサービスの呼び出し設定

JTCインバウンド通信でBeanサービスを呼び出すには、以下のような設定が必要です。

1. webt.propertiesの設定
2. Tuxedoの環境設定
3. Tuxedoドメインの環境設定
4. JEUSとTuxedoの起動

webt.propertiesの設定

以下は、webt.propertiesファイルの基本形式と設定項目についての説明です。

```
tux.beans.service.list = <ServiceName>
tux.beans.<ServiceName>.class = <ClassName>
```

- tux.beans.service.list = string
 - デフォルト値 : NULL
 - Beanサービスを呼び出すためにTuxedoで認識するすべてのサービス名をコンマ(,)で区切って記述します。

サービス名はTuxedoのクライアントとサーバーでtpcallする場合に使用するサービス名です。すべての一般クラスを呼び出せるわけではなく、tmax.jtmax.JtmaxServiceインターフェースを実装したクラスのみ処理することができます。

Tuxedoでサービスを呼び出すと、JTCでは登録した当該クラスをReflection APIを使って作成し、JtmaxServiceのインターフェースのサービス関数を呼び出します。このオブジェクトは一度作成した後は再使用します。

JtmaxServiceのインターフェースは以下のとおりです。

インターフェース	説明
Void initialize()	初めてオブジェクトを作成するときに呼び出す関数です

インターフェース	説明
Void destroy()	呼び出しません。この関数は終了時に自動で呼び出される関数です
BufferAccess service(BufferAccess)	サービスで実行する動作を実装します。引数はTuxBuffer型で入り、TuxBuffer型を返します

実装したクラスをJARファイルに作成してJEUSのlib/applicationの配下に配置すると、JEUSが起動時にクラス・ローダーに含まれるため、JTCで当該クラスのオブジェクトを作成することができます。

- tux.beans.<ServiceName>.class = string
 - デフォルト値 : NULL
 - 当該サービスを呼び出すときに使用されるクラス名を設定します。

以下は、webt.propertiesの設定例です。

```
tux.beans.service.list = GSVC02
tux.beans.GSVC02.class = test.JtcBeanSample
```

Tuxedoの環境設定

SERVICEセクションに以下のようにサービスを登録します。

```
*SERVICE
GSVC02
```

Tuxedoドメインの環境設定

Tuxedoドメインの環境設定ファイルのDM_REMOTE_SERVICESセクションに、Tuxedoで呼び出すJEUSのサービス名を以下のように設定します。

```
*DM_REMOTE_SERVICES
GSVC02
    RNAME = "GSVC02"
    RDOM  = "TUXGW1"

*DM_TDOMAIN
TUXGW1  NWADDR = "//192.168.37.100:9521"
```

設定例

<JtcBeanSample.java>

```
package tmax.qa.sample;

import tmax.common.BufferAccess;
import tmax.jtc.io.TuxStringBuffer;
import tmax.jtmax.JtmaxService;
import tmax.webt.WebtException;
import tmax.webt.WebtSystem;

public class JtcBeanSample implements JtmaxService{

    public void destroy() throws Exception {

    }

    public void initialize() throws Exception {

    }

    public BufferAccess service(BufferAccess tux) throws WebtException {
        TuxStringBuffer tuxbuf = (TuxStringBuffer)tux;
        System.out.println("received string value : " + tuxbuf.getString());
        TuxStringBuffer ret = new TuxStringBuffer(WebtSystem.getDefaultCharset());
        ret.setString(tuxbuf.getString().toUpperCase());
        return ret;
    }

}
```

JEUSとTuxedoの起動

アプリケーションの開発および環境設定が完了すると、JEUSとTuxedoを起動します。起動する方法は、JTCアウトバウンド通信の方法と同じです。詳細は「[1.1.5. JEUSとTuxedoの起動](#)」を参照してください。

1.2.3. サービス関数内でのメッセージの作成方法

本節では、以下のサービス関数内でメッセージを作成する方法を説明します。

- **TuxStringBuffer**

TuxStringBufferは、STRING型のメッセージを構成するときに使用します。


```
TuxStringBuffer tux = new TuxStringBuffer(WebtSystem.getDefaultCharset());
```

構築子内部の引数はcharset-stringで指定します。setStringメソッドを使って文字列を入力し、getStringメソッドを使って文字列を参照します。詳細は「[2.6. TuxStringBuffer/TuxCarrayBufferクラス](#)」を参照してください。

- **TuxCarrayBuffer**

TuxCarrayBufferは、CARRAY型のメッセージを構成するときに使用します。

```
TuxCarrayBuffer tux = new TuxCarrayBuffer(WebtSystem.getDefaultCharset());
```

構築子内部の引数はcharacter-stringで指定します。setBytesメソッドを使ってbyte[]を入力し、getBytesメソッドを使ってbyte[]で参照します。詳細は「[2.6. TuxStringBuffer/TuxCarrayBufferクラス](#)」を参照してください。

- **TuxFieldBuffer**

FML型のメッセージを構成するときに使用します。

```
TuxFieldBuffer tux = new TuxFieldBuffer(true);
```

構築子内部の引数は、fml32を使用する場合は「true」、使用しない場合は「false」に指定します。createField、getFieldメソッドを使って入力、参照を行います。WebtFieldBufferと使用方法が同じです。詳細は「[2.7. TuxFieldBufferクラス](#)」を参照してください。

