

# Tmax WebTAsync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 WebTAsync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 紹介 .....	1
1.1. 概要 .....	1
1.2. インバウンド .....	1
1.3. アウトバウンド .....	2
第2章 例 .....	3
2.1. インバウンドの例 .....	3
2.1.1. 環境設定 .....	3
2.1.2. サーバー開始 .....	5
2.1.3. サービス要求 .....	5
2.2. アウトバウンドの例 .....	8
2.2.1. 環境設定 .....	8
2.2.2. クライアント・プログラム .....	9
2.2.3. 主要機能 .....	16
索引 .....	19



## 図目次

[図 1.1] インバウンドの構造 .....	1
[図 1.2] アウトバウンドの構造 .....	2





# このガイドについて

## 対象読者

本書は、Tmax<sup>®</sup>(以下、Tmax)を使用してプログラムを開発するユーザーを対象としています。同書は、TmaxのAsync JavaゲートウェイとIN/OUT通信を非同期(asynchronous)で行うためのJavaライブラリーであるWebTAsyncについて説明しています。なお、WebTAsyncの概念、インバウンド(Inbound)、アウトバウンド(OutBound)についての説明と例について説明します。

## 前提知識

本書は、Tmaxシステムの概要とTmaxシステムが提供する各種機能や特性などを習得するための手引書です。

本書を理解するためには、以下の事項についての知識が必要です。

- ミドルウェアおよびUNIXシステムについての知識
- Tmaxの基本概念
- Java、Cプログラミングについての知識

## 制限事項

本書を読む前にTmaxの基本概念を熟知している必要があります。実務上の具体的な使用方法や管理・運用についての内容は、各製品ガイドを参照してください。

---

### 参考

Tmaxシステムの開発についての基本的な内容は、『Tmax 運用ガイド』および『Tmax アプリケーション開発ガイド』を、Tmaxが提供するコマンドとC APIについては、『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、計2章と付録で構成されています。

各章の主な内容は以下のとおりです。

- 第1章: 紹介

WebtAsyncについて説明し、WebtAsyncライブラリーを利用したインバウンドとアウトバウンドの構造について説明します。

- 第2章: 例

WebtAsyncのインバウンドとアウトバウンドの例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
<Ctrl>+C	CtrlとC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
<b>参考</b>	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、コマンド実行後の画面に出力された結果物、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

# 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](mailto: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](mailto:info@tmaxsoft.com)  
Web (English): [http://www.tmaxsoft.com/us\\_en/home](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](mailto: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](mailto:info@tmaxsoft.com.cn)  
Web (Chinese): [http://www.tmaxsoft.com/cn\\_en/home\\_cn\\_en](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](mailto:info.bra@tmaxsoft.com)  
Web (Portuguese): [http://www.tmaxsoft.com/br\\_en/home\\_br\\_en](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](mailto:info.rus@tmaxsoft.com)  
Web (Russian): [http://www.tmaxsoft.com/ru\\_ru/home\\_ru\\_ru](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](mailto:info.sg@tmaxsoft.com)  
Web (English): [http://www.tmaxsoft.com/sg\\_en/home\\_sg\\_en](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](mailto:info.uk@tmaxsoft.com)  
Web (English): [http://www.tmaxsoft.com/gb\\_en/home\\_gb\\_en](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](mailto:info.canada@tmaxsoft.com)  
Web (English): [http://www.tmaxsoft.com/ca\\_en/home\\_ca\\_en](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](mailto:info.aus@tmaxsoft.com)  
Web (English): [http://www.tmaxsoft.com/au\\_en/home\\_au\\_en](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](mailto:info.india@tmaxsoft.com)  
Web (English): [http://www.tmaxsoft.com/in\\_en/home\\_in\\_en](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](mailto:cslee@tmaxsoft.com)  
Web (English): [http://www.tmaxsoft.com/tr\\_en/home\\_tr\\_en](http://www.tmaxsoft.com/tr_en/home_tr_en)



# 第1章 紹介

本章では、WebtAsyncについて説明し、WebtAsyncライブラリーを利用したインバウンドとアウトバウンドの構造について説明します。

## 1.1. 概要

WebTAsyncとは、TmaxのAsync JavaゲートウェイとIn/Out通信を非同期(Asynchronous)で行うためのJavaライブラリーです。ユーザー(開発者)は、ライブラリーを利用してTmaxにサービスを要求するか、Tmaxから要求されたサービス进行处理して結果をTmaxに再転送することができます。WebTAsyncを利用すると、既存のJEUS以外に他ベンダーのWebアプリケーション・サーバーとTmaxサーバー間の双方向通信が可能になります。

