

Tmax

WebTJCA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 WebTJCA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. JCA	1
1.2.1. JCAの構造	2
1.2.2. WebTJCA	3
第2章 WebTJCAのAPI	5
2.1. CCI	5
2.1.1. コネクション関連のAPI	6
2.1.2. サービス要求のためのAPI	7
2.1.3. データ表現API	7
2.2. トランザクション	8
2.2.1. XAトランザクション	8
2.2.2. ローカル・トランザクション	9
2.3. ロギング(Logging)	9
2.4. 非同期要求と対話型通信のサポート	9
第3章 WebTJCAのインバウンド通信	11
3.1. インバウンド通信	11
3.2. ワーカー・スレッド	12
3.3. メッセージ・ドリブン・ビーン	13
3.4. ロギング	14
第4章 例	15
4.1. JEUS 6でのWebTJCAの使用	15
4.1.1. リソース・アダプターの構成	15
4.1.2. アウトバウンドの例	19
4.1.3. インフローの例	20
4.2. WebLogicでのWebTJCAの使用	25
4.2.1. リソース・アダプターの構成	25
4.2.2. アウトバウンドの例	27
4.2.3. インフローの例	27
付録 A. WebTJCA ra.xmlの設定	31
A.1. フィールド情報	31
A.1.1. <resourceadapter-class>	31
索引	35

図目次

[図 1.1]	JCAの構造	2
[図 1.2]	インバウンド通信	3
[図 3.1]	インバウンド通信モデル	11
[図 3.2]	インバウンド通信の構造	13

このガイドについて

対象読者

本書は、Tmax[®](以下、Tmax)のWebTJCAを使用して開発を行う開発者を対象としています。

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(Middleware)およびUNIXシステムについて
- Tmaxの基本概念について
- Java、Cプログラミングについて
- JEUSとWebLogicについて

制限事項

本書では、JEUSとWebLogicについては説明していません。したがって、関連内容については両製品のガイドを参照してください。また、実務上の具体的な使用方法や管理・運用については、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4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

WebTJCAについて紹介します。

- 第2章: WebTJCA API

アウトバウンド通信を開発するためにCCIを実装したWebTJCA APIについて説明します。

- 第3章: WebTJCAのインバウンド通信

WebTJCAのインバウンド通信について説明します。

- 第4章: 例

JEUS 6とWebLogicでWebTJCAを使用する方法について例を挙げて説明します。

- 付録A: WebTJCA ra.xmlの設定

WebTJCA ra.xmlの設定時に使用するフィールド情報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
<div>参考</div>	参照/注意事項
[図1.1]	図の名前
[表1.1]	表の名前
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
JEUS	JCAガイド
WebLogic	Programming WebLogic Resource Adapters Deploying Applications to WebLogic Server

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 紹介

本章では、JCAの仕様およびWebTJCAについて紹介します。

1.1. 概要

WebTJCAは、J2EEのJCA(J2EE Connector Architecture)仕様でリソース・アダプターの役割をするライブラリー・パッケージです。ユーザーはこのライブラリーを介してTmaxと通信することができます。

既存のWebTは、JEUSを介するのが前提になるか、そうでなければ単一アプリケーションとしてのみ動作可能であったため、開発者は特定の設定とAPIを利用してTmaxとの通信を行っていました。

WebTJCAは、JCA仕様を実装することで、ユーザーが仕様のみ知っていればWebTの動作方法を知らなくてもTmaxとのサービス要求の送受信が可能です。

1.2. JCA

JCAは、ERP(Enterprise Resource Planning)やレガシー情報システムなどを含む非Webアプリケーション・サーバー(WAS)の環境サービス(エンタープライズ・システム、以下EIS)との連携のために定義された標準フレームワークです。

JCAの登場以前までは、各ベンダーのEISおよびWebアプリケーション・サーバーごとに別途のインターフェースを実装するカスタム・ドライバの実装方法を使用してEISとの連動を実現していました。そのため、特定製品間で連動そのものができなくなる場合を除いても、ドライバごとに多様なインターフェースと開発手法が存在し、それに従わなければならない問題が付きまとっていました。

N * M問題とも呼ばれるこうした状況は、基本的にコードレベルを修正しなければ連動が不可能な状態につながってしまい、それによりJ2EE環境の移植性と拡張性に深刻な制約をもたらしてしまいます。この問題を解決するために、JCA仕様をもって、リソース・アダプターとWebアプリケーション・サーバー間のインターフェースの相互作用を確定しました。そして、標準に適合したWebアプリケーション・サーバーとリソース・アダプターであれば、コードレベルを修正しなくても相互に互換して動作することを目標に発展されました。各EISベンダーのカスタム・ドライバを標準化された構造(Architecture)に置き換えることで、既存に発生していたN * Mの連動問題をN + Mまで低減させることが可能になりました。

