

Tmax スタートガイド

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 スタートガイド

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 TPモニターの紹介	1
1.1. 概要	1
1.2. ミドルウェア	1
1.3. TPモニター	4
第2章 Tmaxの紹介	5
2.1. 概要	5
2.2. Tmaxの構造	5
2.2.1. システム構成	6
2.2.2. TIM	8
2.2.3. ソケット通信	9
2.3. Tmax機能	10
2.3.1. プロセス管理	13
2.3.2. 分散トランザクション	15
2.3.3. 負荷調整(ロードバランシング)	18
2.3.4. 障害対策(フェイルオーバー)	21
2.3.5. ネーミング・サービス	24
2.3.6. プロセス制御	25
2.3.7. RQ機能	26
2.3.8. セキュリティー機能	26
2.3.9. システムのリソース管理	26
2.3.10. マルチドメインおよび多様なゲートウェイ・サービスの提供	27
2.3.11. 多様なクライアント・エージェントの提供	29
2.3.12. 多様な通信方式のサポート	30
2.3.13. 多様な開発方式のサポート	33
2.3.14. メッセージの安定的な伝達	35
2.4. Tmaxの特徴	37
2.5. Tmax導入時の考慮事項	39
2.5.1. システム環境	39
2.5.2. 考慮事項	40
第3章 WebTの紹介	43
3.1. 概要	43
3.2. WebTConnectionPool	43
3.3. WebT-Serverシステム	44
第4章 Tmaxアプリケーション	47
4.1. アプリケーションの構成	47
4.2. バッファ・タイプ	48
4.3. クライアント/サーバー・プログラム	49
4.3.1. クライアント・プログラム	49

4.3.2. サーバー・プログラム	53
4.4. システム環境ファイル	58
4.5. API	59
4.5.1. Tmaxの標準API	59
4.5.2. 非標準API	61
4.6. エラー・メッセージ	67
4.6.1. X/Open DTP関連エラー	67
4.6.2. FDL関連エラー	69
第5章 例	71
5.1. 通信タイプの例	71
5.1.1. 同期型通信	71
5.1.2. 非同期型通信	74
5.1.3. 会話型通信	78
5.2. グローバル・トランザクション・プログラムの例	84
5.3. データベース・プログラム	92
5.3.1. Oracle Insertプログラム	93
5.3.2. Oracle Selectプログラム	97
5.3.3. Informix Insertプログラム	104
5.3.4. Informix Selectプログラム	113
5.3.5. DB2プログラム	121
5.4. データベース連携プログラム	130
5.4.1. 同期型モード(同機種)	130
5.4.2. 同期型モード(異機種)	138
5.4.3. 非同期型モード(同機種)	141
5.4.4. 会話型モード(同機種)	147
5.5. TIPを利用したプログラム	155
5.5.1. TIPの構造	155
5.5.2. TIPの使用	158
5.5.3. TIPの使用例	161
5.5.4. システム環境情報照会プログラム	171
5.5.5. システム統計情報照会プログラム	174
5.5.6. サーバー・プロセスの起動および終了プログラム	178
5.6. 再帰呼び出し(Local recursive call)	184
第6章 ガイドの構成	189
6.1. 概要	189
6.2. ガイドの構成および内容	189
用語集	201
索引	205

図目次

[図 1.1]	ミドルウェアの動作原理	3
[図 2.1]	システム構成	6
[図 2.2]	Tmaxシステムのサービスの実行	8
[図 2.3]	ドメイン・ソケット通信	9
[図 2.4]	2ティアのクライアント/サーバー構造	13
[図 2.5]	3ティアのクライアント/サーバー構造	14
[図 2.6]	X/Open DTPの構造	16
[図 2.7]	2フェーズコミットの動作	17
[図 2.8]	SLMによる負荷調整	19
[図 2.9]	DDRによる負荷調整	20
[図 2.10]	DLMIによる負荷調整	21
[図 2.11]	ハードウェア障害	21
[図 2.12]	負荷調整による障害対策	22
[図 2.13]	サービスバックアップによる障害対策-障害発生前	23
[図 2.14]	サービス・バックアップによる障害対策-障害発生後	23
[図 2.15]	ソフトウェア障害	24
[図 2.16]	ネーミング・サービス	24
[図 2.17]	プロセスタイプ	25
[図 2.18]	RQプロセス	26
[図 2.19]	マルチドメインのサービスフロー	28
[図 2.20]	Tmaxゲートウェイの動作	29
[図 2.21]	RCAの構造	29
[図 2.22]	CAを利用したサービス呼び出しの形態	30
[図 2.23]	同期通信	31
[図 2.24]	非同期通信	31
[図 2.25]	対話型通信	32
[図 2.26]	伝達型-タイプA	32
[図 2.27]	伝達型-タイプB	33
[図 2.28]	クライアントとRDPの直接的なデータ送信	33
[図 2.29]	WinTmaxライブラリーの構造および機能	34
[図 2.30]	HMSの高い可用性	35
[図 2.31]	JMSのAPI互換性	36
[図 2.32]	HMS-キュー方式	36
[図 2.33]	HMS-Topic方式	37
[図 3.1]	WebTとTmax間のサービス・フロー	43
[図 3.2]	WebT-Serverシステム	44
[図 4.1]	Tmaxアプリケーションの構成	47
[図 4.2]	Tmaxの通信バッファのタイプ	48
[図 4.3]	FDLの保存方式	49
[図 4.4]	クライアント・プログラムのフロー	50

[図 4.5]	クライアント・プログラム関数プロセス	52
[図 4.6]	Tmaxクライアント・プログラムの構成	52
[図 4.7]	サーバー・プログラムのフロー	54
[図 4.8]	tpcallとtpforwardの比較	56
[図 4.9]	Tmaxサーバー・プログラムの構成	56
[図 5.1]	2つのデータベースの接続	85
[図 5.2]	同期型モードのフロー図(同機種のデータベース接続)	131
[図 5.3]	同期型モードのフロー図(異機種のデータベース接続)	138
[図 5.4]	非同期型モードのフロー図(同機種のデータベース接続)	141
[図 5.5]	会話型モードのフロー図(同機種のデータベース接続)	147

このガイドについて

対象読者

本書は、Tmax[®](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おり、Tmaxシステムが提供する代表的な機能について説明しています。同書は、開発に先立ってシステムの構成や各機能を習得するために活用できます。実際の開発時には、Tmaxが提供する各種プログラミング・ガイドを参照してください。

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6章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: TPモニターの紹介

ミドルウェアとTPモニターの概念と機能について説明します。

- 第2章: Tmaxの紹介

Tmaxの概念、構造、機能、特徴および導入時の考慮事項について説明します。

- 第3章: WebTの紹介

TmaxとJavaアプリケーション・プログラム間のトランザクション・サービスをサポートするために提供されるWebTの基本機能と役割について説明します。

- 第4章: Tmaxアプリケーション

Tmaxアプリケーションを開発するために必要な基本概念とクライアント/サーバー・プログラムのフローと構成、プログラム開発のためのAPとエラーについて説明します。

- 第5章: 例

様々な環境で開発されたTmaxアプリケーションのプログラムの例について説明します。

- 第6章: ガイドの構成

Tmaxの全ガイドの一覧と各ガイドの内容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
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、コマンド実行後の画面に出力された結果物、サンプル・コード
{ }	必須パラメータ値
[]	オプション・パラメータ値
	選択パラメータ値

システム要件

	要求事項
プラットフォーム	IBM AIX 5.x / 6.1 / 7.1
	HP-UX 11.xx
	SunOS 5.7~5.9 / SunOS 5.10 / SunOS 5.11
ハードウェア	1GB以上のハードディスク空き容量
	512MB以上のメモリー空き容量
データベース	Oracle 9~12
	Tibero 4~5
	DB2
	Informix

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 TPモニターの紹介

本章では、ミドルウェアとTPモニターの概念と機能について説明します。

1.1. 概要

TPモニター(Transaction Processing Monitor)は、各種のプロトコルで動作するセッションと、システムおよびデータベース間における最初の処理単位であるトランザクションを監視し、それらを一貫性のある形で保存および維持する役割をする、トランザクション管理ミドルウェアです。

TmaxはTPモニター基盤の製品です。本章ではTmaxへの理解を助けるべく、ミドルウェアとTPモニターについて説明します。

1.2. ミドルウェア

中央集中型のメインフレーム環境が運用および費用面などで多様な問題をはらんでいる中、そのメインフレーム環境を業務別にホストを分離する開放型の分散システム環境に移行させる、ダウンサイズ(DOWNSIZING)方法論が採用され始めています。

一方、開放型の分散環境は、メインフレームで使用していた業務を複数のサーバーに分散させて管理しているため、相互異なる運用体制間の通信や各サーバー・プログラム間の連動、互換性などの面で問題が発生しています。これを受けて、相互異なるシステムやサーバー・プログラム、ネットワーク・リソースを1つの単一(Single)ユーザー環境で使用したいとのニーズが生まれています。