1.3. ログ設定

1.3.1. Tuxedoの設定

Tuxedoの環境変数設定とログは以下のとおりです。

環境変数の設定

以下の環境変数をTuxedo環境ファイルに設定するか、Linuxカーネルで入力します。

```
export TMTRACE = on
```

ログ

Tuxedoの起動後、サービス実行内容がULOGに保存されます。

```
tail -f ULOG.111604
```

```

172503.Flik!GWTDOMAIN.262612: LIBGWT_CAT:1130: INFO: Disconnected from domain
(domainid=<ARTEMIS>)
172514.Flik!GWTDOMAIN.262612: LIBGWT_CAT:1130: INFO: Disconnected from domain
(domainid=<APOLLON>)
172605.Flik!GWTDOMAIN.262612: LIBGWT_CAT:2005: INFO: Receive connecting request
from remote domain (domainid<ARTEMIS>)
172605.Flik!GWTDOMAIN.262612: LIBGWT_CAT:1128: INFO: Connection accepted
from domain (domainid=<ARTEMIS>)
172614.Flik!GWTDOMAIN.262612: LIBGWT_CAT:2005: INFO: Receive connecting request
from remote domain (domainid<APOLLON>)
172614.Flik!GWTDOMAIN.262612: LIBGWT_CAT:1128: INFO: Connection accepted
from domain (domainid=<APOLLON>)
172134.Flik!qbb01.57514: 11162004: TUXEDO Version 6.5 AIX  2 4 007025954C00.
172134.Flik!qbb01.57514: LIBTUX_CAT:262: INFO: Standard main starting
172135.Flik!qbb01.57514: TRACE:at:  { tpsvrinit(20, "qbb01 -g 24 -i 3100 -u Flik
-U /log/tuxs/ULOG -m 0 -A -r -e /log/tuxs/stderr -o /log/svc/qbb01 -- -I port1024")
172135.Flik!qbb01.57514: ====> Tring to Connect Database .....
172135.Flik!qbb01.57514:          Connecting DB succeed  SQLCODE[0]
172135.Flik!qbb01.57514: TRACE:at:  } tpsvrinit = 0
172135.Flik!qbb01.123172: 11162004: TUXEDO Version 6.5 AIX  2 4 007025954C00.
172135.Flik!qbb01.123172: LIBTUX_CAT:262: INFO: Standard main starting
172135.Flik!qbb01.123172: TRACE:at:  { tpsvrinit(20, "qbb01 -g 24 -i 3101 -u Flik
-U /log/tuxs/ULOG -m 0 -A -r -e /log/tuxs/stderr -o /log/svc/qbb01 -- -I port1024")
172135.Flik!qbb01.123172: ====> Tring to Connect Database .....
172135.Flik!qbb01.123172:          Connecting DB succeed  SQLCODE[0]
172135.Flik!qbb01.123172: TRACE:at:  } tpsvrinit = 0
172141.Flik!xxc01.220310: 11162004: TUXEDO Version 6.5 AIX  2 4 007025954C00.
172141.Flik!xxc01.220310: LIBTUX_CAT:262: INFO: Standard main starting
172141.Flik!xxc01.220310: TRACE:at:  { tpsvrinit(17, "xxc01 -g 91 -i 9170 -u Flik
-U /log/tuxs/ULOG -m 0 -A -r -e /log/tuxs/stderr -o /log/svc/xxc01")
172141.Flik!xxc01.220310: TRACE:at:  { tpopen()
172141.Flik!xxc01.220310: TRACE:at:  } tpopen = 1
172141.Flik!xxc01.220310: TRACE:at:  } tpsvrinit = 0
172814.Flik!xxc01.220310: gtrid x0 x41985b26 x182:
TRACE:at: { tpSERVICE({"myasset_login", 0x10, 0x200dab20, 1296, 0, -1, {0, -2,
-1}})
172814.Flik!xxc01.220310: gtrid x0 x41985b26 x182:
TRACE:at:  { tprealloc(0x200dab20, 65535)
172814.Flik!xxc01.220310: gtrid x0 x41985b26 x182:
TRACE:at:  } tprealloc = 0x200ddc40
172814.Flik!xxc01.220310: gtrid x0 x41985b26 x182:
TRACE:at:  { tpreturn(2, 0, 0x200ddc40, 0, 0x0)
172814.Flik!xxc01.220310: TRACE:at:  } tpreturn [long jump]
172814.Flik!xxc01.220310: TRACE:at:  } tpSERVICE
172816.Flik!qbb01.57514: TRACE:at:  { tpSERVICE({"bb_04900_q1", 0x0, 0x200be3b0,

```

```

1264, 0, -1, {0, -2, -1}})
172816.Flik!qbb01.57514: TRACE:at:      { tprealloc(0x200be3b0, 65535)
172816.Flik!qbb01.57514: TRACE:at:      } tprealloc = 0x200c14d0
172816.Flik!qbb01.57514: TRACE:at:      { tpreturn(2, 0, 0x200c14d0, 0, 0x0)
172816.Flik!qbb01.57514: TRACE:at:      } tpreturn [long jump]
172816.Flik!qbb01.57514: TRACE:at:      } tpSERVICE
172845.Flik!qbb01.123172: TRACE:at:      { tpSERVICE({"bb_04900_q1", 0x0, 0x200be3b0,

1264, 0, -1, {0, -2, -1}})
172845.Flik!qbb01.123172: TRACE:at:      { tprealloc(0x200be3b0, 65535)
172845.Flik!qbb01.123172: TRACE:at:      } tprealloc = 0x200c14d0
172845.Flik!qbb01.123172: TRACE:at:      { tpreturn(2, 0, 0x200c14d0, 0, 0x0)
172845.Flik!qbb01.123172: TRACE:at:      } tpreturn [long jump]
172845.Flik!qbb01.123172: TRACE:at:      } tpSERVICE
172846.Flik!xxc01.220310: gtrid x0 x41985b26 x183:
TRACE:at: { tpSERVICE({"myasset_login", 0x10, 0x200f40c0, 1296, 0, -1, {0, -2,
-1}})
172846.Flik!xxc01.220310: gtrid x0 x41985b26 x183:
TRACE:at:      { tprealloc(0x200f40c0, 65535)
172846.Flik!xxc01.220310: gtrid x0 x41985b26 x183:
TRACE:at:      } tprealloc = 0x200f40c0
172846.Flik!xxc01.220310: gtrid x0 x41985b26 x183:
TRACE:at:      { tpreturn(2, 0, 0x200f40c0, 0, 0x0)
172846.Flik!xxc01.220310: TRACE:at:      } tpreturn [long jump]
172846.Flik!xxc01.220310: TRACE:at:      } tpSERVICE

```

1.3.2. WebTの設定

WebTの環境変数の設定とログは以下のとおりです。

環境変数の設定

webt.propertiesにログ関連の環境変数を設定します。

```

log.level = debug
log.dir = /user/jeus/logs/
log.file = webtJtc.log
log.bufsize = 1024

```

ログ

以下はWebT.logの内容です。

```

tail -f webtJtc.log

[2004.11.16 17:38:06] [I] WebT(version:3.5.6.5) logger start