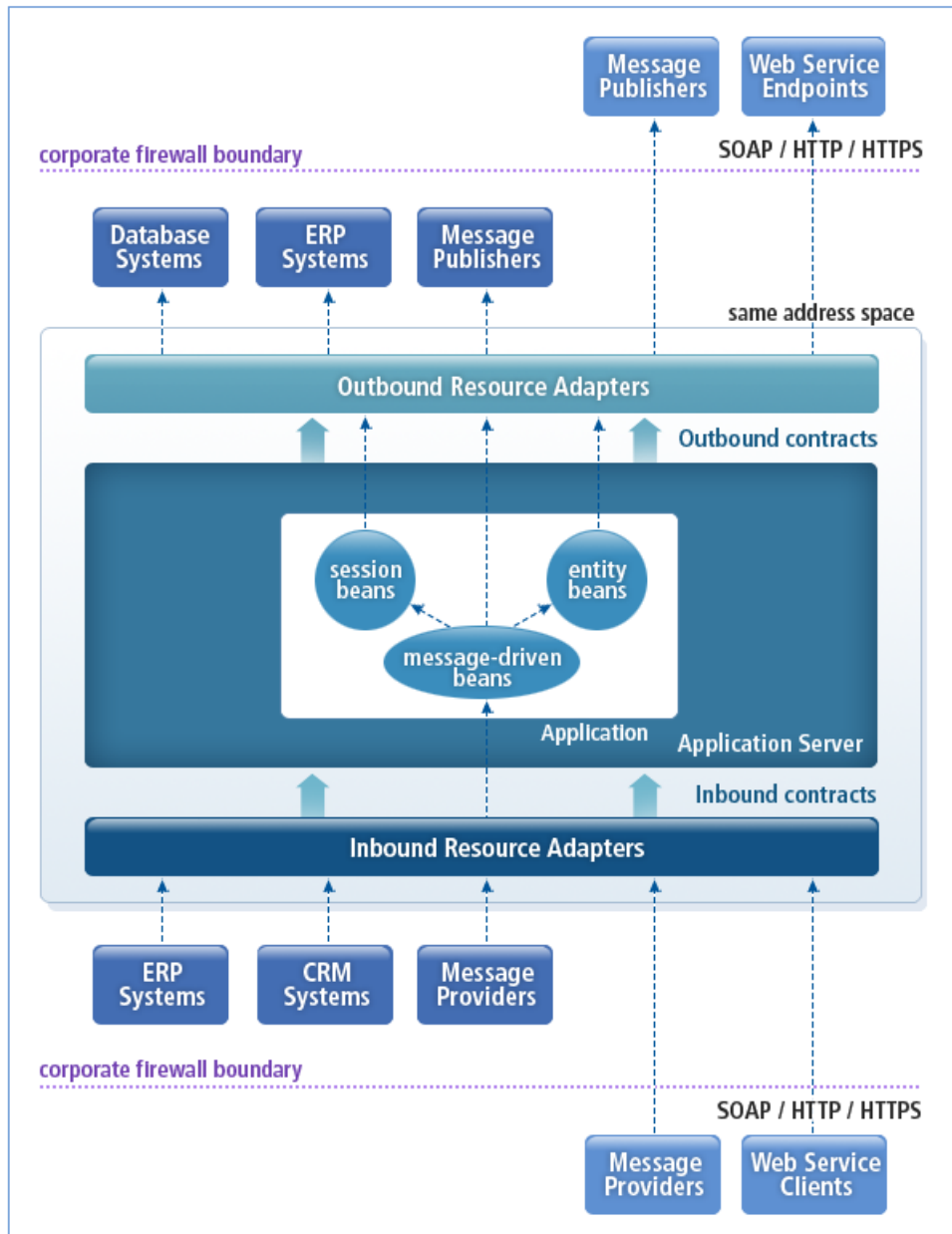
参考

Connector 1.5の仕様は、<http://java.sun.com/j2ee/connector/download.html>よりダウンロードできます。

1.2.1. JCAの構造

以下の図は、JCAの構造です。

[図 1.1] JCAの構造



- アウトバウンド通信

Webアプリケーション・サーバーからEISにサービスを要求することを**アウトバウンド通信**といいます。

JCA仕様により、アプリケーション開発者にCCI(Common Client Interface)仕様が提供されるので、開発者はEISと連動されているアプリケーションを開発することができます。リソース・アダプターはCCI、SPI仕様を実装して提供する必要があります。

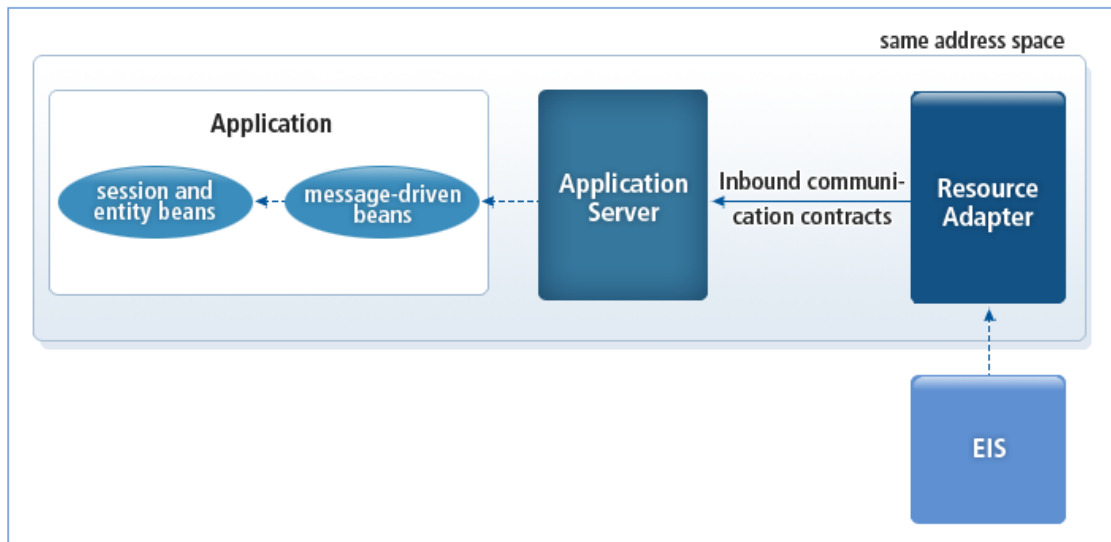
- インバウンド通信

EISからWebアプリケーション・サーバーにサービスを要求することを**インバウンド通信**といいます。

JCA 1.5仕様は、Webアプリケーション・サーバーからEISに要求するサービスだけでなく、EISからWebアプリケーション・サーバーにもサービスが要求可能な仕様を提供します。

以下は、インバウンド通信の構造です。

[図 1.2] インバウンド通信



1.2.2. WebTJCA

WebTJCAはJCA仕様を実装したライブラリーで、webt5.9.7.0.jarよりデプロイされます。このライブラリーにはJCA仕様を実装した部分だけでなく、既存WebTで使われていたライブラリーも含まれており、既存のWebTも使用可能です。またWebTJCAは、アウトバウンドとインバウンド通信を両方ともサポートします。

- アウトバウンド通信

開発者は、webt5.9.7.0.jarを使用し、ra.xmlとWebアプリケーション・サーバー・ベンダーの固有な設定を利用してリソース・アダプターを作成し、Webアプリケーション・サーバーにデプロイする必要があります。

アプリケーション開発者は、CCIを利用してサービス要求を行うアプリケーションを作成することができます。

- インバウンド通信

開発者は、webt5.9.7.0.jarを使用し、ra.xmlとWebアプリケーション・サーバー・ベンダーの固有な設定を利用してリソース・アダプターを作成し、Webアプリケーション・サーバーにデプロイする必要があります。

アプリケーション開発者は、メッセージ・ドリブン・ビーン(Message-driven Bean)を作成し、メッセージ受信部にて実際の要求対象のEJBを要求します。

第2章 WebTJCAのAPI

本章では、アウトバウンド通信を開発するためにCCIを実装したWebTJCA APIについて説明します。

2.1. CCI

CCIはリソース・アダプターの使用に必要な標準クライアントAPIを定義したスペックです。ユーザーはCCIを利用し、多様なリソース・アダプターに対しても同じアプリケーション・コードの作成が可能です。

JCAスペックにおいてCCIは必須の実装条件ではありません。ユーザーはCCIを利用してサービスの呼び出しを行ってもいいですが、ベンダーごとに固有のスペックがあるのでそれを利用してのアプリケーションの作成も可能です。

スペックで実装すべきインターフェースは下記のとおりです。WebTJCAでは以下のCCIを利用してアプリケーションを実装できます。

- アプリケーションで作成するコネクション関連のインターフェース
 - `javax.resource.cci.connectionFactory`
 - `javax.resource.cci.Connection`
 - `javax.resource.cci.ConnectionSpec`
 - `javax.resource.cci.LocalTransaction`
- EISとの相互動作のためのインターフェース
 - `javax.resource.cci.Interaction`
 - `javax.resource.cci.InteractionSpec`
- インフロー(Inflow)構造において、メッセージ・ドリブン・ビーンで実装するメッセージ・リスナー(Message listener)インターフェース
 - `javax.resource.cci.MessageListener`
- EISと送受信するデータを表現するためのインターフェース
 - `javax.resource.cci.Record`

- javax.resource.cci.MappedRecord
- javax.resource.cci.IndexedRecord
- javax.resource.cci.RecordFactory
- javax.resource.cci.Streamable
- javax.resource.cci.ResultSet
- java.sql.ResultSetMetaData
- リソース・アダプターの実装とEISコネクション関連の基本情報を提供するメタデータに関するインターフェース
 - javax.resource.cci.ConnectionMetaData
 - javax.resource.cci.ResourceAdapterMetaData
 - javax.resource.cci.ResultSetInfo
- 追加インターフェース
 - javax.resource.ResourceException
 - javax.resource.cci.ResourceWarning