以下は、中央集中型のメインフレーム環境と開放型の分散システム環境の問題について説明した表です。

● 中央集中型のメインフレーム環境の問題点

区分	問題点
費用面	– 導入コストの高さ – メンテナンス・コストの過剰支出
運用面	– 業務プロセスよりメインフレーム環境が最優先されるため、ユーザー環境が考慮されがたい
システム面	– 異機種間の通信の難しさ – 他のシステムへのプログラム移植が困難 – 業務拡張の難しさ

● 開放型の分散システム環境の問題点

区分	問題点
費用面	<ul style="list-style-type: none"> – ネットワーク、DBMSなど、専門技術が求められる
運用面	<ul style="list-style-type: none"> – 非効率なシステム運用上の問題 – ユーザー環境が考慮されていない – 分散環境による管理の問題 – システムの管理および監督の厳しさ – 障害が発生したときの対処が困難 – 他サーバー間に運用方法が相違 – 複数の供給者の存在による問題 – ユーザーの増加による急激な性能の低下 – サーバー間の負荷の違い
システム面	<ul style="list-style-type: none"> – プロセス管理 – 多様な通信方式が必要 – システムのセキュリティ機能が脆弱 – 異機種のデータベースおよびグローバル・トランザクションの処理が困難 – 異機種サーバー間のプログラム移植の問題 – 開発上の問題(OS、開発言語などがあまりにも多様) – 1種類のミドルウェアにて異質な分散環境においてトランザクション処理およびプロセス管理の機能を提供

こうした開放型の分散環境が持つ問題を解決するために開発されたソフトウェアがミドルウェアです。

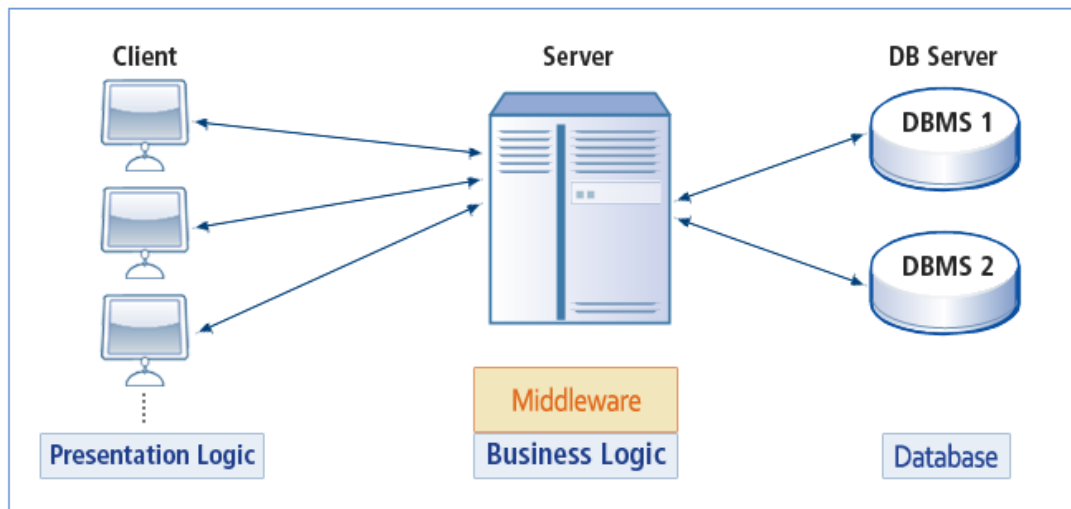
ミドルウェアは分散コンピューティングにおいて単一のユーザー環境を提供し、異機種システム間のネットワーク連携、クライアントとサーバー間の通信またはコンピューターとコンピューターの連携を担当するシステム・ソフトウェアです。ミドルウェアは異機種のハードウェアやプロトコル、通信環境などを1つにつなげ、アプリケーションや運用環境の間で円滑な通信ができるようにします。

クライアントとデータベース・サーバーは直接通信を行わず、ミドルウェアが2つのシステム間の通信を連携します。ミドルウェアが業務に必要なビジネスロジックを持っているので、クライアントはミドルウェアとの通信のみを考慮し、サーバー・プログラムもミドルウェアとの通信のみを考慮すればいいです。クライアントとデータベース・サーバーはミドルウェアを介して1つの単一システム環境を構築することができます。異機種マシ

ンで使用するデータベースが多数存在する環境でもシステム統合を保証でき、データの互換性および整合性を保証できます。こうした環境では最小のリソースで最大の性能が出せます。

以下は、ミドルウェアを使用してクライアント/サーバー環境を構築したシステム構成図です。

[図 1.1] ミドルウェアの動作原理



ミドルウェア製品は、各製品の用途と目的によって、以下の6つに区分できます。

- TPモニター(Transaction Processing Monitor)

異質の分散環境でトランザクションを処理し、各種の処理プロセスを管理する機能を提供するサービスです。Tmaxはミドルウェアの多様な製品群のうちTPモニター製品群に属します。

- WAS(Web Application Server)

Webでトランザクションを処理し、異機種間の相互通信機能(J2EE)を提供するサービスです。

- MOM(Messaging Oriented Middleware)

メッセージをキューに積載して処理し、キューによるメッセージ管理機能を提供(非同期)するサービスです。

- Database Access System

分散環境で複数のデータベース・サーバーを一貫した方法で利用できる環境を提供するサービスです。

- RPC System

ネットワークで他のコンピューターにあるプログラムを実行(同期)するサービスです。

- ORB(Object Request Broker)

クライアント・オブジェクトがORBというソフトウェア・バスを利用してリモートのサーバーのメソッドを呼び出す機能を提供するサービスです。

1.3. TPモニター

初期のほとんどの業務システムはメインフレーム基盤の中央集中型の環境で構成されていました。そうした中、中央集中型の環境は費用や管理に数多くのデメリットを現し始めました。一方、中央集中型の環境の問題点を補完すべく浮上したのが開放型の分散システムです。ところが、開放型の分散システムもシステム運用や管理において問題をはらんでおり、こうした問題を解決するために導入されたソフトウェアがミドルウェアです。ミドルウェアのうち、TPモニターはプロトコルで動作するセッションとシステムおよびデータベース間の最小の処理単位であるトランザクションを監視し、一貫性ある形で保存および維持する役割をするトランザクション管理ミドルウェアです。

以下は、TPツモニターの主要機能です。

- 容易なアプリケーションの開発

複雑な業務プロセスを開発するとき、データ中心から機能中心に分散して開発します。既存メインフレーム環境はビジネス・ロジックとデータ処理ロジックが混在する形で開発されました。そのため、メインフレーム環境でアプリケーションの開発は相当レベルの難度のある作業でした。

ところが、TPモニターを使用すると、ミドルウェアにはビジネス・ロジックを実装し、クライアント・プログラムはユーザー提供するモジュールのみを実装すればいいです。こうした機能の分離により、アプリケーションの開発が簡単になります。

- 効率的なアプリケーションの管理

分散されている各業務システムをTPモニターを使って管理するので、各アプリケーションを効率的に管理できます。

- 異機種DBMSのリソースの管理

DBMSのトランザクションを統合管理するので、異機種データベース間のリソース管理が可能です。

- 負荷分散(ロードバランシング)

最適のリソースの使用を管理するので、負荷を分散して分散トランザクションをサポートします。

- 実行スピードと信頼性の確保

少ないサーバー・リソースで多くのクライアントを管理し、DBMSのオーバーヘッドを減らし、回答時間を向上させます。

第2章 Tmaxの紹介

本章では、Tmaxの概念、構造、機能、特長、導入時の考慮事項について記述します。

2.1. 概要

Tmaxは、Transaction Maximizationの略語で、トランザクション処理の最大化を意味します。Tmaxはシステムの分散環境で異機種コンピューター間のトランザクション処理を完璧に保証しつつ負荷を分散させるほか、エラーが発生したら適切な措置を取る役割をするTPモニターです。

トランザクションの特性をサポートしつつ、ユーザーには最適の開発環境を、そしてクライアント/サーバー環境には最適のソリューションを提供し、性能改善はもちろんすべての障害に完璧に対応します。

Tmaxは分散トランザクション・プロセッシングの国際標準であるX/Open DTP(Distributed Transaction Processing)モデルを遵守し、国際標準化機構のOSI(Open Systems Interconnection group)のDTPサービスに対する機能的な分散と機能構成要素間のAPIおよびシステム・インターフェース定義にしたがって開発されました。また、分散環境で異機種間の透明な業務処理およびOLTP(On-Line Transaction Processing)をサポートし、トランザクション処理のACID(Atomic、Consistent、Isolated、Durable: Transaction Properties)の特性を満たせます。

Tmaxは透明な業務処理によって性能を最大化して重要度の高い基幹業務を完璧に処理し、開発者とシステム管理者に対して最適化されたコンピューティング環境を提供します。また、金融、公共、通信、製造、流通など全産業分野で1日数千万件のトランザクションが発生するミッション・クリティカルなシステムで負荷を分散させ、障害の発生を防止し、顧客システムの安定性を保証します。そのため、大規模(Large-scale)のOLTPアプリケーションの開発や航空、ホテル、病院、安保分野のほか、銀行のオンライン分野、クレジットカードの承認業務、顧客および販売管理業務のような多様な事業分野と業務で使われています。