このライブラリーはTmax 4.0 SP3 Fix#8バージョン以降で動作します。

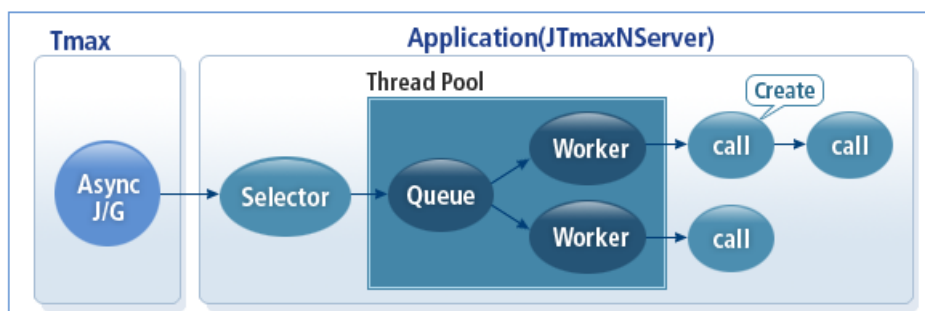
## 1.2. インバウンド

TmaxにてAsync Javaゲートウェイを通じてJTmaxNServerにサービスを要求することを**インバウンド**と定義します。

WebTAsyncライブラリーを利用して作成されたアプリケーション(JTmaxNServer)を起動すると、TmaxのAsync Javaゲートウェイからの要求を待ちます。サービス要求はワーカー・スレッド(Worker Thread)が処理しており、この処理は、ユーザーが事前登録した**コールバック・インターフェース**を呼び出します。既存のJTmaxはejbを呼び出しますが、WebtAsyncは**コールバック・インターフェース**を呼び出すのが差異点です。

下記は、インバウンドの構造です。

[図 1.1] インバウンドの構造



## 1.3. アウトバウンド

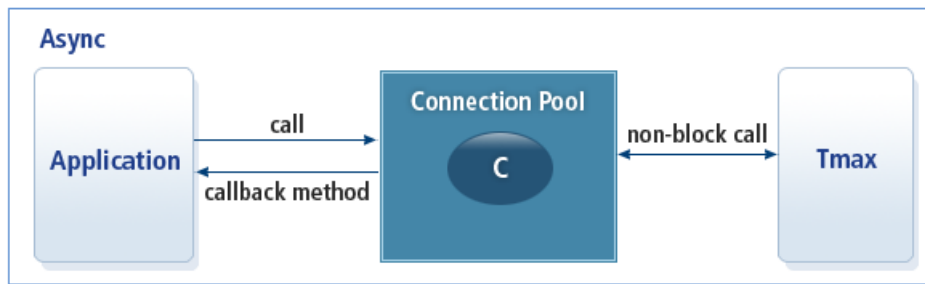
WebtAsyncライブラリーを利用してTmaxにサービスを要求することを**アウトバウンド**と定義します。

WebtAsyncライブラリーを利用して作成したアプリケーションで、TmaxのAsync Javaゲートウェイと複数のコネクトを確立することができます。既存のWebTは、アプリケーション内のコネクション・プールからコネクションを取得し、サービスを要求する形式です。一方、WebtAsyncは内部的に使用する回線を決めてサービス・コネクション・プールを要求します。これは、同一回線で別のトランザクションも要求できることを意味します。

ユーザーは、要求に対する応答を待たず次のジョブが実行でき、結果は新しいスレッドに登録されている**コールバック・メソッド**を呼び出してユーザーに通知します。すなわち、ユーザーが結果に対する**コールバック・インターフェース**を登録したら、新しいスレッドに**コールバック・メソッド**が呼び出される方式です。

下記は、アウトバウンドの構造です。

[図 1.2] アウトバウンドの構造



## 第2章 例

本章では、WebtAsyncのインバウンドとアウトバウンドの例について説明します。

### 2.1. インバウンドの例

#### 2.1.1. 環境設定

インバウンド・サービスを使用するには、TmaxとJTmaxNServerに対する環境設定が必要となります。

##### Tmax環境設定

GATEWAY設定でGWTYPEのみ「JEUS\_ASYNC」とし、以外の設定は既存のJavaゲートウェイと同様にします。

```
*SERVICE
ECHO_STRING SVRNAME= javagwa

*GATEWAY
javagwa GWTYPE=JEUS_ASYNC,
        NODENAME=tmax,
        PORTNO=8550,
        RGWADDR="192.168.11.240",
        RGWPORTNO=8282,
        CLOPT="-r"
```