参考

インターフェースのメソッドについては、JDK JavaDocを参照してください。

2.1.1. コネクション関連のAPI

リソース・アダプターのデプロイを行った後は、ユーザーはJNDIを利用して**ConnectionFactory**を参照することができ、ConnectionFactoryを利用したコネクションの作成も可能です。**getConnection()**を呼び出すとき、新しいコネクションを作成するか、すでに作成したコネクションを返します。

以下は、WebTJCAがコネクションを使用する例です。

```
ConnectionFactory connFactory = (ConnectionFactory)ctx.lookup("TmaxConnector");
Connection conn = connFactory.getConnection();
```

2.1.2. サービス要求のためのAPI

サービス要求のためのInteractionインターフェースはConnectionインターフェースで作成します。どのサービス呼び出すかは、InteractionSpecを実装したクラスを利用します。InteractionSpecを作成するインターフェースはスペックが存在せず、WebTJCAでは**TmaxInteractionSpecImpl**を利用して作成します。WebTJCAは同期呼び出し(tpcall)のみ提供します。

以下は、サービス要求に関する例です。

```
Interaction action = conn.createInteraction();
TmaxInteractionSpecImpl aspect = new TmaxInteractionSpecImpl();
aspect.setSvcName("TOUPPER");
aspect.setAction(InteractionSpec.SYNC_SEND_RECEIVE);

RecordFactory recordFactory = connFactory.getRecordFactory();
MappedRecord sndBuf = recordFactory.createMappedRecord(null);
sndBuf.put("$String", "abc");
MappedRecord rcvBuf = recordFactory.createMappedRecord(null);

if( action.execute(aspect, sndBuf, rcvBuf) )
{
    String result = (String)rcvBuf.get("$String");
    System.out.println(result);
}
```

2.1.3. データ表現API

WebTJCAはTmaxで提供するデータのうち、String型、CArray型、FDL型のデータを提供します。

getRecordFactory()メソッドを利用してConnectionFactoryからRecordFactoryを取得し、**createMappedRecord()**でデータ・オブジェクトを作成して使用するデータを入力します。

以下は、RecordFactoryを作成する例です。

```
RecordFactory recordFactory = connFactory.getRecordFactory();
MappedRecord sndBuf = recordFactory.createMappedRecord(null);
```

以下は、各タイプ別にメソッドを使用する方法です。

- String型の使用

MappedRecordの**put**メソッドを利用して設定します。putのキーとして"\$String"を入力する場合は値をStringで認識し、TmaxにString型のメッセージを渡します。get関数を利用する場合にもキーとして"\$String"を入力すれば、保存されているStringオブジェクトを返します。以下は、Stringを使用してデータを表現する例です。

```
MappedRecord sndBuf = recordFactory.createMappedRecord(null);
sndBuf.put("$String", "abc");
```

- CArray型の使用

MappedRecordの**put**メソッドを利用して設定します。putのキーとして"\$CArray"を入力する場合は値をbyte[]で認識し、TmaxにCArray型のメッセージを渡します。get関数を利用する場合にもキーとして"\$String"を入力すれば、保存されているbyte[]を返します。以下は、CArrayを使用してデータを表現する例です。

```
MappedRecord sndBuf = recordFactory.createMappedRecord(null);
sndBuf.put("$CArray", new byte[] {0, 1, 2});
```

- FDL型の使用

MappedRecordの**put**メソッドを利用して設定します。FDLにて設定したキーをputのキーとして入力する場合は値をその設定キーに該当するフィールド型として認識し、TmaxにFDL型のメッセージを渡します。putメソッドを利用してフィールドキーとデータをペアにして保存します。以下は、FDLを使用してデータを表現する例です。

```
MappedRecord sndBuf = recordFactory.createMappedRecord(null);
sndBuf.put("ID", "tmax001");
sndBuf.put("PASS", "abcd");
sndBuf.put("AGE", 16);
```

putメソッドで保存したデータを参照するには、getメソッドを利用します。getメソッドの返却オブジェクトは当該フィールドの値を保存しているIndexedRecordを実装したcom.tmax.connector.cci.TmaxFdlRecordを返します。ユーザーはIndexedRecordのget(int index)メソッドを利用して保存したフィールド値を参照します。

```
IndexedRecord field = sndBuf.get("ID");
String id = field.get(0);
```

2.2. トランザクション

2.2.1. XAトランザクション

WebTJCAはXAトランザクションをサポートします。XAトランザクション処理を行うには、ra.xmlで<support-xa>項目をtrueに設定し、<transaction-support>項目もXAトランザクションとして設定してください。

XAトランザクションは設定するだけでgetConnectionの実行時に自動でトランザクションに参加します。アプリケーションでコミットやロールバックを行う場合は、2相コミットの方法でTmaxにPrepareおよびCommitの要求をします。

リカバリー処理を行うには、必ず<outbound-resourceadapter><connection-definition>にxidマッピングファイル関連の情報を入力する必要があります。このファイルが存在しない場合は、Webアプリケーション・サーバーが再起動する際にxidマッピング情報が存在せず、リカバリー作業が実行できません。

XAトランザクション処理を行うには、Tmaxのjeusgwを設定して当該ポートに接続します。

2.2.2. ローカル・トランザクション

ローカル・トランザクション処理を行うには、ConnectionでgetLocalTransactionメソッドを利用してください。設定によって、自動でWebアプリケーション・サーバーより処理を実行します。

以下は、ローカル・トランザクションを使用する例です。

```
ConnectionFactory connFactory = (ConnectionFactory)ctx.lookup("TmaxConnector");
Connection conn = connFactory.getConnection();
conn.getLocalTransaction().begin();
....

conn.getLocalTransaction().commit();
```

ローカル・トランザクションを行うには、Tmaxの基本ポートへの接続が必要です。JAVAGWIに接続してはローカル・トランザクションとして処理することができません。

2.3. ロギング(Logging)

リソース・アダプターを使用して発生するログについては、ra.xmlの使用時に設定したログファイルとログレベルが適用された状態で作成します。<outbound-resourceadapter><connection-definition>に、ログ関連設定においてConnectionを使用するログを出力します。詳細については、「[4.1.1. リソース・アダプターの構成](#)」の「ra.xmlの設定」を参照してください。

2.4. 非同期要求と対話型通信のサポート

現在定義したCCIスペックだけではTmaxでサポートする非同期要求と対話型通信をサポートできないので、次のように既存WebT APIを利用して実装する必要があります。

ConnectionからWebtConnectionを参照し、**WebtRemoteService()**や**WebtDialogueService()**を利用して非同期、対話型通信を行うことができます。

以下は、非同期要求の例です。

```
ConnectionFactory connFactory = (ConnectionFactory)ctx.lookup("TmaxConnector");
Connection conn = connFactory.getConnection();
WebtConnection webtConn = ((TmaxConnectionImpl)conn).getInConnection();
WebtRemoteService svc = new WebtRemoteService("TOUPPER", webtConn);
int cd = svc.tpacall(sendBuf);
```

```
rcvBuf = svc.tpgetrply(cd);  
conn.close();
```

以下は、対話型通信の例です。

```
ConnectionFactory connFactory = (ConnectionFactory)ctx.lookup("TmaxConnector");  
Connection conn = connFactory.getConnection();  
WebtConnection webtConn = ((TmaxConnectionImpl)conn).getInConnection();  
WebtDialogueService svc = new WebtDialogueService("TOUPPER_CONV", webtConn);  
svc.tpconnect(sndbuf..  
....  
conn.close();
```