参考

トランザクションの詳細については、『Tmax アプリケーションガイド』の「第6章 トランザクション」を参照してください。

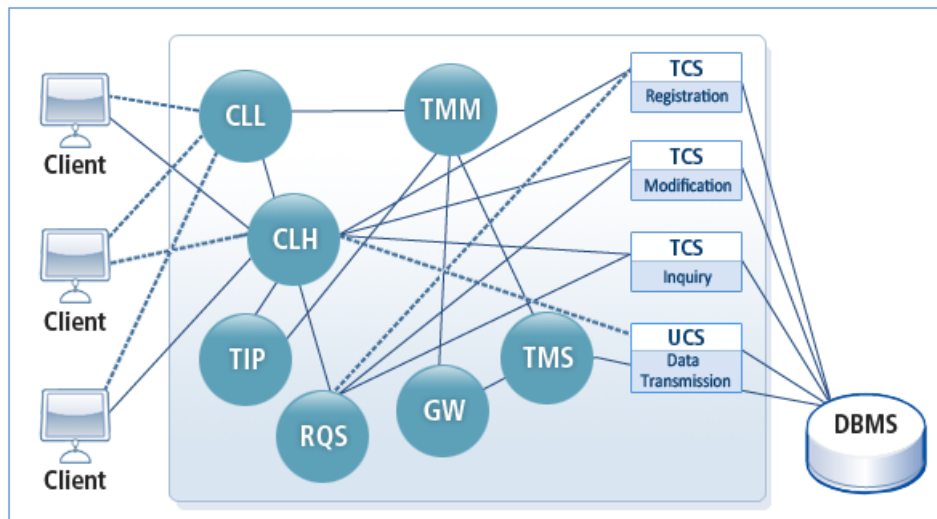
2.2. Tmaxの構造

本節では、システムの構成と機能について説明します。

2.2.1. システム構成

以下は、Tmaxのシステム構成を示した図です。

[図 2.1] システム構成



- TMM (Tmax Manager)

Tmaxシステムを運用管理する中核プロセスで、Tmaxシステムのすべての共有情報とCLL(Client Listener)、CLH(Client Handler)、TMS(Transaction Management Server)およびAP(Application Program)サーバープロセスを管理します。

以下は、TMMの主要機能についての説明です。

主要機能	説明
共有メモリーの割当	設定した環境情報をcflコマンドでコンパイルし、バイナリはエンジンが起動される時点でTMMプロセスによって共有メモリー(Shared Memory)にローディングされます。TMMはローディングされた運用情報を利用してシステムを運用します
プロセス管理	TMMはすべてのシステムに対する運用と終了の主体になります
ログ管理	Tmaxで発生するシステム・ログ(slog)とユーザー・ログ(ulog)を管理します

- CLL (Client Listener)

クライアントとTmaxの連結を担当するプロセスで、クライアント接続管理のためにポート・リスナー(PORT Listener)を設定してクライアントからの要求を受けます。

- CLH (Client Handler)

クライアント・ハンドラーです。クライアントとサーバー間のインターフェースを実行し、サービスを提供する業務処理サーバーにサービスを要求し、サーバーに対する連結および管理を行います。

tpcallのような関数を介して受信されるクライアントの要求を当該サーバーに送信します。また、XAサービス環境でXID採番とコミット/ロールバック(commit/rollback)要求をインターフェースします。

- TMS (Transaction Management Server)

データベース管理および分散トランザクション処理を担当するプロセスで、データベース関連のシステムで動作します。XAサービスで発生するコミット/ロールバックをRM(Resource Manager)に送信します。

- TLM (Transaction Log Manager)

トランザクションが発生するとき、実際のCLHがコミットを実行する前にTLMを使ってトランザクション・ログを保存します。トランザクション・ログはtlogに保存されます。

- RQS (Reliable Queue Server)

Tmaxシステムのディスク・キュー(Disk Queue)を管理するプロセスで、ファイルで発生する読み込み/書き込みを実行します。

- GW (Gateway Process)

複数のドメインで区分されている場合にドメイン間の通信を担当します。

- Tmadmin (Tmax Administrator)

Tmax関連情報のモニタリングおよび環境ファイルの変更などを管理します。

- RACD (Remote Access Control Daemon)

Tmaxがインストールされたすべてのドメインをリモートで統制します。

- TCS (Tmax Control Server)

CLHの要求によってビジネス・ロジックを処理し、結果を返します。

- UCS (User Control Server)

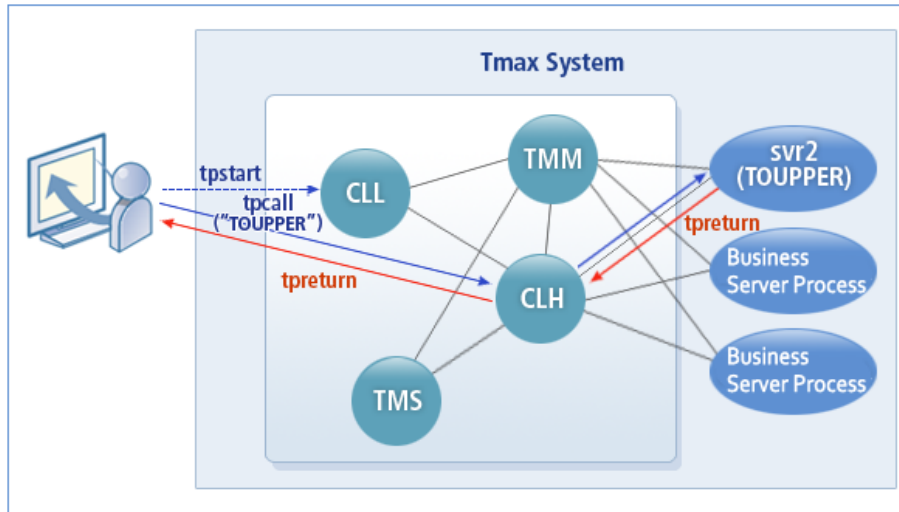
CLHの要求によってビジネス・ロジックを処理して結果を返し、当該プロセスがコントロールを維持します。

- TIP (Tmax Information Provider)

システム環境情報と統計情報を確認し、システムを運用および管理します(boot/down only)。

以下は、Tmaxシステムがサービスを実行する過程を示した図です。

【図 2.2】 Tmaxシステムのサービスの実行



以下は、Tmaxシステムがサービスを実行する手順についての説明です。

1. クライアントがtpstartすると、CLLプロセスが連結要求を実行し、内部動作によってCLHプロセスで連結処理されます。
2. サービス要求がある場合、CLHプロセスですべてのサービスを処理します。
3. CLHプロセスはクライアントのサービス要求(tpcall)を受信し、要求サービスを分析し、業務サーバー・プロセス(svr2)にサービスをマッピングします。
4. 業務サーバー・プロセス(svr2)はサービスを処理した後、CLHに結果を返します(tpreturn)。
5. CLHはサービス処理結果を受信してクライアントに送信します。

2.2.2. TIM

TIM(Tmax Information MAP)はTmaxシステムを運用する中核情報で、Tmaxで管理する共有メモリーです。TIMはエンジン・プロセスのうちTMMプロセスによって生成されます。

TIMは以下の4つの役割をもっています。

- Tmaxシステム設定情報

Tmax環境ファイルである<tmconfig.m>ファイルを管理するため、共有メモリーにローディングし、必要な場合に参照します。

- Tmaxシステム運用情報

Tmaxシステム運用に対する情報を管理します。システムに障害が発生した場合の対応方法、負荷分散のためのロードバランシング(Load Balancing)に対する基準情報、複数の装備で構成されたシステムの場合には各サービスにアクセスするためのネーミング(Naming)サービス情報とアプリケーション位置情報などを管理します。

- アプリケーション状態情報

システムにローディングされるアプリケーションのプロセス状態(Ready/Not Ready/Runningなど)を管理します。

- 分散トランザクション情報

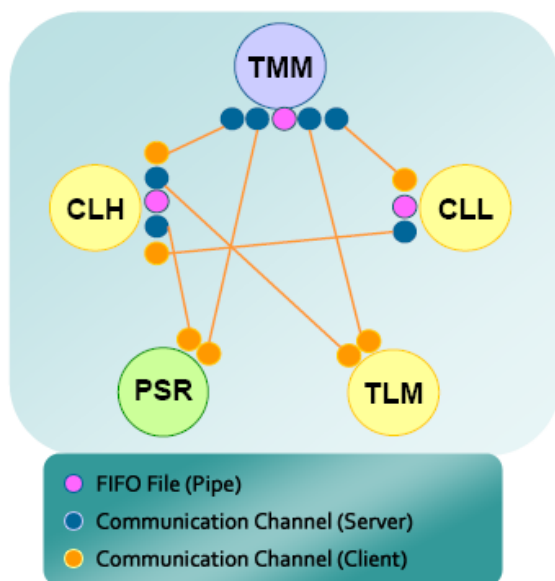
RMとの通信を中継するデータベースの運用情報およびトランザクション処理のための採番情報を管理します。