インバウンドに必要な設定は、GWTYPE=JEUS\_ASYNC、RGWADDR="192.168.11.240"、RGWPORTNO=8282です。RGWADDR、RGWPORTNOは、JTmaxNServerに接続するIPとポート番号です。

Tmaxが起動する場合、JTmaxNServerとmaxchl \* cpcの個数の分だけコネクトが確立されます。

##### JTmaxNServerの環境設定

JTmaxNServerの環境設定は、JTmaxNServerがWebLogicサーバーで起動されることを基準にして説明します。サンプルJTmaxNServerを起動するため、WebLogicサーバーのJTmaxNServerアプリケーションをear形式で配布します。

サンプル・アプリケーションのTmaxInboundEApplication.ear内には、JTmaxServerを起動および終了するためのインプリメンテーションとメッセージを処理するためのJTmaxNJGMessageHandlerが実装されています。

- TmaxInboundEApplication.ear /META-INF/weblogic-application.xml設定

WebLogicサーバーを起動する場合、JTmaxNServerを起動して「/META-INF」内のweblogic-application.xmlに下記のとおり設定します。

TmaxServerLifeCycleクラスは、WebLogicが提供するApplicationLifecycleListenerを継承して実装します。

```
<?xml version="1.0" encoding="UTF-8"?>
<weblogic-application xmlns="http://www.bea.com/ns/weblogic/90">
<listener>
    <listener-class>com.tmax.inbound.TmaxServerLifeCycle
    </listener-class>
    <listener-uri>TmaxInboundEApplicationEJB.jar</listener-uri>
</listener>
</weblogic-application>
```

- webt.properties設定

JTmaxNServer内でWebT APIを使用する場合、ログを残すかFDLを使用するには、webt.propertiesに上記のように設定します。WebLogicサーバーを起動する場合は、「-Dwebt.properties=絶対パス」と設定します。

```
log.level=debug
log.dir=D:\\tmax
log.file=tmax_webt.log
fdl.file=D:\\tmax\\tmax.fdl
```

- com.tmax.inbound.properties設定

JTmaxNServerを起動するための基本設定とメッセージを処理するために必要なEJBの設定ファイルです。

下記の設定は、ECHOSTRINGというサービスが呼び出されたら、echo EJBのechoStringメソッドが呼び出す方式で動作します。この設定ファイルは、サンプルJTmaxNServerの設定ファイルとして必須ファイルではないため、別の方式でいくらかでも実装可能な部分です。

```
ejb.names=ejb1
ejb1.exportname=Echo
ejb1.methodname=echoString
ejb1.tmax.servicename=ECHO_STRING
tmax.inbound.servers=jtmax1
jtmax1.listen.port=8282
```

```
jtmax1.worker.thread.max = 20
jtmax1.worker.connection.max = 50
```

## 2.1.2. サーバー開始

Tmaxからのサービス要求を処理するためのJTmaxNServerをJTmaxNServerImplでサーバー・オブジェクトを作成します。

JTmaxNServerを作成する際にサービスを処理するためのコールバック・インターフェースであるJTmaxNJGMessageHandlerを実装したインプリメンテーションをパラメータとして設定する必要があります。リスニング・ポートを使ってソケットをオープンし、要求を待ちます。そのとき、アクセプター・スレッドを作成することになります。

```
JTmaxNServer server = new JTmaxNServerImpl("JTmaxNServer1", 8881, 100, 50,
                                           new WebLogicJTmaxNJGMessageHandler());
server.startServer();
```

下記は、JTmaxNServerImplの構築子APIについての説明です。

```
JTmaxServerImpl(String serverName, int port, int connection, int thread,
                JTmaxNJGMessageHandler handler)
```

項目	説明
serverName	JTmaxNServerの名前です
port	listening port numberです
connection	許可できるコネクション数です
thread	メッセージを受信する際、コールバック呼び出しするユーザー・スレッドの数です
handler	メッセージ処理のハンドラーです

## 2.1.3. サービス要求

### Tmaxクライアント・サービス要求

Tmaxクライアントでは、Tmaxに存在するサービスを呼び出すのと同様にサービスを要求します。サービス・ロジックを処理した後、serviceHandler.serviceFinished()関数を呼び出したらTmaxサーバーに結果を渡します。

```
...
if ((sndbuf = (char *)tpalloc("STRING", NULL, 0)) == NULL) {
    printf("sndbuf alloc failed !\n");
    tpend();
}
```

```

        exit(1);
    }
    cd = tpcall("ECHO_STRING", sndbuf, 0, 0);
    ...