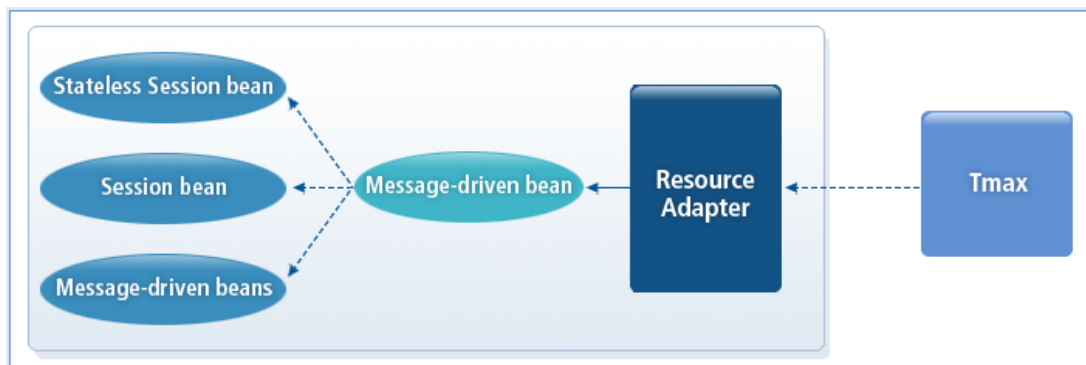
第3章 WebTJCAのインバウンド通信

本章では、WebTJCAのインバウンド通信について説明します。

3.1. インバウンド通信

以下は、インバウンド通信の処理プロセスです

[図 3.1] インバウンド通信モデル



Tmaxリソース・アダプターは、TmaxからWebアプリケーション・サーバーに渡される要求に対してどのサービス呼び出す必要があるかが特定できません。したがって、途中でメッセージ・ドリブン・ビーンを追加し、リソース・アダプターがメッセージ・ドリブン・ビーンにメッセージを渡す上、ユーザーがメッセージ・ドリブン・ビーンで各業務に適合したサービス呼び出せるように作成しなければなりません。リソース・アダプターにおいてどのメッセージ・ドリブン・ビーンにメッセージを渡すかは、次の設定で定義します。

ra.xml

ra.xmlの<activation-spec-class>にcom.tmax.connector.spi.TmaxActivationSpecと書き込み、<config-property-name>にはserviceNameと書き込んで設定します。

```
<inbound-resourceadapter>
<messageadapter>
  <messagelistener>
    <messagelistener-type>javax.resource.cci.MessageListener</messagelistener-type>

  <activation-spec>
    <activation-spec-class>com.tmax.connector.spi.TmaxActivationSpec
  </activation-spec-class>
  <required-config-property>
```

```

    <config-property-name>serviceName</config-property-name>
  </required-config-property>
</activationspec>
</messagelistener>
</messageadapter>
</inbound-resourceadapter>

```

メッセージ・ドリブン・ビーンのejb-jar.xml

ユーザーが開発したメッセージ・ドリブン・ビーンにおいて、ejb-jar.xmlに<message-destination-link>と<activation-config>を書き込んでデプロイを行う場合、当該リソース・アダプターに<activation-config-property>を知らせます。お知らせを受けたTmaxリソース・アダプターは、TmaxがserviceNameに入力したサービスを要求する場合、登録されているメッセージ・ドリブン・ビーンにメッセージを渡します。

```

<message-destination-link>TmaxRA</message-destination-link>
<activation-config>
  <activation-config-property>
    <activation-config-property-name>serviceName</activation-config-property-name>

    <activation-config-property-value>ECHOSTRING,CREATE_ACCOUNT_B,TRANSFER_MONEY_B

  </activation-config-property-value>
</activation-config-property>
</activation-config>

```

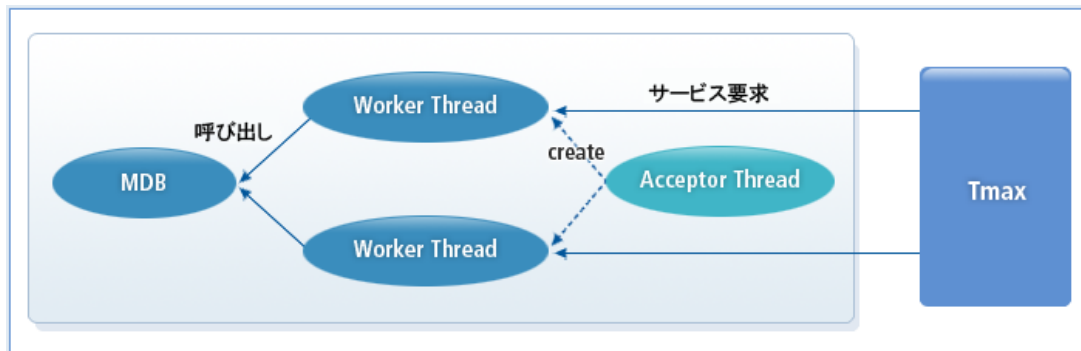
3.2. ワーカー・スレッド

インバウンド通信は、設定されたポートを取得するための1つのスレッドが作成され、取得したソケットを読み込むためのスレッドがソケットごとにもう1つ作成されます。作成されるスレッド数は、アプリケーション・サーバーのワーカー・マネージャーに最大値として設定されます。設定は各ベンダーのルールに従います。

Tmaxクライアントからの要求に対して、RAのアクセプタ・スレッドでそれぞれのワーカー・スレッドを作成します。作成された各ワーカー・スレッドが該当する要求をMDBに転送すると、MDBでその要求を区別して処理します。TmaxのJAVAGWと接続する必要があります。

以下の図は、インバウンド通信の構造です。

[図 3.2] インバウンド通信の構造



3.3. メッセージ・ドリブン・ビーン

メッセージ・ドリブン・ビーンの作成時には、`javax.resource.cci.MessageListener`を実装する必要があります。

以下は、メッセージ・ドリブン・ビーンの設定例です。

```
public Record onMessage(Record in) throws ResourceException {
    String svcName = (String)((MappedRecord)in).get("$ServiceName");

    if( svcName.equalsIgnoreCase("echostring") ) {
        return in;
    }
    else if( svcName.equalsIgnoreCase("create_account_b")) {
    }
    else if( svcName.equalsIgnoreCase("transfer_money_b")) {
    }
    return in;
}
```

- `public Record onMessage(Record in) throws ResourceException`

リソース・アダプターが要求を渡す際に呼び出されるメソッドです。レコードはMappedRecordで渡します。ユーザーはスペシャル・キーを使ってサービス名や特定の値を設定および参照することができます。OnMessageの戻り値が、Tmaxから要求元に渡される応答メッセージとなります。

成功か失敗かを知らせるため、戻り値に\$ReturnCode、\$UserCodeを入力します。成功時には\$ReturnCodeを0と設定し、失敗時には\$ReturnCodeを-1と設定した後、\$UserCodeにTmaxが参照可能なユーザー・コードを入力します。

```
// 成功時
MappedRecord out = (MappedRecord) in.clone();
out.put("$ReturnCode", new Integer(0));
```

```
// 失敗時
out.put("$ReturnCode", new Integer(-1));
out.put("$UserCode", new Integer(usrcode));
```