```

[illegible]

```
[NAVIS] commit : f7 27 cf 40 00 00 00 00 00 00 00 00 [0]::0
[NAVIS] prepare : f7 27 cf 40 00 00 00 00 00 00 00 00 [0]::0
[2004.11.16 19:03:37] 160.61.194.107 "GET /test/XATest.jsp?cnt=1" 200 512ms
[2004.11.16 19:03:38] 160.61.194.107 "GET /test/NonXATest.jsp?cnt=1" 200 231ms
[2004.11.16 19:03:51] 160.61.194.107 "GET /test/TuxTest1.jsp" 200 1ms
TuxDomain : [FOCUS_TEST]
[NAVIS] started : f7 27 cf 40 01 00 00 00 00 00 00 00 [0]::0
[NAVIS] end : f7 27 cf 40 01 00 00 00 00 00 00 00 [0]::0
[NAVIS] commit : f7 27 cf 40 01 00 00 00 00 00 00 00 [0]::0
[NAVIS] prepare : f7 27 cf 40 01 00 00 00 00 00 00 00 [0]::0
```


第2章 JTCのクラスと関数

本章では、JTCのクラスと関数について説明します。

2.1. 概要

以下は、JTCが提供するクラス一覧です。

クラス	説明
TuxService	Tuxedoのサービスを実行するための関数を提供するクラスです
TuxAsyncService	非同期リスナー構造をサポートするためのTuxServiceの継承クラスです
TuxRemoteService Manager	クラスター・ドメインでサービスを呼び出すためのクラスです
TuxBuffer	Tuxedoとデータを送受信するために使用されるTuxedoデータ・バッファの上位クラスです
TuxStringBuffer TuxCarrayBuffer	String/Byte配列のデータを使用するために必要なバッファを作成するクラスです
TuxFieldBuffer	TuxedoのFML/FML32バッファを使用するためのデータ・クラスです

以下の節で、各クラスとクラスで提供する関数について説明します。

2.2. TuxServiceクラス

Tuxedoのサービスを実行するための関数を提供するクラスです。すべての通信がTuxServiceを使用して行われます。

以下は、TuxServiceクラスで提供する関数についての説明です。

関数	説明
createCarrayBuffer	Tuxedoとの通信に使用できるByte配列型バッファを作成します
createFieldBuffer	Tuxedoとの通信に使用できるFMLバッファを作成します
createStringBuffer	Tuxedoとの通信に使用できるString型バッファを作成します
setAttribute	サービスに適用するユーザー・フラグを設定します
setDefaultCharset	String型データを使用する場合、Byte配列⇄文字列にエンコードおよびデコードする文字セットを指定します

関数	説明
setServiceName	TuxServiceインスタンスの基本サービス名を変更します
tpacall	非同期型方式でTuxedoのサービスを呼び出します
tpcall	同期型方式でTuxedoのサービスを呼び出します
tpconnect	Tuxedoの会話型通信を設定します
tpdiscon	Tuxedoの会話型通信を切断します
tpgetrply	tpacallの応答を取得します
tprecv	設定された会話型サービスを使用してメッセージを受信します
tpsend	設定された会話型サービスを使用してメッセージを送信します
TuxService	TuxServiceを作成します

2.2.1. createCarryBuffer

Tuxedoとの通信に使用できるByte配列型バッファを作成する関数です。

- プロトタイプ

```
public TuxBuffer createCarryBuffer([int size])
```

- パラメータ

パラメータ	説明
size	createCarryBufferで作成するCarryBufferのサイズです

- 戻り値

TuxBufferを返します。

2.2.2. createFieldBuffer

Tuxedoとの通信に使用できるFMLバッファを作成する関数です。

- プロトタイプ

```
public TuxBuffer createFieldBuffer([boolean isFML32, [int size,]]
                                   TuxFieldTable table)
```

- パラメータ

パラメータ	説明
isFML32	以下の値のいずれかを指定します <ul style="list-style-type: none"> – True: FML32を使用する場合 – False: FML16を使用する場合
size	createFieldBufferで作成するFieldBufferのサイズです
table	TuxFieldGen()を使用して作成された表クラスです

- 戻り値

TuxBufferを返します。

- 例

```
java -cp ../webtJtc.jar tmax.webt.util.TuxFieldGen -32 com.tys.sde.wtc anta.fld

package com.tys.sde.wtc;

import tmax.webt.util.TuxFieldTable;

public class anta.fld implements TuxFieldTable {
    public static final int    ftp_hostdir = 167804166;
    public static final int    ftp_filename = 167804165;
    public static final int    ftp_dir = 167804164;
    public static final int    ftp_pswd = 167804163;
    public static final int    ftp_id = 167804162;
    public static final int    ftp_svr = 167804161;
    public static final int    split_filename = 167807162;
    public static final int    split_data = 167807161;
    public static final int    carray01 = 201349593;
    public static final int    string99 = 167792259;
    public static final int    string98 = 167792258;
    public static final int    string97 = 167792257;
    public static final int    string96 = 167792256;
    public static final int    string93 = 167792253;
}
javac -classpath ../webtJtc.jar;sdeLib.jar;%JEUS_HOME%/lib/
system/jeus.jar WTCService.java 2> err
```

2.2.3. createStringBuffer

Tuxedoとの通信に使用できるString型バッファを作成する関数です。

- プロトタイプ

```
public TuxBuffer createStringBuffer([int size])
```

- パラメータ

パラメータ	説明
size	createStringBufferで作成するStringBufferのサイズです

- 戻り値

TuxBufferを返します。

2.2.4. setAttribute

サービスに適用するユーザー・フラグを設定する関数です。

- プロトタイプ

```
public void setAttribute(int attr, boolean value)
```

- パラメータ

パラメータ	説明
attr	対象フラグの値です
value	WebtAttributeの設定および解除を指定します <ul style="list-style-type: none">– True: WebtAttributeを設定します– False: WebtAttributeを解除します

2.2.5. setDefaultCharset

String型データを使用する場合に、Byte配列⇄文字列にエンコードおよびデコードする文字セットを指定する関数です。

ISO-8859-1、UFT-8、EUC-KRなどの文字セットを指定でき、指定していない場合はJavaのデフォルト値であるISO-8859-1に指定されます。バッファーを作成する前に必ず値を設定します。

- プロトタイプ

```
public void setDefaultCharset(String charset)
```

- パラメータ

パラメータ	説明
charset	エンコードする文字セットの値です(デフォルト値 : ISO-5589-1)

- 例

```
TuxService service = new TuxService("SAMPLE", "TUXDOM");
service.setDefaultCharset("ISO-8859-1");
TuxBufferImpl buf = service.createStringBuffer(); // ISO-8859-1

service.setDefaultCharset("ISO-8859-2");
TuxBufferImpl buf2 = service.createStringBuffer(); // ISO-8859-2
```