2.2.3. ソケット通信

TmaxはUNIXドメイン・ソケット(Domain Socket) 通信方式を使用します。UNIXドメイン・ソケット通信方式はソケットAPIを修正せずに使用し、ファイルを利用して内部プロセス間に通信を行う方式です。これは、ポート基盤の外部通信方式と違って、ファイルを利用した内部通信方式で、さらに安定さと高スピードを提供するメリットがあります。ファイルによる通信方式とカーネルでメッセージを管理するとの特徴はFIFO(名前付きパイプ、Named Pipe)と似ていますが、FIFOと違って双方向通信が可能であり、多数のクライアント/サーバー環境を構築しやすいです。

以下は、CLL、CLH、TLMがUNIXドメイン・ソケット通信をする過程を示した図です。PSRで表示されたアプリケーション・サーバーはCLL、CLH、TLMと連結されます。

[図 2.3]ドメイン・ソケット通信



2.3. Tmax機能

以下は、Tmaxの主要機能についての説明です。

- プロセス管理

Tmaxは3ティア基盤のクライアント/サーバー環境を提供し、クライアントごとに生成されるサーバーの業務処理プロセス数を調整し、システムを最適の環境で活用できるように管理します。

- トランザクション管理

分散トランザクションが処理されるとき、2フェーズ・コミット(2 Phase Commit)をサポートしてデータ整合性を保証し、簡単ないくつかの関数(tx_begin、tx_commit、tx_rollbackなど)を提供するほか、グローバル・トランザクションを容易にします。また、マルチスレッド方式のトランザクション・マネージャーを提供して少ないリソースに対して高い効率性を保証します。そして動的ロギングにより、エラーが発生する場合、迅速に対応するのでリカバリー/ロールバック(Recovery/Rollback)による安定性を保証します。すべてのトランザクションは中央で管理するのでトランザクションに対するスケジューリングや管理が簡単です。

- 負荷調整

Tmaxは以下のような3つの方式によって負荷調整機能を提供し、全体システムの処理量を増大させて処理時間を減らせます。

- SLM (System Load Management)
- DDR (Data Dependent Routing)
- DLM (Dynamic Load Management)

- 障害対策

ハードウェア的な障害が発生する場合はロードバランシング(Load Balancing)による障害対策とサービス・バックアップを行うことで正常に運用できます。サーバー・プロセスがダウンしてしまうソフトウェア的な障害が発生する場合、プロセスがリスタートされるので無停止サービスを提供できます。

- ネーミング・サービス(Naming Service)

ネーミング・サービスによって分散システム内のサービス位置情報を名前で提供し、簡単かつ容易にサービス呼び出しが可能で、位置の透明性を提供します。

- プロセス制御

Tmaxは、以下の3つの方式のデータ送信プロセスを提供します。これについての詳細な説明は「[2.3.6. プロセス制御](#)」を参照してください。

- TCS (Tmax Control Server)

- UCS (User Control Server)
- POD (Processing On Demand)
- RQ (Reliable Queue)機能の提供

サービス実行中に障害などで要求内容が消えてしまうことを防止するディスク・キュー(Disk Queue)処理によってデータを保存し、信頼性ある処理を保証します。
- セキュリティー機能

Tmaxはディフィー・ヘルマン(Diffie-Hellman)アルゴリズム基盤のデータ保護機能を提供し、UNIXで提供するセキュリティー機能を含めて以下の3段階のセキュリティー機能をサポートします。

 - 第1段階:システム接続認証

全体Tmaxシステム(ドメイン)に対する単一パスワードを設定し、このパスワードを登録したクライアントに限ってのみTmaxシステムへの接続を許可します。
 - 第2段階:ユーザー認証

Tmaxシステムに登録されているユーザーIDに限ってのみ認証を行ってTmax提供のサービスを使用できるように許可します。
 - 第3段階:サービス・アクセス認証

セキュリティー・レベルの高さが求められるサービスについて、権限のあるユーザーに対してのみ当該サービスをサポートします。Tmax v4.0からサポートしています。

参考

セキュリティー機能の詳細については、『Tmax アプリケーションガイド』の「第8章 セキュリティー・システム」を参照してください。

-
- 便利なAPIおよび多様な通信方式のサポート

同期通信(Synchronous)、非同期通信(Asynchronous)、対話型通信(Conversational)、リクエスト・フォワード(Request Forwarding)、アラーム通知(Notify)、メッセージの同時伝達(Broadcasting)のような多様な通信方式をサポートし、便利なAPIを提供します。
 - システム管理とリソース管理
 - プロセス状態、サービス・キューイング状態、サービス処理件数、平均サービス処理時間などのような全体システムの進行状況をモニタリングでき、システム状態およびキュー管理システムの統計分析および報告書の作成が可能です。

– アプリケーションとデータベースを統合して管理するので、リソースの効率的な管理が可能です。

- マルチドメインおよび多様なゲートウェイ・サービスの提供

長距離の分散システム間に相互にデータを交換し、他のプラットフォーム基盤のシステム間に簡単に連動します。SNA CICS、SNA IMS、TCP CICS、TCP IMS、OSI TPなどのような多様なゲートウェイ・モジュールをサポートします。ドメイン間のトランザクション・サービス処理およびルーティング機能が可能です。

- 多様なクライアントエージェントの提供

2ティア・システムから3ティア・システムへの変換が容易になるよう多様なエージェントを提供します。

区分	説明
RCA (Raw Client Agent)	マルチスレッド方式でプロセスを効率的に処理できるマルチポートをサポートします
SCA (Simple Client Agent)	Non-Tmaxクライアント/Tmaxクライアント両方に対応できるマルチポートをサポートします

- 多様な開発方式のサポート

区分	説明
RDP (Real Data Processor)	UDP通信データ・タイプを使用してTmaxシステムを経由せずに直接データを転送できるようにサポートします
Window制御	マルチ・ウィンドウ設定のためのクライアント・ライブラリー(WinTmax Library)を提供し、データを同時多発的に受け入れることができます

- 拡張サポート

- Webとの連動

クライアント/サーバー環境とWeb環境をJava Applet/Servlet、PHPなどを使って連動することで、迅速な応答時間を保証し、システム性能を向上できます。Tmaxではサービス連動を容易にするために**WebT**を提供します。WebTの詳細については、『Tmax WebT ユーザガイド』を参照してください。

- メインフレームとの連動

IBMなどレガシーシステムに存在していたアプリケーションサービスに対し、Host-linkを利用してクライアント/サーバー環境のアプリケーションサービスと同様にアクセスできます。

- 他のミドルウェアへの移行が容易

他のミドルウェア(Tuxedo、TOP END、Entera)で開発されたシステムもソース修正無しでTmaxに簡単に移行でき、さらに向上された性能とレベル高いテクニカルサポートおよびコスト削減の効果を期待できます。

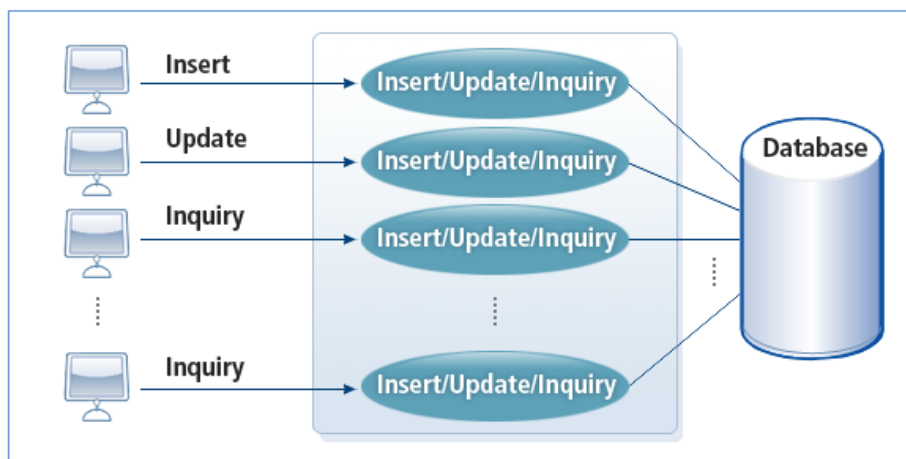
2.3.1. プロセス管理

既存の2ティア基盤のクライアント/サーバー環境はアプリケーション開発の最も一般的な方法で、1つのクライアントに対してサーバー・プロセスが1つずつ生成される方式です。つまり、クライアントの数が増加するとサーバー・プロセスの数も一緒に増加します。こうしたクライアントとサーバー・プロセスの増加はプロセス生成に必要な時間とファイルおよびデータベースの開始/終了(open/close)に長い時間が費やされ、1つのサーバーは連結されたクライアントでのみ使用できるので、サーバー・プロセスの顕著な使用低下と管理費用の上昇の問題が発生しました。

2ティア基盤のクライアント/サーバー環境は、こうした問題を解決すべくミドルウェアのTPモニターを利用して3ティアで構成したクライアント/サーバー構造を採用しました。TPモニターのTmaxは生成されたプロセスの数を調整し、休止(idle)中のサーバー・プロセスをスケジューリングし、サーバー・プロセスのチューニング(Tuning)による最適のシステムを運用します。