3.4. ロギング

ra.xmlの<resourceadapter>でログ関連の設定を行うとき、インバウンド作業に関するログを出力します。

インバウンド・ロギングの設定時には、ra.xmlに以下のような設定を追加します。

```
<resourceadapter>
  <config-property>
    <config-property-name>name</config-property-name>
    <config-property-type>java.lang.String</config-property-type>
    <config-property-value>tmaxin1</config-property-value>
  </config-property>
</resourceadapter>
```

第4章 例

本章では、JEUS 6とWebLogicでWebTJCAを使用する方法について例を挙げて説明します。

4.1. JEUS 6でのWebTJCAの使用

4.1.1. リソース・アダプターの構成

JCAを利用するには、その前提としてリソース・アダプターの構成が必要です。リソース・アダプターにはWebアプリケーション・サーバーを提供するベンダーの固有の設定と、JCAで定義した設定を定義する必要があります。リソース・アダプターはJava Archive(JAR)の形式でRARでパッケージングします。

下記は、JCAで定義したリソース・アダプターに含まれるリストです。

Contents of RAR file	Requirements	Relative location within RAR file
Deployment Descriptor	Required	META-INF/ra.xml
howto.html, image files, etc.	Optional	Arbitrary (that is, could be at root level or sub-level)
JAR files	Optional	Arbitrary
Platform-specific native libraries	Optional	Arbitrary

開発者は、ra.xmlとjeus-connector-dd.xmlを記述します。

参考

ra.xml, jeus-connector-dd.xmlの設定とデプロイ方法の詳細については、『JEUS JCAガイド』を参照してください。

ra.xmlの設定

インフローとアウトバウンドを一緒に設定します。以下は、ra.xmlを設定する例です。

```
<?xml version="1.0" encoding="UTF-8"?>
<connector id="Connector_ID" version="1.5" xmlns="http://java.sun.com/xml/ns/j2ee"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
```

```

http://java.sun.com/xml/ns/j2ee/connector_1_5.xsd">
<description>Tmax Sample Resource Adapter</description>
<display-name>TmaxRA</display-name>
<vendor-name>TmaxSoft</vendor-name>
<eis-type>TP Monitor</eis-type>
<resourceadapter-version>1.0.0.0</resourceadapter-version>

<resourceadapter>
<resourceadapter-class>com.tmax.connector.spi.TmaxResourceAdapterImpl
</resourceadapter-class>
<config-property>
<config-property-name>port</config-property-name>
<config-property-type>java.lang.Integer </config-property-type>
<config-property-value>6556</config-property-value>
</config-property>
<config-property>
<config-property-name>fdlfile</config-property-name>
<config-property-type>java.lang.String</config-property-type>
<config-property-value>C:\TmaxSoft\JEUS6.0\lib\application\tmax.fdl
</config-property-value>
</config-property>
<outbound-resourceadapter>
<connection-definition>
<managedconnectionfactory-class>
com.tmax.connector.spi.TmaxManagedConnectionFactoryImpl
</managedconnectionfactory-class>
<config-property>
<config-property-name>host</config-property-name>
<config-property-type>java.lang.String</config-property-type>
<config-property-value>192.168.1.43</config-property-value>
</config-property>
<config-property>
<config-property-name>host_port</config-property-name>
<config-property-type>java.lang.Integer</config-property-type>
<config-property-value>11120</config-property-value>
</config-property>
<config-property>
<config-property-name>backup</config-property-name>
<config-property-type>java.lang.String</config-property-type>
<config-property-value>192.168.1.43</config-property-value>
</config-property>
<config-property>
<config-property-name>backup_port</config-property-name>
<config-property-type>java.lang.Integer</config-property-type>
<config-property-value>15300</config-property-value>
</config-property>
<config-property>

```



```
<config-property-name>support_xa</config-property-name>
<config-property-type>java.lang.Boolean</config-property-type>
<config-property-value>true</config-property-value>
</config-property>
<config-property>
<config-property-name>headerType</config-property-name>
<config-property-type>java.lang.String</config-property-type>
<config-property-value>extendedV4</config-property-value>
</config-property>
<config-property>
<config-property-name>connectTimeout</config-property-name>
<config-property-type>java.lang.Integer</config-property-type>
<config-property-value>15000</config-property-value>
</config-property>
<config-property>
<config-property-name>fdlFile</config-property-name>
<config-property-type>java.lang.String</config-property-type>
<config-property-value>C:\TmaxSoft\JEUS6.0\lib\application\tmax.fdl
</config-property-value>
</config-property>
<config-property>
<config-property-name>xidMapperPath</config-property-name>
<config-property-type>java.lang.String</config-property-type>
<config-property-value>C:\TmaxSoft\JEUS6.0\lib\application\</config-property-value>
</config-property>
<config-property>
<config-property-name>xidMapperFile</config-property-name>
<config-property-type>java.lang.String</config-property-type>
<config-property-value>webtxidmapper</config-property-value>
</config-property>
<config-property>
<config-property-name>logLevel</config-property-name>
<config-property-type>java.lang.String</config-property-type>
<config-property-value>DEBUG</config-property-value>
</config-property>
<config-property>
<config-property-name>logDir</config-property-name>
<config-property-type>java.lang.String</config-property-type>
<config-property-value>d:\</config-property-value>
</config-property>
<config-property>
<config-property-name>logFile</config-property-name>
<config-property-type>java.lang.String</config-property-type>
<config-property-value>webt.log</config-property-value>
</config-property>
<config-property>
<config-property-name>logValidDays</config-property-name>
```

```

<config-property-type>java.lang.Integer</config-property-type>
<config-property-value>1</config-property-value>
</config-property>
<config-property>
<config-property-name>connectionName</config-property-name>
<config-property-type>java.lang.String</config-property-type>
<config-property-value>tmax1</config-property-value>
</config-property>
<connectionfactory-interface>javax.resource.cci.ConnectionFactory
</connectionfactory-interface>
<connectionfactory-impl-class>com.tmax.connector.cci.TmaxConnectionFactoryImpl
</connectionfactory-impl-class>
<connection-interface>javax.resource.cci.Connection</connection-interface>
<connection-impl-class>com.tmax.connector.cci.TmaxConnectionImpl
</connection-impl-class>
</connection-definition>
<transaction-support>XATransaction</transaction-support>
<reauthentication-support>false</reauthentication-support>
</outbound-resourceadapter>
<inbound-resourceadapter>
<messageadapter>
<messagelistener>
<messagelistener-type>javax.resource.cci.MessageListener</messagelistener-type>
<activation-spec>
<activation-spec-class>com.tmax.connector.spi.TmaxActivationSpec
</activation-spec-class>
<required-config-property>
<config-property-name>serviceName</config-property-name>
</required-config-property>
</activation-spec>
</messagelistener>
</messageadapter>
</inbound-resourceadapter>
</resourceadapter>
</connector>

```

jeus-connector-dd.xmlの設定

以下は、jeus-connector-dd.xmlを設定する例です。export-nameはアウトバウンド・コネクションが参照可能なJNDIの名前です。

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<jeus-connector-dd xmlns="http://www.tmaxsoft.com/xml/ns/jeus">
<module-name>TmaxRA</module-name>
<use-workmanager>true</use-workmanager>
<connection-pool>

```