2.2.6. setServiceName

TuxServiceインスタンスの基本サービス名を変更する関数です。

- プロトタイプ

```
public void setServiceName(String serviceName)
```

- パラメータ

パラメータ	説明
serviceName	呼び出すTuxedoのサービス名です

- 例

```
TuxService service = new TuxService("SAMPLE", "TUXDOM");
Service.tpcall(buf); // calls "SAMPLE" service

service.setServiceName ("SAMPLE2");
Service.tpcall(buf); // calls "SAMPLE2" service
```

2.2.7. tpacall

非同期型方式でTuxedoのサービス呼び出す関数です。クライアントは、サービス呼び出した後に該当サービスの呼び出しに関するハンドラ値を取得することになり、このハンドラを使用していつでも結果値を取得できます。

- プロトタイプ

```
public TuxCallDescriptor tpacall([String svcname,] TuxBuffer input
                                [, WebtAttribute attr])
```

- パラメータ

パラメータ	説明
svcname	Tuxedoのサービス名です
input	呼び出すTuxBuffer型のバッファです
attr	サービスの属性値です

- 戻り値

TuxCallDescriptorを返します。

2.2.8. tpcall

同期型方式でTuxedoのサービスを呼び出す関数です。クライアントは指定された時間まで、ブロックされた状態で結果値を待機します。TuxServiceインスタンスを作成する際に設定した基本サービス以外のサービスを呼び出す場合、サービス名を指定することができます。

- プロトタイプ

```
public TuxBuffer tpcall(String svcname, TuxBuffer input[, WebtAttribute attr],
                        int timeout)
```

- パラメータ

パラメータ	説明
svcname	Tuxedoのサービス名です
input	要求するTuxBuffer型のバッファです
timeout	サービスを要求する際の最大実行時間です

- 戻り値

TuxBufferを返します。

- 例

```
try {
    TuxBufferImpl rcvbuf = service.tpcall(sndbuf);
    out.println(rcvbuf.getString());
}
```

```

    }
    catch (WebtException e) {
        e.printStackTrace();
        out.println("error Occurred");
        return;
    }
}

```

2.2.9. tpconnect

Tuxedoの会話型通信を設定する関数です。

- プロトタイプ

```

public TuxCallDescriptor tpconnect([TuxBuffer sndbuf,] [WebtAttribute attr,]
                                   boolean recvNext)

```

- パラメータ

パラメータ	説明
sndbuf	要求するTuxBuffer型のバッファです
attr	サービスの属性値です
recvNext	以下の値のいずれかを指定します <ul style="list-style-type: none"> – True: 次の応答を受信します – False: 応答を1回のみ受信します

- 戻り値

TuxCallDescriptorを返します。

2.2.10. tpdicon

Tuxedoの会話型通信を切断する関数です。

- プロトタイプ

```

public void tpdicon(TuxCallDescriptor cd)

```

- パラメータ

パラメータ	説明
cd	tpconnect()の実行に対して返されたハンドラ値です

2.2.11. tpgetrply

tpacallの応答を取得するための関数です。

- プロトタイプ

```
public TuxBuffer tpgetrply(TuxCallDescriptor cd[, WebtAttribute attr]
                           [, int timeout])
```

- パラメータ

パラメータ	説明
cd	tpacall()の実行に対して返されたハンドラ値です
attr	サービスの属性値です
timeout	サービスを要求する際の最大実行時間です

- 戻り値

TuxBufferを返します。

- 例

```
TuxService service = new TuxService("READDATA", "TUXDOM");
TuxBufferImpl sndbuf = service.createCarrayBuffer();
sndbuf.setBytes(toconv);

try {
    TuxCallDescriptor cd = service.tpacall(sndbuf);
    TuxBufferImpl rcvbuf = service.tpgetrply(cd);
    out.println(rcvbuf.getString());
}
catch (WebtException e) {
    e.printStackTrace();
    out.println("error Occurred");
    return;
}
```

2.2.12. tprecv

設定された会話型サービスを使用してメッセージを受信する関数です。

- プロトタイプ

```
public TuxBuffer tprecv(TuxCallDescriptor cd[,WebtAttribute attr])
```

- パラメータ

パラメータ	説明
cd	tpacall()関数の実行に対して返されたハンドラ値です
attr	サービスの属性値です

- 戻り値

TuxBufferを返します。

- 例

```
TuxService service = new TuxService("TMXDOM", "TOUPPER_CONVN");
TuxBuffer sndbuf = service.createStringBuffer();
sndbuf.setString("conversation service");
TuxCallDescriptor cd = service.tpconnect(false);

if(cd == null) {
    out.println("tpconnect failed!");
}
else {
    service.tpsend(cd, sndbuf, true);
    WebtAttribute attr = new WebtAttribute();
    attr.setTPNOTIME(true);

    if(service.isRecvNext()) {
        TuxBuffer rcvbuf = service.tprecv(cd,attr);
    }
}
service.tpdicon(cd);
```

2.2.13. tpsend

設定された会話型サービスを使用してメッセージを送信する関数です。

- プロトタイプ

```
public void tpsend(TuxCallDescriptor cd,TuxBuffer tx[, WebtAttribute attr],
                  boolean recvNext)
```

- パラメータ

パラメータ	説明
cd	tpacall()関数の実行に対して返されたハンドラ値です
tx	要求するTuxBuffer型のバッファです
attr	サービスの属性値です
recvNext	以下の値のいずれかを指定します <ul style="list-style-type: none"> – True: 次の応答を受信します – False: 応答を1回のみ受信します

2.2.14. TuxService

TuxServiceを作成する関数です。Tuxedoドメインが1つのみ設定されている場合、サービス名のみをパラメータに設定することが可能です。

- プロトタイプ

```
public TuxService([String domainName,] String serviceName)
```

- パラメータ

パラメータ	説明
domainName	Tuxedoのドメイン名です
serviceName	Tuxedoのサービス名です

- 戻り値

TuxServiceを返します。

2.3. TuxAsyncServiceクラス

非同期リスナー構造をサポートするためのTuxServiceの継承クラスです。TuxAsyncServiceで提供する関数は以下のとおりです。

関数	説明
getXAResource	XAリソースを作成します

関数	説明
tpacall	非同期型方式でTuxedoのサービスを呼び出します
TuxAsyncService	TuxServiceを作成します

以下は、TuxAsyncServiceを使用した非同期リスナー構造の使用例です。

```
TmaxXid xid = new TmaxXid(1,4,0);
TuxAsyncService service = new TuxAsyncService("TUXGW1","TUX_INSERT");
try {
    service.getXAResource().start(xid, 0);
}
catch (XAException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
TuxBuffer input = service.createStringBuffer();
input.setString("abc");
TuxAsyncMsgListener listener = new TestMsgListener(service, xid);
service.tpacall(input, null, listener );
```