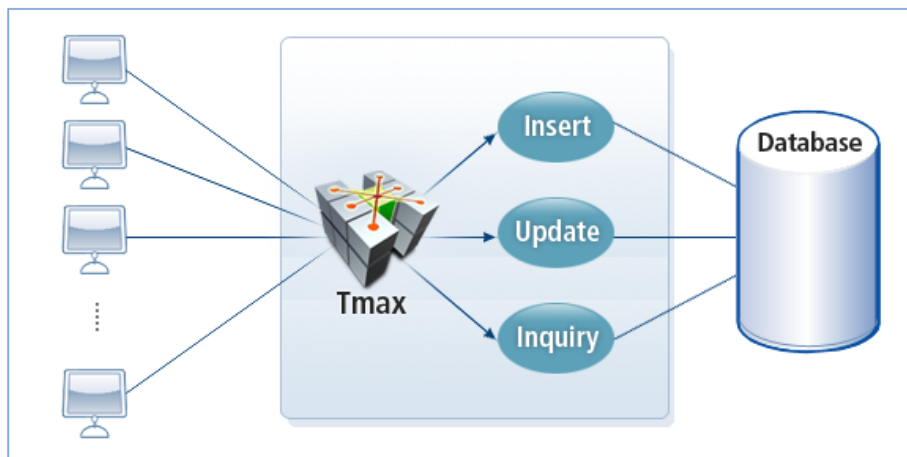
以下は、2ティアのクライアント/サーバー構造を示した図です。

[図 2.4] 2ティアのクライアント/サーバー構造



以下は、3ティアのクライアント/サーバー構造を示した図です。

【図 2.5】 3ティアのクライアント/サーバー構造



3ティア環境は2ティア環境より性能、拡張性、管理機能、障害対策の側面で向上されたシステムを構築でき、内部でプロセス管理モジュールによって安定的なサービスを提供します。3ティア環境でTmaxの内部プロセス管理モジュールは生成されるプロセス数を調整し、休止(idle)中のサーバー・プロセスをスケジューリングし、サーバー・プロセスのチューニング(Tuning)による最適のシステムを運用します。

以下は、2ティアと3ティアのクライアント/サーバー環境の比較です。

● 適用分野

2ティア環境	3ティア環境
<ul style="list-style-type: none"> – 小規模部署の単位別の業務 – 単一サーバー – 少数ユーザー(おおよそ50人未満) – バッチ業務分野 	<ul style="list-style-type: none"> – 全社的な業務 – マルチサーバー – 大規模ユーザー(50人以上) – OLTP業務および大量トランザクションの処理

● メリット

2ティア環境	3ティア環境
<ul style="list-style-type: none"> – プログラムの開発期間の短縮(テストが容易) – 初期の導入コストの削減 – 単純なプログラム開発が容易 	<ul style="list-style-type: none"> – アプリケーション開発のモジュール化 – 異機種のH/Wおよびデータベース環境で相互連動が可能 – システム・リソースを最大限に活用して性能向上 – 拡張が容易

2ティア環境	3ティア環境
	<ul style="list-style-type: none"> システム管理、負荷分散、障害対策の追加セキュリティ機能

● デメリット

2ティア環境	3ティア環境
<ul style="list-style-type: none"> トランザクション量が増加するとき、急激に性能が低下 他のプラットフォームおよびデータベース環境で相互連動が不可能 拡張が難しい システム管理、負荷分散、障害対策の追加セキュリティ機能の実装が不可能 	<ul style="list-style-type: none"> プログラムの開発時に長時間が必要(クライアント/サーバー結合テスト) 初期の導入コストの増加 単純なプログラムもクライアント/サーバーを分離して開発

2.3.2. 分散トランザクション

トランザクションは論理的に1つの単位で処理することで多様なリソースを一括して活用でき、分散されたリソース間の資料の整合性を維持できます。分散トランザクションはネットワーク上のシステム間のトランザクションを意味し、トランザクションのACID(Atomic、Consistent、Isolated、Durable: transaction properties)の特性を満たせる必要があります。Tmaxシステムは異機種または同種の複数DBMS間の分散されたトランザクションでACIDを保証します。

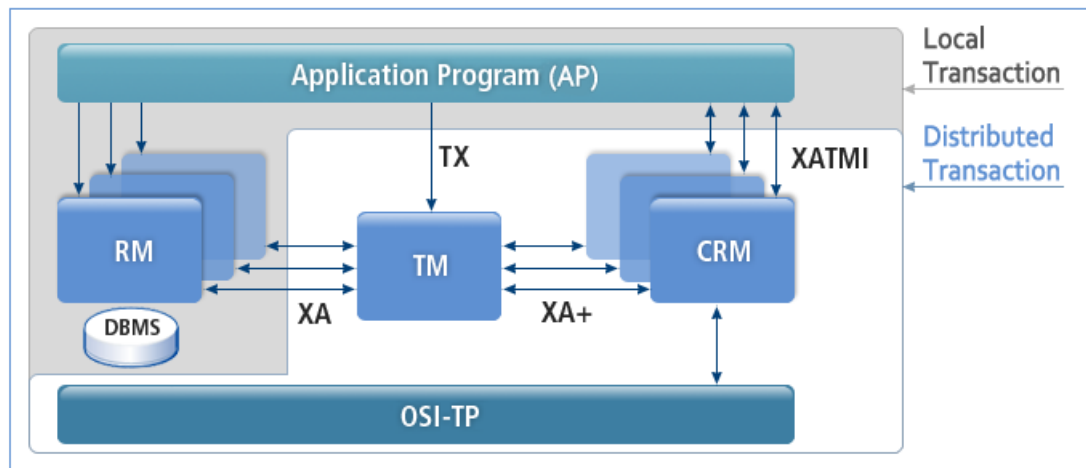
区分	説明
原子性(Atomicity)	トランザクションが実行されるか全く実行されない(all or nothing)2つの場合のみ存在しなければなりません
一貫性(Consistency)	トランザクションの実行結果を1つの一貫した状態から他の一貫した状態の共有リソースに更新します
孤立性(Isolation)	共有リソースを変えるとき、トランザクションの結果がコミットされるまでトランザクションの外部に現れません
永続性(Durability)	正常に完了されたトランザクションの結果は永久に反映されます

Tmaxの分散トランザクション管理は基本的に国際標であるのX/OpenのDTPモデルを遵守します。X/Open DTPモデルに従ってAP(Application Program)、TM(Transaction Manager)、RM(Resource Manager)、CRM(Communication Resource Manager)で構成されており、X/Open DTPモデルを遵守する標準関数の

集合体であるATMI関数をサポートします。また、X/Open DTPを遵守する相互異なる種類のDBMS間に発生するトランザクションをまとめて処理します。

以下は、X/Open DTPの構造です。

[図 2.6] X/Open DTPの構造



- AP (Application Program)
DT(Distributed Transaction)のBoundaryを提供します。
- RM (Resource Manager)
データベースなどのリソースにアクセス機能を提供します。
- TM (Transaction Manager)
DT別にIDを生成して進行を管理し、完了および失敗する場合、復旧機能を提供します。
- CRM (Communication Resource Manager)
分散AP間の通信を制御します。
- OSI-TP (Open System Interconnection-Transaction Processing)
相互異なるTM領域との通信を担当します。

分散トランザクションを処理する場合、2フェーズコミット(2 Phase Commit)をサポートしてデータの整合性を保証し、APIを提供してグローバル・トランザクションを容易にします。分散トランザクション管理とは、複数地域に分散されている環境で多数の異機種ハードウェア・プラットフォームおよびデータベースを利用して実行するトランザクション管理を意味します。

- 2フェーズコミット(Two-phase commit、2PC)プロトコル

2つ以上の同種および異種のデータベースが関連しているグローバル・トランザクションではトランザクションの属性を保証するために2PCを使用します。2フェーズコミット(Two-phase commit)とは、2つ以上のデータベースが連動するとき、ACID属性を完全に保証するために2段階処理をすることを言います。

– 第1段階：準備フェーズ(Prepare Phase)

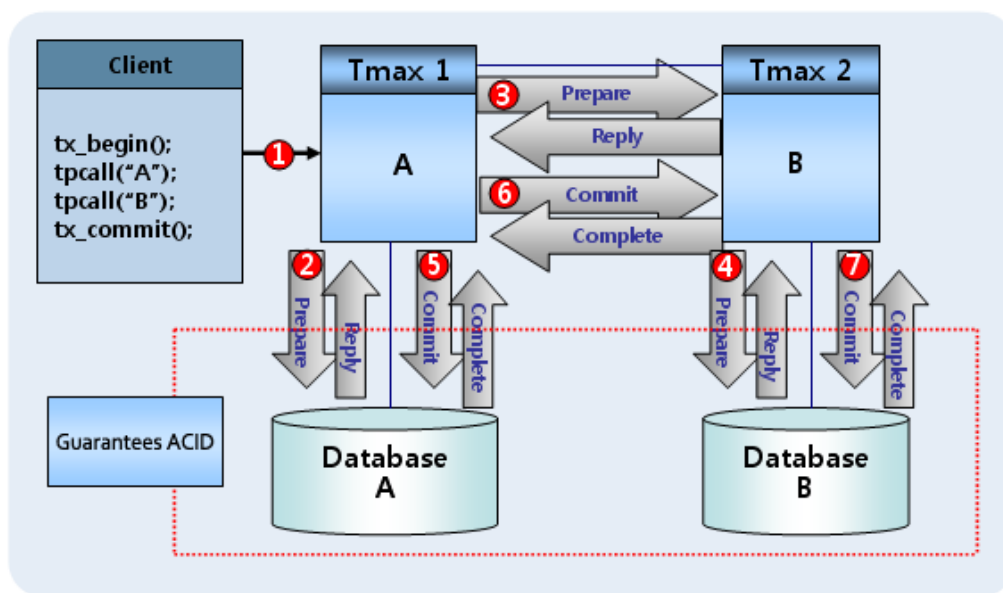
トランザクションに関連するすべてのデータベースにトランザクションを処理する準備がされているかを確認します。すべてのデータベースから準備が完了すると信号を渡します。1つの分散トランザクションとしてまとまっているそれぞれのデータベースやネットワーク、サーバーなどがコミットまたはロールバックを実行できるかをチェックし、データベースを準備させる段階です。