```

## JTmaxNServerにサービス要求

Tmaxサーバーがクライアントからサービス要求を受けた後、Async Javaゲートウェイを通じてJTmaxNServerに当該サービスを要求します。JTmaxNServerが要求を受けたら、JTmaxNJGMessageHandlerのserviceExecuteが呼び出されます。開発者はserviceExecuteメソッドにてserviceNameに該当するサービス処理ロジックを実装します。

```

...
public void serviceExecute(int sessionId, Xid xid, String serviceName,
                           WebtBuffer receiveBuffer, JTmaxNServiceHandler
                           serviceHandler) {
    // サービス処理ロジック部分
    EchoHome home = (EchoHome) PortableRemoteObject.narrow(objref,
                                                             EchoHome.class);

    Echo echo = home.create();
    String value = echo.echoString(receiveBuffer.getString());
    WebtBuffer sendBuffer = new WebtStringBuffer();
    sendBuffer.setString(value);
    if (serviceHandler != null) {
        serviceHandler.serviceFinished
            (JTmaxNServiceHandler.TPSUCCESS, EJB_SVC_SUCCESS, sendBuffer);
    }
}
...

```

EJBCallWorker、XACommitWorker、XAPrepareWorker、XARollbackWorkerは、サービス进行处理するために作成したユーザー・スレッド・クラスです。

```

public class WebLogicJTmaxNJGMessageHandler implements
    JTmaxNJGMessageHandler {
    public WebLogicJTmaxNJGMessageHandler () {
    }
    private XAResource getXAResource() {
        return
TxHelper.getServerInterposedTransactionManager().getXAResource();
    }
    public void bulkRecover(Xid[] xids) throws NotSupportedException {
        if (xids == null) {
            return;
        }
        TmaxXid tmaxXid = null;
        for (int i = 0; i < xids.length; i++) {

```

```

        tmaxXid = (TmaxXid) xids[i];
        int decision = tmaxXid.getDecision();
        switch (decision) {
        case tmax.webt.io.Webt.TM_XA_COMMIT:
        case tmax.webt.io.Webt.TM_XA_COMMIT2:
            xaCommit(-1, xids[i], false, null);
            break;
        case tmax.webt.io.Webt.TM_XA_RBACK:
        case tmax.webt.io.Webt.TM_XA_RBACK2:
            xaRollback(-1, xids[i], null);
            break;
        default:
            break;
        }
    }
}

public void closeSession(int sessionId) {
    System.out.println("closed sessionId = " + sessionId);
}

public void closeTrunk(TmaxAliveInfo info) {
    for (int i = 0; i < info.getTrunkSize(); i++) {
        Hashtable<Integer, String> table = info.getTrunk(i);
        Iterator<Integer> it = table.keySet().iterator();
        while (it.hasNext()) {
            Integer key = (Integer) (it.next());
            String ip = (String) (table.get(key));
            System.out.println("client index = " + key + ", ip = " + ip);
        }
    }
}

public Xid[] recover() {
    Xid[] xids = null;
    try {
        xids = getXAResource().recover(XAResource.TMSTARTRSCAN);
    } catch (XAException e) {
        e.printStackTrace();
        return null;
    }
    return xids;
}

public void serviceExecute(int sessionId, String serviceName,
    WebtBuffer receiveBuffer) {
    new Thread(new EjbCallWorker(sessionId, serviceName,
        receiveBuffer)).start();
}

public void serviceExecute(int sessionId, Xid xid, String serviceName,
    WebtBuffer receiveBuffer, JTmaxNServiceHandler

```

```

        serviceHandler) {
            new Thread(new EjbCallWorker(sessionId, xid, serviceName,
                receiveBuffer, serviceHandler)).start();
        }
    public void xaCommit(int sessionId, Xid xid, boolean onePhrase,
        JTmaxNServiceHandler serviceHandler) {
        new Thread(new XACommitWorker(sessionId, xid, onePhrase,
            serviceHandler)).start();
    }
    public void xaPrepare(int sessionId, Xid xid,
        JTmaxNServiceHandler serviceHandler) {
        new Thread(new XAPrepareWorker(sessionId, xid,
            serviceHandler)).start();
    }
    public void xaRollback(int sessionId, Xid xid,
        JTmaxNServiceHandler serviceHandler) {
        new Thread(new XARollbackWorker(sessionId, xid,
            serviceHandler)).start();
    }
}