以下は、TuxAsyncServiceを使用してTuxAsyncMsgListenerを実装する例です。TuxAsyncMsgListenerインターフェースはvoid handleEvent(TuxBuffer rcvBuffer)とvoid handleError(Exception e)メソッドを持っています。このメソッドの受信動作を実装します。

```
public class TestMsgListener implements TuxAsyncMsgListener {
    TuxAsyncService service;
    Xid xid;

    public TestMsgListener(TuxAsyncService service, Xid xid) {
        this.service = service;
        this.xid = xid;
    }

    public void handleError(Exception e) {
        System.out.println(e);
    }

    public void handleEvent(TuxBuffer rcvBuffer) {
        System.out.println(Thread.currentThread().getName()
            +" ] rcv " + rcvBuffer);

        try {
            service.getXAResource().prepare(xid,
                new TestTuxPrepareMsgListener(service,xid));
        }
        catch (XAException e) {
            // TODO Auto-generated catch block

```

```

        e.printStackTrace();
    }
}

```

2.3.1. getXAResource

XAリソースを作成する関数です

- プロトタイプ

```
public TuxAsyncXAResource getXAResource()
```

- 戻り値

TuxAsyncXAResourceを返します。

非同期リスナーのためにTuxAsyncXAResourceに追加実装されたメソッドは以下のとおりです。

```

public int prepare(Xid xid, TuxAsyncMsgListener listener)
public void commit(Xid xid, TuxAsyncMsgListener listener)
public void rollback(Xid xid, TuxAsyncMsgListener listener)

```

2.3.2. tpacall

非同期型方式でTuxedoのサービスを呼び出します。TuxAsyncMsgListenerインターフェースはvoid handleEvent(TuxBuffer rcvBuffer)とvoid handleError(Exception e)メソッドを持っています。このメソッドの受信動作を実装します。「[2.2.7. tpacall](#)」の応答を処理するためのリスナーをパラメータに追加します。

- プロトタイプ

```

public TuxCallDescriptor tpacall(TuxBuffer input, WebAttribute attr,
                                TuxAsyncMsgListener listener)

```

- パラメータ

パラメータ	説明
input	要求するTuxBuffer型のバッファです
attr	サービスの属性値です
listener	受信メッセージの動作が実装されたオブジェクトです

- 戻り値

TuxCallDescriptorを返します。

2.3.3. TuxAsyncService

TuxServiceを作成する関数です。Tuxedoドメインが1つのみ設定されている場合、サービス名のみをパラメータに設定することが可能です。

- プロトタイプ

```
public TuxAsyncService([String domainName,] String serviceName)
```

- パラメータ

パラメータ	説明
domainName	Tuxedoのドメイン名です
serviceName	Tuxedoのサービス名です

- 戻り値

TuxAsyncServiceを返します。

2.4. TuxRemoteServiceManagerクラス

JTCクラスターを使用する際は、効率的に呼び出すためにこのクラスのAPIを使用します。このクラスはサービス呼び出す際に適切なリモート・ドメインを使用して呼び出すように動作します。

TuxServiceを使用して呼び出す場合は、毎回異なるドメインで呼び出される可能性があります。TuxedoRemoteServiceManagerを使用してサービス呼び出す場合は、1つのサービスが特定のリモート・ドメインを使用して呼び出されると、それ以降はそのサービスは常に同じドメインを使用して呼び出されます。

以下は、TuxRemoteServiceManagerで提供する関数についての説明です。

関数	説明
TuxedoRemoteServiceManager	クラスの構築子です
createStringBuffer	Tuxedoとの通信に使用できるString型バッファを作成します
createCarrayBuffer	Tuxedoとの通信に使用できるByte配列型バッファを作成します
createFiledBuffer	Tuxedoとの通信に使用できるFiled型バッファを作成します
tpcall	Tuxedoサービスを呼び出します

2.4.1. TuxedoRemoteServiceManager

TuxedoRemoteServiceManagerクラスを作成します。

- プロトタイプ

```
Public TuxedoRemoteServiceManager()
```

2.4.2. createStringBuffer

Tuxedoとの通信に使用できるString型バッファを作成する関数で、動作はTuxServiceクラスのcreateStringBuffer関数と同一です。

- プロトタイプ

```
Public TuxBuffer createStringBuffer([int size])
```

- 戻り値

TuxBufferを返します。

2.4.3. createCarrayBuffer

Tuxedoとの通信に使用できるCarray型バッファを作成する関数で、動作はTuxServiceクラスのcreateCarrayBuffer関数と同一です。

- プロトタイプ

```
Public TuxBuffer createCarrayBuffer([int size])
```

- 戻り値

TuxBufferを返します。

2.4.4. createFiledBuffer

Tuxedoとの通信に使用できるFiled型バッファを作成する関数で、動作はTuxServiceクラスのcreateFiledBuffer(TuxFieldTable)関数と同一です。

- プロトタイプ

```
Public TuxBuffer createFiledBuffer(TuxFieldTable table)
```

- パラメータ

パラメータ	説明
table	TuxFieldGenを使用して作成された表クラスです

- 戻り値

TuxBufferを返します。

2.4.5. tpcall

Tuxedoサービスを呼び出す関数です。

- プロトタイプ

```
Public TuxBuffer tpcall(String svcname, TuxBuffer input)
```

- パラメータ

パラメータ	説明
svcname	要求するサービス名です
input	要求するTuxBuffer型のバッファです

- 戻り値

TuxBufferを返します。

2.5. TuxBufferクラス

Tuxedoとデータを送受信するために使用されるTuxedoデータ・バッファの上位クラスです。特別な関数は提供せず、下位データ・バッファ・クラスのインターフェースを提供します。Tuxedoデータ・バッファはTuxServiceインスタンスのメソッドを使用するか、WebtBufferクラスのStaticメソッドを使用して作成します。