– 第2段階：コミットフェーズ(Commit Phase)

すべてのデータベースから正常信号を受信するとコミットされます。1つでも異常信号を受信するとロールバック処理してグローバル・トランザクションを完了します。コミット・メッセージをすべてのノードに送り、各ノードではRMでコミット要求を実行する段階です。ただし、コミット・フェーズに参加するすべてのノードがコミットの完了(Commit Complete)によってtx_commit()を要求した側に正常処理されたことを知らせる時点でデータ変更作業が完了された点に注意します。

以下は、2フェーズコミットの動作過程を示した図です。

[図 2.7] 2フェーズコミットの動作



● グローバル・トランザクション(Global Transaction)

多数の異機種ハードウェアおよびデータベースを1つの論理的な単位(トランザクション)で取り扱って作業します。2つ以上の同種または異機種システム上に存在するDBMSと関連されたグローバル・トランザクションを処理時、2フェーズコミットをサポートしてデータ整合性を保証することで分散トランザクションを完璧に処理します。Tmaxシステムは非常に簡単な関数(tx_begin、tx_commit、tx_rollbackなど)を提供して

グローバル・トランザクションを容易にサポートします。グローバル・トランザクションで処理されるとき、ノード間の通信はクライアント・ハンドラーによって実行されます。

以下は、グローバル・トランザクションの動作についての説明です。

– 第1段階：準備フェーズ(Prepare Phase)

分散トランザクションを発生させたノード(グローバルコーディネーター)が分散トランザクションに参加したノードに対してコミットまたはロールバックを実行していいかを確認する段階です。

– 第2段階：コミットフェーズ(Commit Phase)

分散トランザクションに参加したノードがグローバル・コーディネーターに対して分散コミットしてもいいとの応答(準備済み)を受けてトランザクションをコミットします。いかなるノードでも準備されていないとの応答があればトランザクションをロールバックします。

- リカバリー/ロールバック(Recovery / Rollback)

トランザクションが失敗すれば現在使用中のRMの内容が変更されても以前の内容に回復します。

- トランザクションの中央管理

ノードが物理的に遠く離れていてもノード間に発生するトランザクションに対して中央で管理および統制します。

- トランザクション・スケジューリング

優先順位を考慮した同時性制御手法をサポートします。

2.3.3. 負荷調整(ロードバランシング)

Tmaxは多様な負荷調整機能を提供し、全体システムの処理量を増大させて処理時間を短縮します。TmaxはSLM(System Load Management)による負荷調整、DDR(Data Dependent Routing)による負荷調整、DLM(Dynamic Load Management)による負荷調整を提供します。

SLMによる負荷調整

SLM(System Load Management)は定義された負荷比率で分散処理する方式です。ハードウェア性能によって固有の作業処理量を設定して1つのノードのサービス要求量が作業処理量を超過すると他のノードにサービス連結を転換する方式です。ノードごとに処理量の設定を変えて負荷を処理できます。

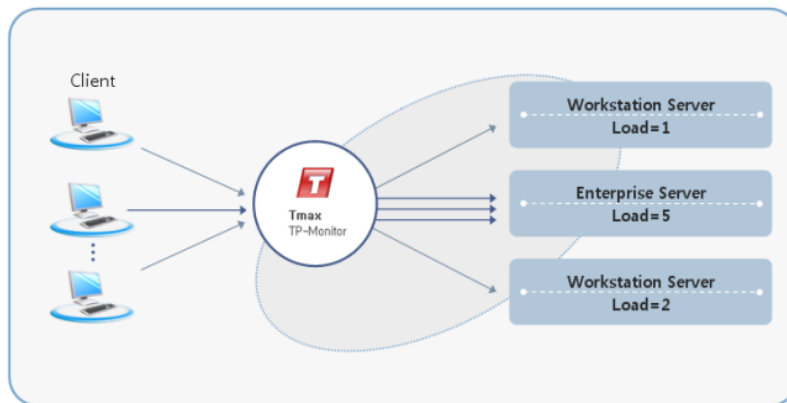
SLMは以下のような動作方式で各プロセスで処理されます。

1. クライアントの要求をCLHが受信します。
2. CLHはTIMを使ってSLMサービスかどうかを判断してサーバー・グループ別の処理件数を確認します。

3. CLHが適切なサーバー・グループにスケジューリングします。

以下は、SLMによって負荷を調整するプロセスの例で、Node 1、Node 2、Node 3に作業処理量をそれぞれ1:5:2で設定しておく、Node 1で1つの作業を処理し、その次の作業はNode 2で、そしてその次の作業はNode 3で処理します。その後連続した3つの作業はNode 2ですべて処理します。

[図 2.8] SLMによる負荷調整



DDRによる負荷調整

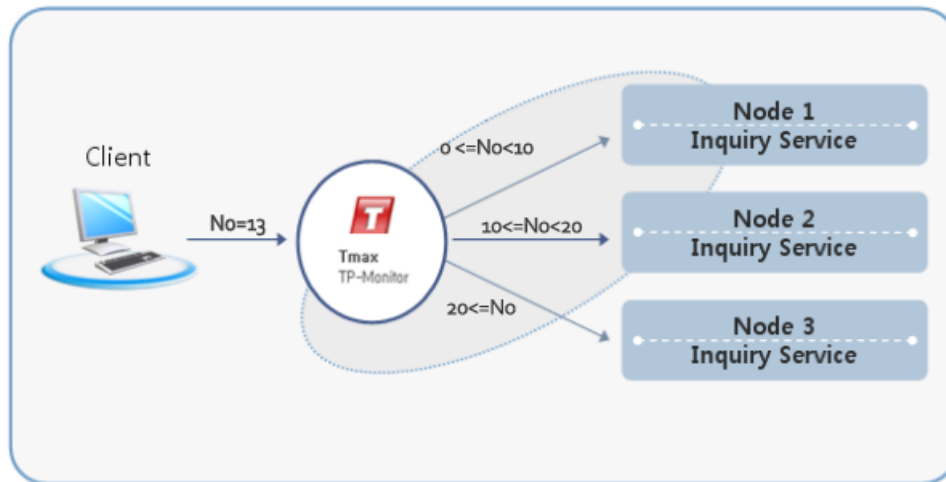
DDR(Data Dependent Routing)はデータ値によって分散処理する方式です。複数のノードで共通したサービスを提供するとデータ範囲によってノード間のルーティングができるように指定します。入力されたフィールドの値を確認して適切なサーバー・グループに当該サービスを要求します。

DDRは以下のような動作方式で各プロセスで処理されます。

1. クライアントの要求をCLHが受信します。
2. CLHはTIMを使ってDDRサービスかどうかを判断し、CLHが区分フィールド値を確認します。
3. CLHが定義されたサーバー・グループにスケジューリングします。

以下は、DDRによって負荷を調整するプロセスの例で、年齢層別にそれぞれ異なるノードで顧客照会サービスを提供すれば、Node 1では0~10歳まで、Node 2では11~20歳まで、そしてその他の年齢はNode 3で処理されます。

[図 2.9] DDRによる負荷調整



DLMによる負荷調整

DLM(Dynamic Load Management)は負荷比率によって動的に処理グループを選択する方式です。特定ノードに負荷が集中される場合、Tmaxの道程負荷調整方法によって負荷を分散して全体システムの処理量を増加させて処理時間を短縮します。システムの負荷は起動されているプロセスのキューイング状態で判断されます。