```

---

#### 参考

関連するAPIの詳細情報は、**WebtAsync 2.0.0.0 API**を参照してください。

---

## 2.2. アウトバウンドの例

### 2.2.1. 環境設定

アウトバウンド・サービスを使用するには、Tmaxに対する環境設定が必要となります。

#### Tmax環境設定

Tmax環境設定ファイルのGATEWAY設定にてGWTYPEのみ「JEUS\_ASYNC」とし、以外の設定は既存のJavaゲートウェイと同様にします。

```

*SERVICE
TOUPPER SVRNAME=svr2a

*GATEWAY
javagwa GWTYPE=JEUS_ASYNC,
NODENAME=tmax,
PORTNO=8550,

```



```
RGWADDR="192.168.11.240",  
RGWPORTNO=8282,  
CLOPT="-r"
```

アウトバウンドに必要な設定は、GWTYPE=JEUS\_ASYNC、PORTNO=8550です。PORTNOは、WebtAsyncアプリケーションから接続要求を受けるポート番号です。

## 2.2.2. クライアント・プログラム

### tpcallの例

TmaxのAsync Javaゲートウェイと接続するためのWebtEndpointオブジェクトを作成し、接続情報およびその他の設定を行います。WebtEndpointでAsync Javaゲートウェイに接続できるコネクション数は、GATEWAYセクションのMAXINRGW項目で設定することができます。MAXINRGWのデフォルト値は32なので、デフォルト値を使用する場合の接続可能な最大コネクション数は、「32+30=62」です。

```
public class TpcallSample {  
    public final String TMAX_HOST_ADDR = "192.168.1.48";  
    public final int TMAX_JGW_PORT = 8881;  
    public final String BACKUP_HOST_ADDR = "192.168.1.44"; public final  
    int BACKUP_JGW_PORT = 8881;  
    public final int MAX_THREAD = 20;  
    public final int MAX_CONNECTION = 100;  
    public String serviceName = "TOUPPER";  
    public TpcallSample() {  
    }  
    public void excute() {  
        WebtEndpoint webtEndpoint = new WebtEndpointImpl();  
        webtEndpoint.setMainServerInfo(TMAX_HOST_ADDR, TMAX_HOST_PORT);  
        webtEndpoint.setFailoverServerInfo(BACKUP_HOST_ADDR, BACKUP_HOST_PORT);  
  
        webtEndpoint.setTimeout(3 * 1000, 4 * 1000);  
        try {  
            webtEndpoint.startClientManager("WebT-001", MAX_THREAD,  
                                           MAX_CONNECTION);  
        }  
        catch (IOException e) {  
            e.printStackTrace();  
        }  
        webtEndpoint.startCheckFailBack(new  
            SampleFailBackMonitor(webtEndpoint), 5 * 1000, 1 * 1000);  
        WebtNService service = new WebtNServiceImpl(webtEndpoint);  
        WebtBuffer sndBuffer = new WebtStringBuffer();  
        WebtBuffer receiveBuffer = null;  
        try {
```

```

        sndBuffer.setString("abc/ABC");
        receiveBuffer = service.tpcall(sndBuffer, serviceName,
                                       new WebtAttribute());
        System.out.println("receive data = "
                           + receiveBuffer.getString());
    }
    catch (WebtException e) {
        e.printStackTrace();
    }
    catch (ConnectorException e) {
        e.printStackTrace();
    }
    finally {
        webtEndpoint.stopClientManager();
    }
}
public static void main(String[] args) {
    new TpcallSample().excute();
}
}

```

## tpacallの例

tpacallの場合、サービスの要求に対する応答を受けるためにWebtEventHandlerを実装したコールバック・インターフェースを、tpacall呼び出し時に設定する必要があります。Tmaxから正常応答を受ける場合、WebtEventHandlerのhandleEventメソッドが呼び出され、エラーが発生する場合はhandleErrorメソッドが呼び出されます。WebtEventHandlerは、[SampleWebtEventHandler](#)の例を参照してください。

```

public class TpacallSample {
    public final String TMAX_HOST_ADDR = "192.168.1.48";
    public final int TMAX_JGW_PORT = 8881;
    public final int MAX_THREAD = 20;
    public final int MAX_CONNECTION = 100;
    public final int BLOCK_TIME = 60;
    public String serviceName = "TOUPPER";
    public TpacallSample() {
    }
    public void excute() {
        WebtEndpoint webtEndpoint = new WebtEndpointImpl();
        webtEndpoint.setMainServerInfo(TMAX_HOST_ADDR, TMAX_HOST_PORT);
        webtEndpoint.setUseOnlyTpacall(true);
        try {
            webtEndpoint.startClientManager("WebT-001", MAX_THREAD,
                                           MAX_CONNECTION);
        }
        catch (IOException e) {