2.6. TuxStringBuffer/TuxCarrayBufferクラス

それぞれString型データとByte配列データを使用するためのバッファです。

以下は、TuxStringBuffer/TuxCarrayBufferが提供する関数についての説明です。

関数	説明
getBytes	格納されているString値をByte配列の値に変換して読み込みます
getString	格納されているString値を読み込みます
setBytes	Byte配列の値を格納します
setString	String値を格納します

2.6.1. getBytes

格納されているString値をByte配列の値に変換して読み込む関数です。

- プロトタイプ

```
public byte[] getBytes()
```

- 戻り値

格納されているString値をByte配列に変換して返します。

2.6.2. getString

格納されているString値を読み込む関数です。

- プロトタイプ

```
public String getString([String charset])
```

- パラメータ

パラメータ	説明
charset	エンコードのための文字セットを指定でき、不要な場合はNULLを設定します

- 戻り値

格納されているString値を返します。

2.6.3. setBytes

Byte配列の値を格納する関数です。

- プロトタイプ

```
public void setBytes(byte[] val)
```

- パラメータ

パラメータ	説明
val	要求するByte型のデータです

2.6.4. setString

String値を格納する関数です。

- プロトタイプ

```
public void setString(String value[, String charset])
```

- パラメータ

パラメータ	説明
value	要求するString型のデータです
charset	デコードのための文字セットを指定することもでき、不要な場合はNULLを設定します

2.7. TuxFieldBufferクラス

TuxedoのFML/FML32バッファを使用するためのデータ・クラスです。TuxFieldBufferはバッファ内部に固有のキー値とキー名を持つ複数のフィールドを所有し、各フィールドは再度複数の値を持つことができます。ユーザーは、値を格納するために、実際に値を入れられるフィールドを先に作成します。

文字列またはByte配列を格納する型と数字型を格納する型に区分され、フィールドの型と他の型のデータ格納を行った場合、WebtBufferExceptionが発生します。

以下は、TuxFieldBufferが提供する関数についての説明です。

関数	説明
createField	TuxFieldBuffer内に新しいフィールドを作成します
getField	キー値に該当するフィールドを返します
getFields	フィールド・バッファ内にあるすべてのフィールドをベクトル(Vector)に返します

2.7.1. createField

TuxFieldBuffer内に新しいフィールドを作成する関数です。

- プロトタイプ

```
public WebtField createField(int fldkey|string fldkey)
```

- パラメータ

パラメータ	説明
int fldkey	作成するint型のフィールド・キーの値です

パラメータ	説明
string fldkey	作成するString型のフィールド・キーの値です

- 戻り値

キー値に該当するフィールドが既に作成されている場合、既に作成されたフィールドを返します。

- 例

```
Table table = new Table()
TuxBufferImpl buf = WebtBuffer.createTuxFieldBuffer(true, table);

// フィールド・キーの値がINT1のWebtFieldオブジェクトを作成します。
WebtField field1 = buf.createField(table.INT1);

// フィールド・キーの名前が"INT2"のWebtFieldオブジェクトを作成します。
WebtField field2 = buf.createField("INT2");
```

2.7.2. getField

キー値に該当するフィールドを返す関数です。

- プロトタイプ

```
public WebtField getField(int fldkey|string fldkey)
```

- パラメータ

パラメータ	説明
int fldkey	返されたTuxbufferから読み込むint型のフィールド・キーの値です
string fldkey	返されたTuxbufferから読み込むString型のフィールド・キーの値です

- 戻り値

キー値に該当するフィールドを返します。フィールドが存在しない場合、NULLを返します。

2.7.3. getFields

フィールド・バッファ内にあるすべてのフィールドをベクトル(Vector)に返す関数です。

- プロトタイプ

```
public Vector getFields()
```


- 戻り値

フィールド・バッファ内にあるすべてのフィールドをベクトル(Vector)に返します。

- 例

```
Table table = new Table();
TuxBufferImpl buf = WebtBuffer.createTuxFieldBuffer(true, table);

WebtField field = buf.createField(table.INPUT);
field.add("VALUE1");
field.add("VALUE2");
field.add("VALUE3");

Vector fields = buf.getFields();
String[] fieldValues = (String[])fields.toArray(new String[0]);

for (int i=0; i++; i<fieldValues.length) {
    System.out.println("Value[" + i + "] : " + fieldValues[i]);
}
```


第3章 JTCモニタリング

本章では、JEUSコンソール・ツールとWebAdminを利用したJTCモニタリングに必要な設定とモニタリング方法について説明します。

3.1. 概要

JTCの起動中、JTCのローカル・ドメインおよびリモート・ドメインの現在の環境設定の情報を照会したり、ドメインを通じた呼び出しおよび応答回数を確認したりするなどの管理を行うために、JTCモジュールはJEUSのコンソール・ツールとWebAdminによりJTCの情報を提供します。

参考

JEUS WebAdminおよびコンソール・ツールを利用したJTC管理機能は、JEUS v6.0 Fix#7以上からサポートします。

3.2. 環境設定

JTC管理機能を利用するには、別途配布されるモジュールを適用することを含めて追加の設定を行う必要があります。

JEUSコンソール・ツールを利用してJTC Adminを利用するには、JEUSインストール・ディレクトリー配下のlib/systemディレクトリーに別途配布されたjext_webtadmin.jarファイルを置きます。

その他に必要な作業は、以下のとおりです。

- モニタリング関連設定（ディレクトリー設定およびmbean設定）
- webadmin.applicationの設定
- node.xmlの設定
- catalog_ko.xmlの設定
- jeus.cssの設定

参考

必要な作業別の設定方法については、『Tmax Webユーザガイド』の「5.2. 環境設定」を参照してください。

3.3. JTCモニタリング

本節では、JEUSのコンソール・ツールとWebAdminを利用したJTCのモニタリング方法について説明します。

3.3.1. JEUSコンソール・ツールを利用したモニタリング

JEUSはコンテナの状態を確認するためのjeusadminコマンドを提供します。このコマンドは新しく追加されたコマンドであり、すべての情報を最も詳しく出力します。

管理ツールを使用するには、コンソール画面で以下のように実行します。このコマンドを実行すると、jeusadminのコマンド・プロンプト画面に移動します。

```
jeusadmin <container name> -U <user nam> -P <user passwd>
```