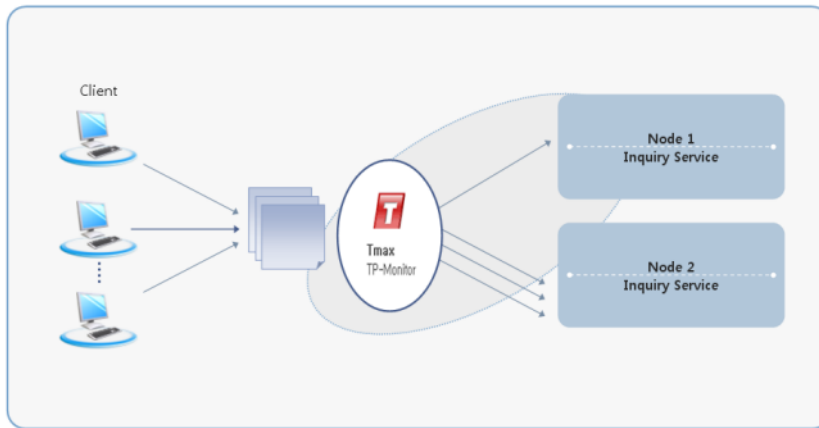
Tmaxシステムはプロセス別にメモリー・キューを管理し、要求されたサービスに対して現在マッピング可能なプロセスがない場合にはメモリー・キューに保存します。メモリー・キューに積載されている取引の件数がシステムの負荷を意味します。

DLMは以下のような動作方式で各プロセスで処理されます。

1. クライアントの要求をCLHが受信します。
2. CLHはTIMを使ってDLMサービスかどうかを判断し、CLHがサーバー別にキューイング件数を確認します。
3. CLHが臨界値に達すると次のサーバー・グループにスケジューリングします。

以下は、DLMによって負荷を調整するプロセスの例で、Node 1、Node 2でサービスを同一に提供し、Node 1にサービス要求が集中される場合、Tmaxで提供する動的分散アルゴリズムによって負荷を分散して処理します。

[図 2.10] DLMによる負荷調整



2.3.4. 障害対策(フェイルオーバー)

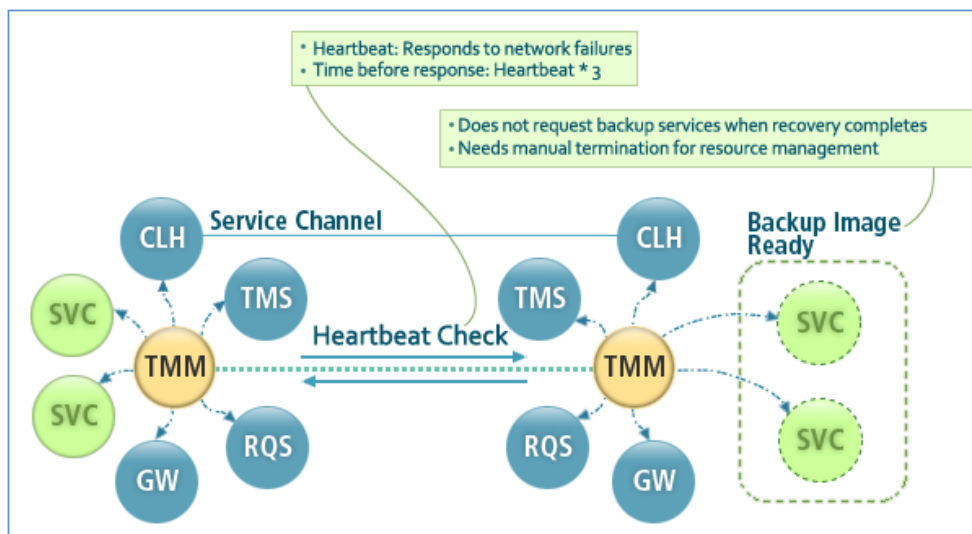
Tmaxはシステムリソースの高い可用性を保証すべく、マシン、ネットワーク、システム、サーバー・プロセスなどの障害が発生する場合は、障害対策によって中断されないサービスを提供できます。障害はハードウェア的な障害とソフトウェア的な障害に区分できます。

ハードウェア障害

ハードウェア的な障害が発生する場合、ロードバランシングによる障害対策とサービス・バックアップによる正常運用が可能です。サーバー・プロセスがダウンされるソフトウェア的な障害が発生するときもプロセスが再開されて中断なくサービスが提供されます。

Tmaxのシステム構成は各ノードが相互異なるノードを監視するPeer-to-Peer方式です。したがって、いくら多くのノードであっても同一な条件で即刻的に障害に対応できます。

[図 2.11] ハードウェア障害

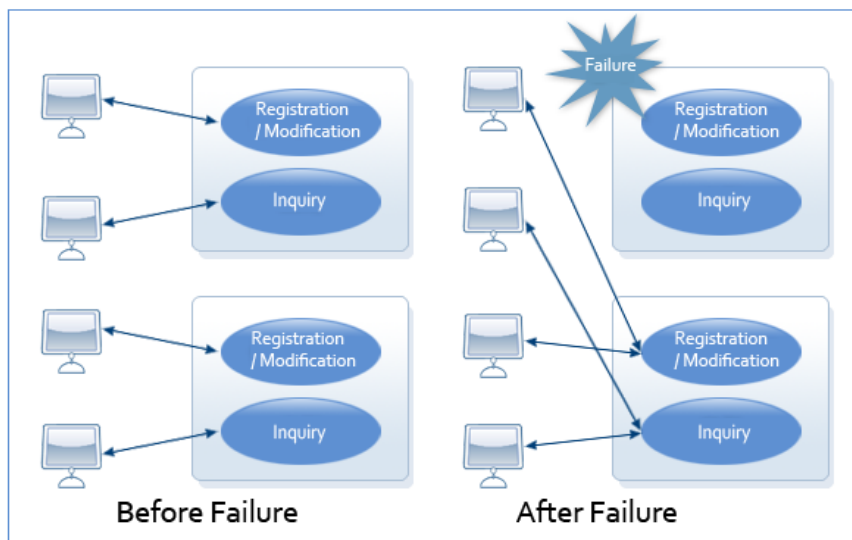


ハードウェア障害には以下のような2つの障害対策があります。

- 負荷調整による障害対策

1つのサービスが複数のノードで提供される場合には1つのノードで障害が発生しても他のノードで継続的なサービスを提供します。クライアントからバックアップ・ノードに再接続してサービスを要求します。

[図 2.12] 負荷調整による障害対策



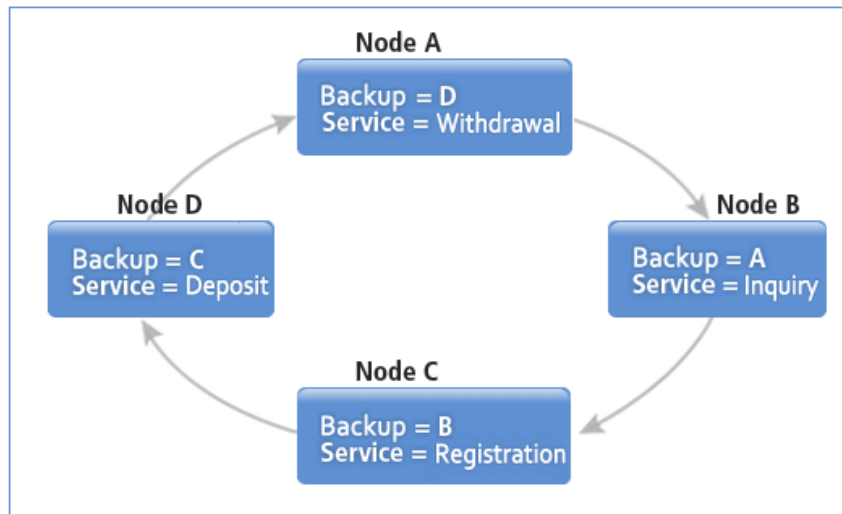
- サービス・バックアップによる障害対策

ノード障害が発生する場合、他のノードであらかじめ準備されているバックアップ・プロセスを動作させてサービスを処理します。

- 障害発生前(正常運用)

すべてのノードにおける障害を完璧にサポート(Peer-to-Peer方式)

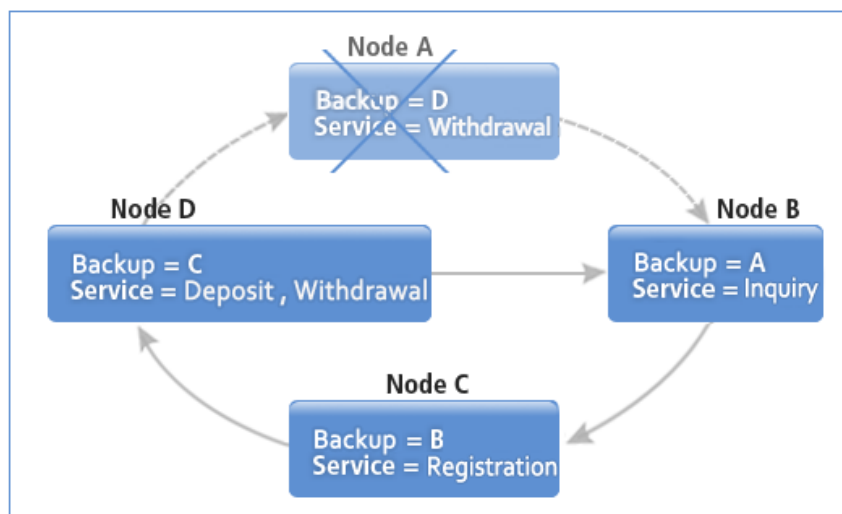
[図 2.13] サービスバックアップによる障害対策-障害発生前



– 障害発生後(正常運用)