```

```

        e.printStackTrace();
    }
    WebtNService service = new WebtNServiceImpl(webtEndpoint);
    SampleWebtEventHandler handler = new SampleWebtEventHandler();
    WebtBuffer sndBuffer = new WebtStringBuffer();
    WebtBuffer receiveBuffer = null;
    try {
        sndBuffer.setString("abc/ABC");
        service.tpacall(sndBuffer, serviceName, new WebtAttribute(), handler);

    }
    catch (WebtException e) {
        e.printStackTrace();
    }
    catch (ConnectorException e) {
        e.printStackTrace();
    }
    int checker = 0;
    while (true) {
        if (handler.isSuccess()) {
            receiveBuffer = handler.getReceiveBuffer();
            break;
        }
        else {
            if (handler.isFail()) {
                break;
            }
            else {
                System.out.println("Data don't receivedyet " + (checker++));
                try {
                    Thread.sleep(1000);
                }
                catch (InterruptedException e) {
                    e.printStackTrace();
                }
                if (checker >= BLOCK_TIME) {
                    System.out.println("Block Time over("+BLOCK_TIME+"");
                    break;
                }
            }
        }
    }
    webtEndpoint.stopClientManager();
    if (receiveBuffer != null) {
        System.out.println("receive data = " +
            receiveBuffer.getString());
    }
}

```

```

    }
    public static void main(String[] args) {
        new TpacallSample().excute();
    }
}

```

## Xatpacallの例

WebTAsyncはXAトランザクションのみサポートします。ユーザーはWebTAsyncライブラリーが提供するXAリソース関連のAPIを利用してXA処理を行います。

tpacallでサービスを要求し、handleEventにてメッセージ応答を受けた後、service.getUsingXAResource()メソッドを利用して使用中のXAリソースでend、prepre、commitを実行します。XASampleWebtEventHandlerは、[XASampleWebtEventHandler](#)の例を参照してください。

```

public class XaTpacallSample {
    public final String TMAX_HOST_ADDR = "192.168.1.48";
    public final int TMAX_JGW_PORT = 8350;
    public final int MAX_THREAD = 20;
    public final int MAX_CONNECTION = 100;
    public final int BLOCK_TIME = 60;
    public String serviceName = "FDLINS";
    public XaTpacall() {
    }
    public void excute() {
        WebtEndpoint webtEndpoint = new WebtEndpointImpl();
        webtEndpoint.setMainServerInfo(TMAX_HOST_ADDR, TMAX_JGW_PORT);
        webtEndpoint.setRecoverHandler(new RecoverWebtHandler());
        try {
            webtEndpoint.startClientManager("WebT-001",
                MAX_THREAD, MAX_CONNECTION);
        }
        catch (IOException e) {
            e.printStackTrace();
        }
        WebtNSService service = new
            WebtNSServiceImpl(webtEndpoint);
        XAResource resource = service.getXAResource();
        XASampleWebtEventHandler handler = new
            XASampleWebtEventHandler(service);
        Xid xid = getUniqueXid();
        try {
            resource.start(xid, XAResource.TMNOFLAGS);
        }
        catch (XAException e) {
            e.printStackTrace();
        }
    }
}

```

```

WebtBuffer sndBuffer = getEmployeeFieldBuffer(1111);
WebtBuffer receiveBuffer = null;
try {
    service.tpacall(sndBuffer, serviceName, new
        WebtAttribute(), handler);
}
catch (WebtException e) {
    e.printStackTrace();
}
catch (ConnectorException e) {
    e.printStackTrace();
}
int checker = 0;
while (true) {
    if (handler.isSuccess()) {
        receiveBuffer = handler.getReceiveBuffer();
        break;
    }
    else {
        if (handler.isFail()) {
            break;
        }
        else {
            System.out.println("Data don'treceived yet "
+
                (checker++));
            try {
                Thread.sleep(1000);
            }
            catch (InterruptedException e) {
                e.printStackTrace();
            }
            if (checker >= BLOCK_TIME) {
                System.out.println("Block Timeover
                    (" + BLOCK_TIME+ ")");
                break;
            }
        }
    }
}
webtEndpoint.stopClientManager();
if (receiveBuffer != null) {
    System.out.println("receive buffer = " +receiveBuffer);
}
}
private int getRandomIndex() {
    return new Random().nextInt(10);
}