```

<export-name>TmaxConnector</export-name>
<transaction-support>XATransactionOnly</transaction-support>
<pool-management>
<min>5</min>
<max>20</max>
</pool-management>
</connection-pool>
</jeus-connector-dd>

```

デプロイ

- フォルダの構成

```

webt5.9.7.0.jar
META-INF/ra.xml
META-INF/jeus-connector-dd.xml

```

- RARファイルの作成

```
jar TmaxRA.rar
```

- JCAでJEUSにデプロイ

JEUS WebAdminで「アプリケーション・モジュールのデプロイ」を利用してデプロイ

4.1.2. アウトバウンドの例

以下は、TmaxのTOUPPERサービスを呼び出すjspの例です。

```

<%@ page language="java" contentType="text/html; charset=EUC-KR"
    pageEncoding="EUC-KR"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<%@page import="javax.resource.cci.Connection"%>
<%@page import="javax.naming.InitialContext"%>
<%@page import="javax.resource.cci.ConnectionFactory"%>
<%@page import="javax.resource.cci.RecordFactory"%>
<%@page import="javax.resource.cci.MappedRecord"%>
<%@page import="javax.resource.cci.Interaction"%>
<%@page import="javax.resource.cci.InteractionSpec"%>
<%@page import="com.tmax.connector.cci.TmaxInteractionSpecImpl"%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
<title>Insert title here</title>

```

```

</head>
<body>
<%
    InitialContext ctx = new InitialContext();
    ConnectionFactory connFactory = (ConnectionFactory)ctx.lookup("TmaxConnector");
    Connection conn = connFactory.getConnection();

    Interaction action = conn.createInteraction();
    TmaxInteractionSpecImpl aspect = new TmaxInteractionSpecImpl();
    aspect.setSvcName("TOUPPER");
    aspect.setAction(InteractionSpec.SYNC_SEND_RECEIVE);

    RecordFactory recordFactory = connFactory.getRecordFactory();
    MappedRecord sndBuf = recordFactory.createMappedRecord(null);
    sndBuf.put("$String", "abc");
    MappedRecord rcvBuf = recordFactory.createMappedRecord(null);

    if( action.execute(aspect, sndBuf, rcvBuf) )
    {
        String result = (String)rcvBuf.get("$String");
    }
    conn.close();
    System.out.println("end lookup");
%>
</body>
</html>

```

4.1.3. インフローの例

JCAのインフローはJCAでTmaxからのサービス要求がある場合、登録されているメッセージ・ドリブン・ビーンを呼び出す構造です。メッセージ・ドリブン・ビーンを実装およびデプロイして使用します。

TmaxListenerBean.java

以下は、WebTJCAでサービス要求を受信し、メッセージ・ドリブン・ビーンのOnMessageを呼び出し、サービス名にしたがって他のEJBを呼び出す構造の例です。

```

package com.tmax.sample.echoMBean;

import java.rmi.RemoteException;
import javax.ejb.CreateException;
import javax.ejb.EJBException;
import javax.ejb.MessageDrivenBean;
import javax.ejb.MessageDrivenContext;

```

```

import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.resource.ResourceException;
import javax.resource.cci.MappedRecord;
import javax.resource.cci.MessageListener;
import javax.resource.cci.Record;
import javax.rmi.PortableRemoteObject;
import com.tmax.sample.TmaxBLSample;
import com.tmax.sample.TmaxBLSampleHome;

public class TmaxListenerBean implements MessageListener, MessageDrivenBean {
    /**
     *
     */
    private static final long serialVersionUID = 7971664840039920708L;
    public TmaxListenerBean() {
    }

    public void ejbCreate() throws EJBException {
    }

    public void ejbRemove() throws EJBException {
        // TODO Auto-generated method stub
    }

    public void setMessageDrivenContext(MessageDrivenContext ctx)
        throws EJBException {
    }

    public Record onMessage(Record in) throws ResourceException {
        System.out.println(in);
        String svcName = (String)((MappedRecord)in).get("$ServiceName");

        if( svcName.equalsIgnoreCase("echostring") )
        {
            return in;
        }
        else if( svcName.equalsIgnoreCase("create_account_b"))
        {
            System.out.println(((MappedRecord)in).get("$Message"));
            Context initial;
            try {
                System.setProperty("java.naming.factory.initial",
                                    "jeus.jndi.JNSContextFactory");
                System.setProperty("java.naming.factory.url.pkgs", "jeus.jndi.jns.url");
            }
            catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
}

```

```

        System.setProperty("java.naming.provider.url", "192.168.11.253");

        initial = new InitialContext();

        Object object = initial.lookup("TmaxBLSample");    // return EJB Home

        if( object == null )
        {
            return createFailRecord(in, -6);
        }

        TmaxBLSampleHome home = (TmaxBLSampleHome)
        PortableRemoteObject.narrow(object, TmaxBLSampleHome.class);
        TmaxBLSample sample = home.create();
        if( sample.createAccount(in) )
        {
            MappedRecord out = (MappedRecord) in.clone();
            out.put("$ReturnCode", new Integer(0));
            out.put("RESULT", "OK");
            return out;
        }
        else
        {
            return createFailRecord(in, -1);
        }
    } catch (NamingException e) {
        e.printStackTrace();
        return createFailRecord(in, -2);
    } catch (RemoteException e) {
        e.printStackTrace();
        return createFailRecord(in, -3);
    } catch (CreateException e) {
        e.printStackTrace();
        return createFailRecord(in, -4);
    } catch (CloneNotSupportedException e) {
        e.printStackTrace();
        return createFailRecord(in, -5);
    }
}

else if( svcName.equalsIgnoreCase("transfer_money_b"))
{
    System.out.println(((MappedRecord)in).get("$Message"));

    Context initial;
    try {
        System.setProperty("java.naming.factory.initial",
            "jeus.jndi.JNSContextFactory");
    }
}

```

```

        System.setProperty("java.naming.factory.url.pkgs", "jeus.jndi.jns.url");

        System.setProperty("java.naming.provider.url", "192.168.11.253");

        initial = new InitialContext();

        Object object = initial.lookup("TmaxBLSample");    // return EJB Home

        if( object == null )
        {
            return createFailRecord(in, -6);
        }

        TmaxBLSampleHome home = (TmaxBLSampleHome)
        PortableRemoteObject.narrow(object, TmaxBLSampleHome.class);
        TmaxBLSample sample = home.create();
        if( sample.transferMoney(in) )
        {
            MappedRecord out = (MappedRecord) in.clone();
            out.put("$ReturnCode", new Integer(0));
            out.put("RESULT", "OK");
            return out;
        }
        else
        {
            return createFailRecord(in, -1);
        }
    } catch (NamingException e) {
        e.printStackTrace();
        return createFailRecord(in, -2);
    } catch (RemoteException e) {
        e.printStackTrace();
        return createFailRecord(in, -3);
    } catch (CreateException e) {
        e.printStackTrace();
        return createFailRecord(in, -4);
    } catch (CloneNotSupportedException e) {
        e.printStackTrace();
        return createFailRecord(in, -5);
    }
}

return in;
}

public Record createFailRecord(Record src, int usrcode)
{
    MappedRecord out = null;

```