JTCの状態情報を確認するためのコマンドはjtcadminです。jtcadminの使用形式は以下のとおりです。

```
jtcadmin [-d domainName] [-l]
```

各オプションの説明と実行結果は以下のとおりです。jtcadminで出力される環境設定の項目とその値は、ユーザーがwebt.propertiesファイルで設定した項目および値と一致します。

- オプションなしで実行する場合

オプションなしで実行すると、設定されているすべてのJTCの情報が出力されます。

- [-d domainName]

特定のリモート・ドメインの情報を出力します。

```
container name : Joonsoo-PC_container2
domain info --

domainName :          TMXDOM
interval :           30
timeout :             -1
tx timeout :          -1
read timeout :        -1
thread min :          1
thread max :          1
ip :                  192.168.33.84
port :                9111
notx :                 false
backup :               null
service list :
request :              0
reply :                0
```

```
prepare :          0
commit :           0
rollback :         0
```

- [-l]

ローカル・ドメインの情報、ローカル・サービスの呼び出し回数、ロギング情報を出力します。

```
container name : Joonsoo-PC_container2
Local Info --
log directory :      null
log file :          null
prepare count :      0
commit count :       0
rollback count :     1
remote domain list : TMXDOM
ejb call list :
-----
service name      call count
-----
GSVC01            1

cluster domain list : TDOM1:TMXDOM, TMXDOM2
```

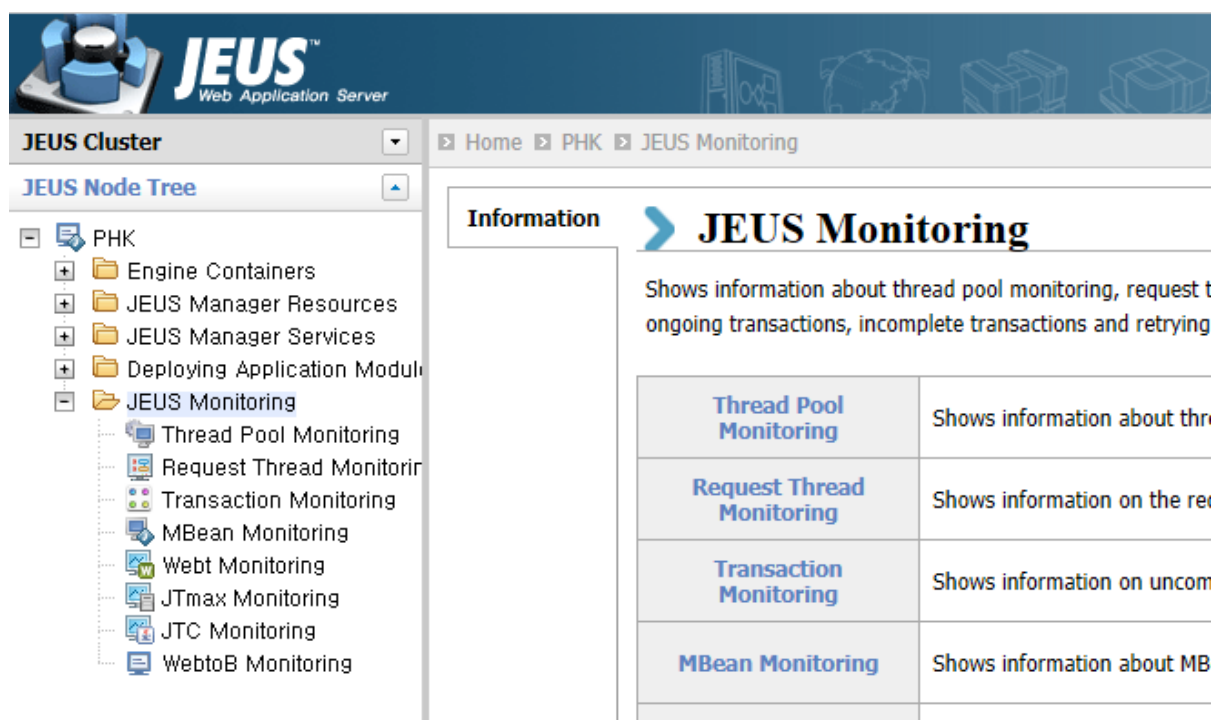
3.3.2. JEUS WebAdminを利用したモニタリング

コンテナの状態を確認するもう一つの方法は、Webブラウザで利用可能なJEUS WebAdminツールを利用する方法です。JEUSを起動した後、Webブラウザに次のアドレスを入力すると、webadmin画面に移動します。

```
http://<ip address>:9744/webadmin
```

ユーザー名とパスワードを入力してログインすると、以下の画面が表示されます。画面左の**JEUS Node Tree**の**[JEUS Monitoring]**項目の下にWebT、JTmax、JTCモニタリング・メニューがあります。