```

```

    }
    private Xid getUniqueXid() {
        byte[] formatId = new byte[4];
        formatId[0] = (byte) getRandomIndex();
        formatId[1] = (byte) getRandomIndex();
        formatId[2] = (byte) getRandomIndex();
        formatId[3] = (byte) getRandomIndex();
        int globalId = getRandomIndex() * 100;
        int branchId = getRandomIndex() * 100;
        return new TmaxXid(formatId, globalId, branchId);
    }
    private WebtBuffer getEmployeeFieldBuffer(int empNo) {
        WebtBuffer sendBuffer = new WebtFieldBuffer();
        sendBuffer.createField("EMPNO").add(empNo);
        sendBuffer.createField("ENAME").add("tmax001");
        sendBuffer.createField("JOB").add("qa");
        sendBuffer.createField("MGR").add(7839);
        sendBuffer.createField("SAL").add(1000);
        sendBuffer.createField("COMM").add(100);
        sendBuffer.createField("DEPTNO").add(20);
        return sendBuffer;
    }
    public static void main(String[] args) {
        new XaTpacallSample().excute();
    }
}

```

WebtEndpointは、TmaxとCLHまたはAsync Javaゲートウェイを通じて接続できますが、XAサービスを要求する場合は、必ず**Async Javaゲートウェイ**・ポートで接続します。

CLHで接続して非XAサービスを要求する場合は、WebtEndpointに接続した後、下記のとおり設定する必要があります。

```
webtEndpoint.setUseOnlyTpacall(true);
```

## SampleWebtEventHandlerの例

WebtEventHandler interfaceインターフェースを実装したクラスで、サービス要求に対する応答結果を受信して成功する場合はhandleEventが呼び出されます。失敗する場合は、handleErrorが呼び出されます。

```

public class SampleWebtEventHandler implements WebtEventHandler {
    private WebtBuffer receiveBuffer;
    private boolean success;
    private boolean fail;
    public SampleWebtEventHandler() {
    }
}

```

```

public void handleError(WebtException we) {
    fail = true;
    System.out.println("handleError = " + we.toString());
}
public void handleEvent(int type, WebtBuffer buffer, int len,int flags) {

    receiveBuffer = buffer;
    success = true;
    System.out.println("handleEvent");
}
public WebtBuffer getReceiveBuffer() {
    return receiveBuffer;
}
public boolean isSuccess() {
    return success;
}
public boolean isFail() {
    return fail;
}
}

```

## XASampleWebtEventHandlerの例

XAサービス処理のため、WebtEventHandlerインターフェースを実装したクラスで、サービス要求に対する応答結果を受信して成功する場合はhandleEventが呼び出されます。失敗する場合は、handleErrorが呼び出されます。

```

public class XASampleWebtEventHandler implements WebtEventHandler {
    private WebtNService service;
    private boolean fail;
    private boolean success;
    private WebtBuffer receiveBuffer;
    public XASampleWebtEventHandler(WebtNService service) {
        this.service = service;
    }
    public void handleError(WebtException we) {
        try {
            IWebtNXAResource resource
                = service.getWebtNUsingXAResource();
            resource.end(service.getXid(), XAResource.TMFAIL);
        } catch (XAException e) {
            e.printStackTrace();
        }
        fail = true;
        System.out.println("handleError = " + we.toString());
    }
}

```

```

public void handleEvent(int type, WebtBuffer buffer, int len,int flags) {

    try {
        IWebtNXAResource resource
        = service.getWebtNUsingXAResource();
        Xid xid = service.getXid();
        resource.end(xid, XAResource.TMSUCCESS);
        resource.prepare(xid);
        resource.commit(xid, true);
    } catch (XAException e) {
        e.printStackTrace();
    }
    receiveBuffer = buffer;
    success = true;
    System.out.println("handleEvent");
}
public WebtBuffer getReceiveBuffer() {
    return receiveBuffer;
}
public boolean isFail() {
    return fail;
}
public boolean isSuccess() {
    return success;
}
}

```

## 2.2.3. 主要機能

### タイムアウト設定

メッセージを転送する際、受信を待つ時間とTmaxに接続をトライする時間は、WebtEndpointの **setTimeout**(long connectTimeout, long readTimeout)で設定します。単位は、millisecondです。

setTimeoutメソッドはグローバル的に設定する方法です。メッセージ転送ごとの設定は、WebtNServiceの下記のメソッドを使用します。

```

tpacall(WebtBuffer sndBuffer, String serviceName, WebtAttribute attribute,
        long readtimeout, WebtEventHandler handler)