```

    try {
        out = (MappedRecord) src.clone();
    }
    catch (CloneNotSupportedException e) {
        e.printStackTrace();
        return null;
    }
    out.put("$ReturnCode", new Integer(-1));
    out.put("$UserCode", new Integer(usrcode));
    out.put("RESULT", "FAIL");
    return out;
}
}

```

ejb-jar.xml

EJB 2.0ファイルのためのデプロイメント・ディスクリプタとして、以下のようなオプション情報が含まれています。

```

<?xml version="1.0" encoding="UTF-8"?>
<ejb-jar id="ejb-jar_ID" version="2.1" xmlns="http://java.sun.com/xml/ns/j2ee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/ejb-jar_2_1.xsd">
  <display-name>TmaxListenerBean</display-name>
  <enterprise-beans>
    <message-driven>
      <display-name>TmaxListenerBean</display-name>
      <ejb-name>TmaxListenerBean</ejb-name>
      <ejb-class>com.tmax.sample.echoMBean.TmaxListenerBean</ejb-class>
      <transaction-type>Container</transaction-type>
      <message-destination-link>TmaxRA</message-destination-link>
      <activation-config>
        <activation-config-property>
          <activation-config-property-name>serviceName</activation-config-property-name>

          <activation-config-property-value>
            ECHOSTRING,CREATE_ACCOUNT_B,TRANSFER_MONEY_B
          </activation-config-property-value>
        </activation-config-property>
      </activation-config>
    </message-driven>
  </enterprise-beans>
</ejb-jar>

```


jeus-ejb-dd.xml

JEUSのEJB 2.0ファイルのためのデプロイメント・ディスクリプタです。

```
<?xml version="1.0"?>
<jeus-ejb-dd xmlns="http://www.tmaxsoft.com/xml/ns/jeus">
  <beanlist>
    <jeus-bean>
      <ejb-name>TmaxListenerBean</ejb-name>
      <mdb-resource-adapter-name>TmaxRA</mdb-resource-adapter-name>
    </jeus-bean>
  </beanlist>
</jeus-ejb-dd>
```

参考

jeus-ejb-dd.xmlの詳細内容については、『JEUS EJBガイド』の「5.3.1.4. jeus-ejb-dd.xmlのデプロイメント・ディスクリプタの作成」を参照してください。

4.2. WebLogicでのWebTJCAの使用

WebLogicでWebTJCAを使用する方法について説明します。

4.2.1. リソース・アダプターの構成

JCAを利用するには、その前提としてリソース・アダプターの構成が必要です。リソース・アダプターにはWebアプリケーション・サーバーを提供するベンダーの固有の設定と、JCAで定義した設定を定義する必要があります。リソース・アダプターはJava Archive(JAR)の形式でRARでパッケージングします。

下記は、JCAで定義したリソース・アダプターに含まれるリストです。

Contents of RAR file	Requirements	Relative location within RAR file
Deployment Descriptor	Required	META-INF/ra.xml
howto.html, image files, etc.	Optional	Arbitrary (that is, could be at root level or sub-level)
JAR files	Optional	Arbitrary
Platform-specific native libraries	Optional	Arbitrary

開発者は、ra.xmlとjeus-connector-dd.xmlを記述します。

ra.xmlの設定

インフローとアウトバウンドを一緒に設定します。ra.xml設定の詳細については、「[付録 A. WebTJCA ra.xml の設定](#)」を参照し、設定方法については、「[4.1.1. リソース・アダプターの構成](#)」の「ra.xmlの設定」例を参照してください。

weblogic-ra.xmlの設定

以下は、weblogic-ra.xmlを設定する例です。例題にてのjndi-nameはアウトバウンド・コネクションが参照可能なJNDIの名前です。

```
<?xml version="1.0" ?>
<weblogic-connector xmlns="http://www.bea.com/ns/weblogic/90">
  <jndi-name>TmaxRA</jndi-name>
  <enable-access-outside-app>true</enable-access-outside-app>
  <outbound-resource-adapter>
    <connection-definition-group>
      <connection-factory-interface>javax.resource.cci.ConnectionFactory
      </connection-factory-interface>
    <connection-instance>
      <jndi-name>TmaxConnector</jndi-name>
      <connection-properties>
        <pool-params>
          <initial-capacity>2</initial-capacity>
          <max-capacity>10</max-capacity>
          <capacity-increment>1</capacity-increment>
          <shrinking-enabled>true</shrinking-enabled>
          <shrink-frequency-seconds>60</shrink-frequency-seconds>
        </pool-params>
      </connection-properties>
    </connection-instance>
  </connection-definition-group>
</outbound-resource-adapter>
</weblogic-connector>
```

デプロイ

- フォルダの構成

```
webt5.9.7.0.jar
META-INF/ra.xml
META-INF/weblogic-ra.xml
```

- RARファイルの作成

```
jarでTmaxRA.rarを作成
```

- JCAでWebLogicにデプロイ

```
WebLogicのAdministratorで「アプリケーションのデプロイ」を利用してデプロイ
```

参考

weblogic-ra.xml設定の詳細については、『BEA WebLogic Serverガイド』の「Programming WebLogic Resource Adapters」を参照してください。デプロイ方法の詳細については、『BEA WebLogic Serverガイド』の「Deploying Applications to WebLogic Server」を参照してください。

4.2.2. アウトバウンドの例

TmaxのTOUPPERサービスを呼び出すjspのサンプル・ファイルです。アウトバウンドの例については、[「4.1.2. アウトバウンドの例」](#)のOutboundの設定例を参照してください。

4.2.3. インフローの例

JCAのインフローはJCAでTmaxからのサービス要求がある場合、登録されているメッセージ・ドリブン・ビーンを呼び出す構造です。メッセージ・ドリブン・ビーンを実装およびデプロイして使用してください。

TmaxListenerBean.java

WebTJCAでサービス要求を受信し、メッセージ・ドリブン・ビーンのOnMessageを呼び出し、サービス名にしたがって他のEJBを呼び出す構造です。[「4.1.3. インフローの例」](#)のTmaxListenerBean.javaの例を参照してください。

ejb-jar.xml

EJB 2.0ファイルのためのデプロイメント・ディスクリプタとして、以下のようなオプション情報が含まれています。

全EJBの構造情報を格納している<enterprise-beans>、アプリケーション・デプロイメントのためのオプション情報を格納している<assembly-descriptor>で構成されており、該当するビーンのトランザクション処理方法やセキュリティ・ポリシーなどを設定します。特に例の設定は、MDBで構成されているTmaxListenerBeanをデプロイするためのものです。

```
<?xml version="1.0" encoding="UTF-8"?>
<ejb-jar id="ejb-jar_ID" version="2.1"
```

```

xmlns="http://java.sun.com/xml/ns/j2ee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/ejb-jar_2_1.xsd">
  <display-name>TmaxListenerBean</display-name>
  <enterprise-beans>
    <message-driven>
      <display-name>TmaxListenerBean</display-name>
      <ejb-name>TmaxListenerBean</ejb-name>
      <ejb-class>com.tmax.sample.echoMBean.TmaxListenerBean</ejb-class>
      <messaging-type>javax.resource.cci.MessageListener</messaging-type>
      <transaction-type>Container</transaction-type>
      <message-destination-link>TmaxRA</message-destination-link>
      <activation-config>
        <activation-config-property>
          <activation-config-property-name>serviceName
        </activation-config-property-name>
          <activation-config-property-value>
            ECHOSTRING,CREATE_ACCOUNT_B,TRANSFER_MONEY_B
          </activation-config-property-value>
        </activation-config-property>
      </activation-config>
    </message-driven>
  </enterprise-beans>
  <assembly-descriptor>
    <message-destination>
      <message-destination-name>TmaxRA</message-destination-name>
    </message-destination>
  </assembly-descriptor>
</ejb-jar>

```

JEUSと違って、WebLogicでは以下のように設定します。WebLogicではメッセージ・リンクとアセンブリー・ディスクリプターを設定してはなりません。