[図 3.1] webtadmin画面



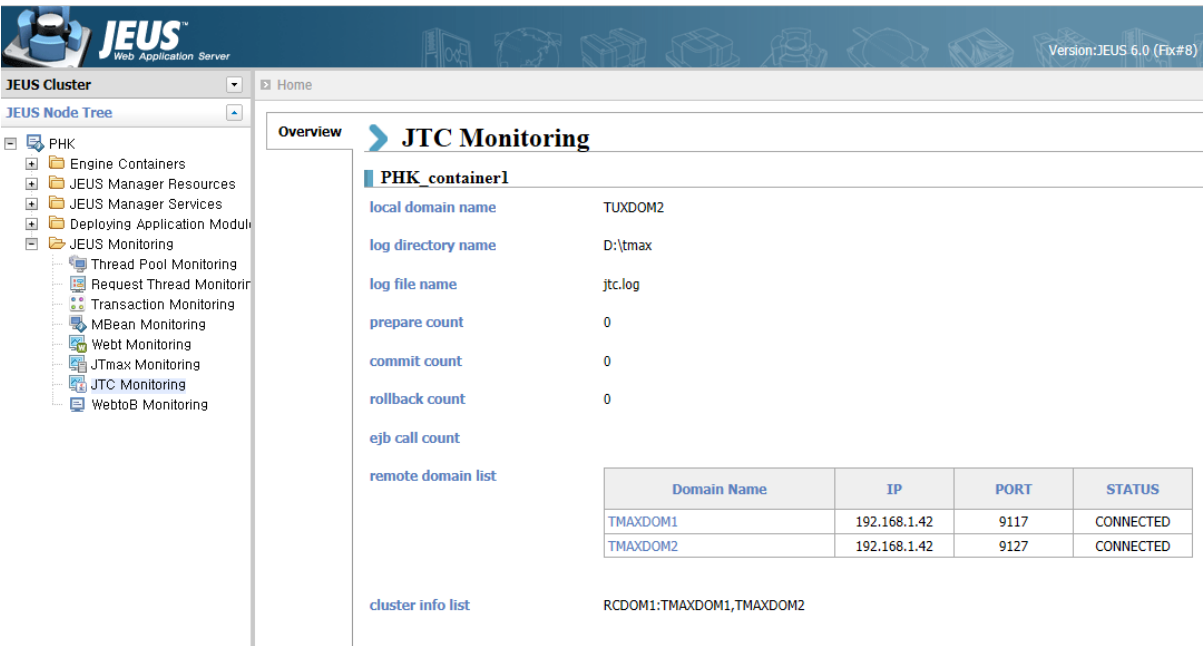
参考

1. JEUS WebAdminについての詳細は『JEUS WebAdminガイド』を参照してください。
2. WebT、JTmaxのモニタリングについての詳細は、『Tmax WebTユーザガイド』を参照してください。

JEUS WebAdmin画面の**JEUS Node Tree**で**[JEUS Monitoring] > [JTC Monitoring]**を選択すると、以下のように**JTC Monitoring**画面に移動します。

JTC Monitoring画面では、モニタリング情報とロギング情報、およびコネクション・グループ情報を表示します。出力される環境設定項目とその値はユーザーがwebt.propertiesファイルで設定した項目および値と一致します。

[図 3.2] JTC Monitoring画面



画面に表示された各項目についての説明は、以下のとおりです。

項目	説明
local domain name	設定したローカル・ドメインの名前です
log directory name	ログファイルが保存されるディレクトリーの名前です
log file name	ログファイルの名前です
prepare count	xa_prepareの処理件数です
commit count	xa_commitの処理件数です
rollback count	xa_rollbackの処理件数です
ejb call count	設定したEJBの呼び出し回数です
remote domain list	リモート・ドメインの一覧です。各リモート・ドメインの情報が表示されます。表示される項目についてはこの表下の説明を参照してください
cluster info list	クラスターに設定したドメインの一覧です

以下は、「remote domain list」項目で表示されるリモート・ドメイン情報についての説明です。

項目	説明
Domain Name	リモート・ドメインの名前です
IP	当該リモート・ドメインのIPアドレスです
PORT	当該リモート・ドメインのポート番号です
STATUS	当該リモート・ドメインの状態が表示されます

項目	説明
	<ul style="list-style-type: none"> – CLOSED : 接続終了 – CONNECTING : 接続中 – ESTABLISHING : 接続後認証中 – CONNECTED : 認証後接続完了 – SHUTDOWN : 終了

JTC Monitoring画面の「remote domain list」項目でドメイン名をクリックすると、当該リモート・ドメインの設定環境や運用情報などの詳細情報を確認することができます。

[図 3.3] JTC Monitoring画面 - リモート・ドメインの詳細情報

The screenshot shows the JEUS Web Application Server interface. On the left, the 'JEUS Node Tree' is expanded to 'JTC Monitoring'. The main area is titled 'JTC Remote Domain Monitoring' and shows details for a remote domain named 'TMAXDOM1'. The parameters listed are:

Parameter	Value
remote domain name	TMAXDOM1
ip	192.168.1.42
port	9117
status	CONNECTED
interval	3
backup domain	
not transaction	false
buffer size	8192
timeout	10
txtimeout	60
readtimeout	5
min thread	1
max thread	1
callcount	0
replycount	0
preparecount	0
commitcount	0
rollbackcount	0
service list	

各項目は設定情報を出力し、「callcount」、「replycount」、「preparecount」、「commitcount」、「rollbackcount」は各処理件数の合計です。

索引

C

- cluster info list, 47
- commit count, 47
- createCarrayBuffer, 24, 36
- createField, 39
- createFieldBuffer, 24
- createFiledBuffer, 36
- createStringBuffer, 25, 36

E

- ejb call count, 47

G

- getBytes, 38
- getField, 40
- getFields, 40
- getString, 38
- getXAResource, 34

J

- jtcadmin
 - [-d], 44
 - [-l], 45

L

- local domain name, 47
- log directory name, 47
- log file name, 47

P

- prepare count, 47

R

- remote domain list, 47
 - Domain Name, 47

- IP, 47
- PORT, 47
- STATUS, 47
- rollback count, 47

S

- setAttribute, 26
- setBytes, 38
- setDefaultCharset, 26
- setServiceName, 27
- setString, 39

T

- tpacall, 27, 34
- tpcall, 28, 37
- tpconnect, 29
- tpdiscon, 29
- tpgetrply, 30
- tprecv, 31
- tpsend, 31
- TuxAsyncService, 35
- TuxedoRemoteServiceManager, 35
- Tuxedoドメインの環境設定
 - CONNECTION_POLICY, 11
 - DMTLOGDEV, 11
- TuxService, 32

W

- webt.propertiesファイル
 - log.bufsize, 2
 - log.dir, 2
 - log.file, 2
 - log.file.date.format, 3
 - log.level, 2
 - log.valid.days, 3
- tux.<BackUp domain Name>.addr[.x], 6
- tux.<BackUp domain Name>.backup, 7
- tux.<BackUp domain Name>.notx, 7
- tux.<BackUp domain Name>.port[.x], 6
- tux.<clusterName>.remote.name.list, 3
- tux.<clusterName>.svc, 6
- tux.<domainName>.addr[.0], 5

tux.<domainName>.buffer.size, 4
tux.<domainName>.default.readtimeout, 5
tux.<domainName>.default.timeout, 5
tux.<domainName>.default.txtimeout, 5
tux.<domainName>.interval[.0], 6
tux.<domainName>.interval[.x], 6
tux.<domainName>.port[.0], 5
tux.<domainName>.svc, 6
tux.<domainName>.thread.max, 5
tux.<domainName>.thread.min, 5
tux.buffer.size, 4
tux.cluster.name.list, 3
tux.default.table, 4
tux.default.table.file, 4
tux.default.table.file.16, 4
tux.default.timeout, 4
tux.default.txtimeout, 4
tux.local.name, 3
tux.mbean, 7
tux.remote.name.list, 3