```

タイムアウトが発生した場合は、ユーザー・ハンドラーのhandleErrorを呼び出します。詳細内容は、[tpcallの例](#)を参照してください。



## フェイルオーバー

WebtAsyncライブラリーは、tpacall時に接続されたコネクトが存在しない場合は再接続を試みます。WebtEndPointのsetMainServerInfoで設定したメイン・サーバーに接続できない場合は、setFailoverServerInfoに設定されているバックアップ・サーバーに接続を試みます。詳細内容は、[tpcallの例](#)を参照してください。

## フェイルバック

バックアップ・サーバーに接続されている状態でメイン・サーバーが復旧されたら、メイン・サーバーに再接続するようにWebtEndPointはstartCheckFailBackをサポートします。このメソッドは、バックアップ・サーバーに接続されている場合、メイン・サーバーの状況をチェックしてユーザーに通知する役割をします。ユーザーはパラメータとして入るFailBackMonitorを実装し、availableMainServerが呼び出されたらメッセージをサスペンド(suspend)します。なお、すべてのメッセージが受信された後に接続を終了したら、次のメッセージはメイン・サーバーに渡されます。メイン・サーバーの復旧可否をチェックするためにスレッドを1つ作成します。詳細内容は、[tpcallの例](#)を参照します。

```
public class SampleFailBackMonitor implements FailBackMonitor {
    private WebtEndpoint webtEndpoint;
    public SampleFailBackMonitor(WebtEndpoint webtEndpoint) {
        this.webtEndpoint = webtEndpoint;
    }
    public void availableMainServer() {
        webtEndpoint.disconAllConnections();
        System.out.println("Disconnected all connection!!");
    }
}
```

## トランザクション・リカバリー

トランザクション処理中、システム障害によってTmaxにペンディング(pending)されたトランザクションが発生した場合、再接続を確立する際にペンディングされたトランザクションを処理するため、WebtAsyncライブラリーはTmaxでペンディングされたトランザクションのxidリストを取得して、ユーザー・アプリケーションに当該トランザクションのデシジョン(decision)を問い合わせます。

```
public class RecoverSampleWebtHandler implements WebtNRecoverHandler {
    public RecoverSampleWebtHandler() {
    }
    public Xid[] recoverRecieved(Xid[] xids) {
        TmaxXid xid = null;
        for (int i = 0; i < xids.length; i++) {
            xid = (TmaxXid) xids[i];
        }
    }
}
```

```

        xid.setDecision(Webt.TM_XA_COMMIT);
        // xid.setDecision(Webt.TM_XA_RBACk);
    }
    return xids;
}
}

```

## ログ関連機能

セクター・スレッドでオプションに従って、要求構文(Tmaxヘッダーを除く)のログレベルを問わず、指定した長さの分だけHexaタイプでログを残します。

```

webtasync.appmsg.isdump=<true | false>
webtasync.appmsg.dump.length=<Length>

```

- webtasync.appmsg.isdump=<true | false>

機能を有効にします。

- webtasync.appmsg.dump.length=<Length>

要求構文の中から、出力するデータの長さを設定します。

---

### 参考

ユーザー・アプリケーションは、WebtNRecoverHandlerのrecoverRecievedメソッドを実装する場合、デシジョンを定義する必要があります。詳細内容は、[Xatpacallの例](#)を参照してください。

---

# 索引

## J

JTmaxNServerにサービス要求, 6

JTmaxNServer環境設定, 3

com.tmax.inbound.properties設定, 4

TmaxInboundEApplication.ear

/META-INF/weblogic-application.xml設定, 4

webt.properties設定, 4

## T

Tmaxクライアント・サービス要求, 5

## W

WebTAsync, 1

## あ

アウトバウンド, 2

アウトバウンド・サービスの主な機能

タイムアウト設定, 16

トランザクション・リカバリー, 17

フェイルオーバー, 17

フェイルバック, 17

ログ関連機能, 18

アウトバウンド・サービス・クライアント・プログラムの例

SampleWebtEventHandlerの例, 14

tpacallの例, 10

tpcallの例, 9

XASampleWebtEventHandlerの例, 15

Xatpacallの例, 12

アウトバウンド・サービス環境設定, 8

Tmax環境設定, 8

アウトバウンド構造, 2

インバウンド, 1

インバウンド・サービス環境設定, 3

JTmaxNServer環境設定, 3

Tmax環境設定, 3

インバウンド構造, 1

## た

タイムアウト設定, 16

トランザクション・リカバリー, 17

## は

フェイルオーバー, 17

フェイルバック, 17

## ら

ログ関連機能, 18