```

<!-- message--link>TmaxRA</message-destination-link -->
<!-- assembly-descriptor>
  <message-destination>
    <message-destination-name>TmaxRA</message-destination-name>
  </message-destination>
</assembly-descriptor -->

```

weblogic-ejb-jar.xml

WebLogicに依存的な属性に関する部分を定義するファイルで、キャッシング、クラスタリング、オブティマイジング関連の内容が含まれます。

```
<?xml version="1.0" ?>
<weblogic-connector xmlns="http://www.bea.com/ns/weblogic/90">
  <jndi-name>TmaxRA</jndi-name>
<?xml version="1.0" encoding="UTF-8"?>
<weblogic-ejb-jar
xmlns="http://www.bea.com/ns/weblogic/90"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.bea.com/ns/weblogic/90
http://www.bea.com/ns/weblogic/90/weblogic-ejb-jar.xsd">
  <weblogic-enterprise-bean>
    <ejb-name>TmaxListenerBean</ejb-name>
    <message-driven-descriptor>
      <resource-adapter-jndi-name>TmaxRA</resource-adapter-jndi-name>
    </message-driven-descriptor>
  </weblogic-enterprise-bean>
</weblogic-ejb-jar>
```


付録 A. WebTJCA ra.xmlの設定

本章では、WebTJCA ra.xmlの設定時に使用するフィールド情報について説明します。

A.1. フィールド情報

フィールド情報は、http://java.sun.com/xml/ns/j2ee/connector_1_5.xsdに従っています。

A.1.1. <resourceadapter-class>

「com.tmax.connector.spi.TmaxResourceAdapterImpl」と設定します。

<config-property>

以下は、<connector><resourceadapter><config-property>に格納可能な情報で、インフローの設定を定義します。

名前	タイプ	説明
port	String	インフロー・アダプターをリッスンするポート番号を登録します
fdlfile	String	使用するFDLファイルのパスを指定します
logLevel	String	出力するログレベルです。none、info、debugが使用できます
logDir	String	出力するログファイルを保存するパスです。設定しない場合、stdioで出力します
logFile	String	出力するログファイルの名前です。設定しない場合、stdioで出力します
logValidDays	Integer	保存するログファイルの作成周期です。1の場合、日次でログファイルを作成します
logFileFormat	String	保存するログファイルの名前の形式です (例: MMddyyyy)

<outbound-resourceadapter><connection-definition>

- <managedconnectionfactory-class>

「com.tmax.connector.spi.TmaxManagedConnectionFactoryImpl」と定義します。

- <config-property>

以下は、<config-property>に格納可能な情報で、アウトバウンドの設定を定義します。

名前	タイプ	説明
host	String	接続するメインTmaxのIPアドレスです (例: 192.168.1.43)
host_port	Integer	接続するメインTmaxのポート番号です
backup	String	接続するバックアップTmaxのIPアドレスです
backup_port	Integer	接続するバックアップTmaxのポート番号です
support_xa	Boolean	XAの対応有無です(デフォルト値: false) (例: true or false)
headerType	String	接続するTmaxのバージョンです(デフォルト値 : 3.x以下) (例: 4.0以上 : extendedV4)
connectTimeout	Integer	Tmaxへの接続を試みる時間です(単位 : 秒)
encryption	Boolean	暗号化の有無です(デフォルト値 : false) (例: true or false)
userName	String	ユーザー名です
userPasswd	String	ユーザー・パスワードです
domainName	String	ドメイン名です
domainPasswd	String	ドメイン・パスワードです
tpTimeout	Integer	Tmaxにサービスを要求する際に待機する時間です(単位 : 秒)
txTimeout	Integer	トランザクション全体を待機する時間です(単位 : 秒)
tpTimeout	Integer	Tmaxにサービスを要求する際に待機する時間です(単位 : 秒)
fdlFile	String	使用するFDLファイルのパスです。例) /data/tmax/fdl/tmax.fdl
xidMapperPath	String	XAを使用するとき、Webアプリケーション・サーバーのxidとTmaxで使用するxidのマッピングを記録するxid mappingファイルを保存するパスです (例: /data/was/log/webt/xid/)
xidMapperFile	String	XAを使用するとき、Webアプリケーション・サーバーのxidとTmaxで使用するxidのマッピングを記録するxid mappingファイルを保存するファイル名です (例: xidmapper.log)

名前	タイプ	説明
logLevel	String	出力するログレベルで、none、info、debugが使用可能です
logDir	String	出力するログファイルを保存するパスで、設定しない場合はstdioで出力します
logFile	String	出力するログファイルの名前で、設定しない場合はstdioで出力します
logValidDays	Integer	保存するログファイルの作成周期です (例: 1の場合、日次でログファイルを作成します)
logFileFormat	String	保存するログファイル名の形式です (例: MMddyyyy)
connectionName	String	コネクション名です

- <connectionfactory-interface>

「javax.resource.cci.ConnectionFactory」と定義します。

- <connectionfactory-impl-class>

「com.tmax.connector.cci.TmaxConnectionFactoryImpl」と定義します。

- <connection-interface>

「javax.resource.cci.Connection」と定義します。

- <connection-impl-class>

「com.tmax.connector.cci.TmaxConnectionImpl」と定義します。

索引

C

CCI, 5
ConnectionFactory, 6

E

EIS, 1
ejb-jar.xml, 12

G

getRecordFactory(), 7

J

javax.resource.cci.MessageListener, 13
 public Record onMessage(Record in) throws
 ResourceException, 13
JCA, 1
JEUS 6でのWebTJCAの使用設定
 jeus-connector-dd.xml, 18
 ra.xml, 15
 デプロイ, 19

P

public Record onMessage(Record in) throws
ResourceException, 13

R

ra.xml, 11, 14

T

TmaxInteractionSpecImpl, 7

W

WebLogicでのWebTJCAの使用設定
 ra.xml, 26
 weblogic-ra.xml, 26

インフローの例, 20, 27

デプロイ, 26

WebtDialogueService(), 10

WebTJCA, 1

WebtRemoteService(), 9

X

Xアトランザクション, 8

あ

アウトバウンド通信, 2

インバウンド通信, 3

インバウンド通信の処理プロセス, 11

インバウンド通信の構造, 12

インバウンド通信の設定

 ejb-jar.xml, 12

 ra.xml, 11

インフローの例

 ejb-jar.xml, 24, 27

 jeus-ejb-dd.xml, 25

 TmaxListenerBean.java, 20, 27

 weblogic-ejb-jar.xml, 28

た

対話型通信のサポート, 10

 WebtDialogueService(), 10

は

非同期要求のサポート, 9

 WebtRemoteService(), 9

ま

メッセージ・ドリブン・ビーン

 javax.resource.cci.MessageListener, 13

メッセージ・ドリブン・ビーンの設定, 13

メッセージ・ドリブン・ビーンの設定例, 13

ら

ロギングの設定例, 14

 ra.xml, 14

ローカル・トランザクションのサポート, 9