障害が発生されたとき、指定されたバックアップ・ノードで中断しないサービスを提供

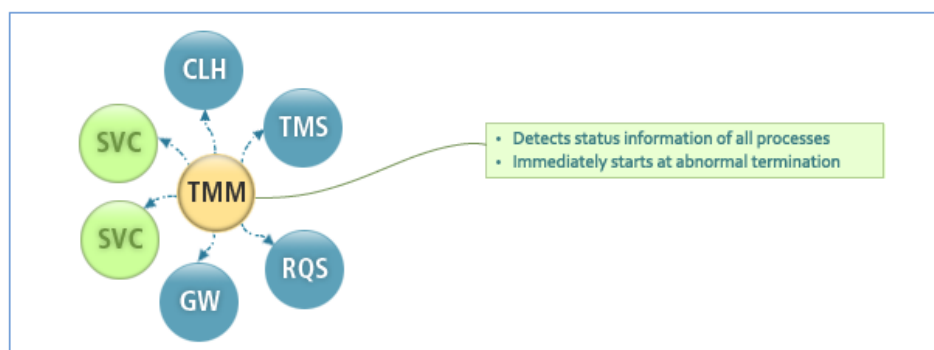
[図 2.14] サービス・バックアップによる障害対策-障害発生後



ソフトウェア障害

プログラムの内部的なバグ、またはユーザーの手落ちによって異常にサーバー・プロセスがダウンされた場合、自動で再開始できる機能を提供します。ところが、TMS、CAS、CLHのようなシステム・プロセスは制限無く無限に再開始され、これらプロセスは異常終了されたときには処理中であったサーバー・プロセスも一緒にダウンされる可能性があるので注意が必要です。

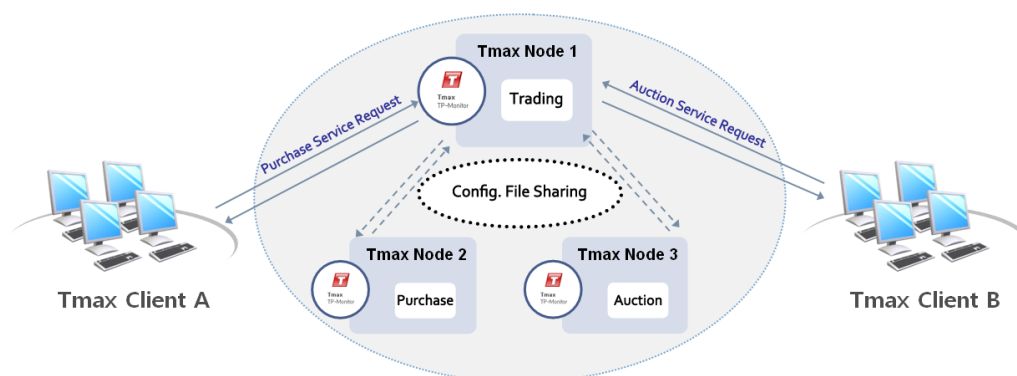
[図 2.15] ソフトウェア障害



2.3.5. ネーミング・サービス

Tmaxで提供するネーミング・サービスは、サービス呼び出しが簡単にできるようにすることで位置透明性 (Location Transparency)を保証します。Tmaxはネーミング・サービスによって分散システム内のサービス位置情報を名前前で提供し、簡単かつ容易なサービス呼び出しが可能で、位置透明性を提供します。ネーミング・サービスは簡単明瞭なサービス呼び出しが可能で、サービス名だけ呼び出して該当するサービスを受け取るのでプログラミングが簡単です。クライアントはサーバーのアドレスを知らなくてもサービス名からサーバーの情報を取得できます。

[図 2.16] ネーミング・サービス



2.3.6. プロセス制御

Tmaxでは3種類のサーバー・プロセスをサポートします。

- TCS (Tmax Control Server)

クライアントが要求すれば受動的に実行されるプロセスで、ほとんどはこのプロセスが使われます。クライアントの要求を処理するサーバーは事前に起動されている状態でなければなりません。クライアントの要求プロセスを処理する典型的な処理方法です。呼び出し側の要求をTmaxハンドラーから受信して業務を処理し、その結果をTmaxに返して次の要求を待機します。

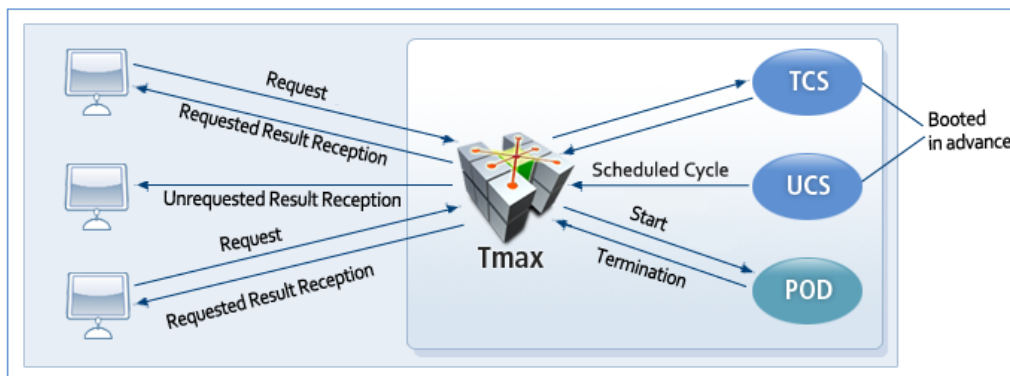
- UCS (User Control Server)

UCSは呼び出し側の要求がなくても能動的にデータを渡せるプロセスで、Tmaxの固有機能です。クライアントの要求を処理するサーバーは事前に起動されている状態でなければなりません。クライアントの要求がなくてもクライアントに持続的にデータを送信でき、同時にTCSプロセスと同様に呼び出し側の要求を受け取って業務を処理することもできます。つまり、UCSはTCSと同様にクライアントの要求を処理しつつ、自発的かつ能動的に業務を処理できる機能が付与されたプロセスです。

- POD (Processing On Demand)

PODはクライアントの要求があるときのみサーバー・プロセスが起動されて処理できるようにする方式です。クライアントの要求があるときのみ起動し、業務実行後は終了されるプロセスで、要求が多くない業務に適合している形です。

[図 2.17] プロセスタイプ



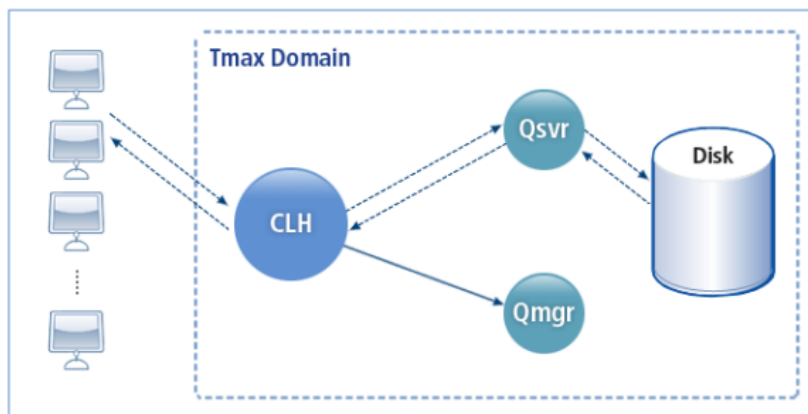
参考

サーバー・プロセス制御の詳細については、『Tmax アプリケーションガイド』の「第3章 サーバー・プログラム」を参照してください。

2.3.7. RQ機能

TmaxのRQ(Reliable Queue)はサービス実行途中の障害などによって要求内容が消えてしまうことを防止してサービスが信頼性ある形で処理されるようにします。多くの時間が必要な作業や、必ず信頼性の保証が必要な業務の場合、要求を受けた業務をいったんディスクに保存して処理します。システム障害または他の致命的な要因が発生したときにもシステム復旧後に当該業務を処理する方式です。

[図 2.18] RQプロセス



tpenq()関数の返却値によって要求されたサービスが正確にディスクに保存されたかを確認することができます。キューイング業務処理はキュー・マネージャー(Queue manager、Qmgr)が独立的に担当することで他の業務処理には全く影響を与えません。tpdeq()関数を使用してキュー・マネージャーの処理結果に対する正常・異常状態を確認できます。

2.3.8. セキュリティー機能

Tmaxはディフィー・ヘルマン(Diffie-Hellman)アルゴリズム基盤のデータ保護機能を提供し、UNIXで提供するセキュリティー機能を含めて5段階のセキュリティー機能をサポートします。

2.3.9. システムのリソース管理

システム管理(System Management)

プロセス状態、サービス・キューイング状態、サービス処理件数、平均サービス処理時間のような全体システムの進行状況をモニタリングすることができ、システム状態およびキュー管理システムの統計分析および報告書を作成することができます。

システム管理は静的システム管理、動的システム管理、モニタリングおよび運用(Administration)機能をサポートします。

- 静的システム管理

ユーザー環境に合わせてTmaxシステムを構成するサービス、サーバー、サーバー・グループ、ノード、ドメインなどに対する全般的なシステム環境を設定します。

- 動的システム管理

Tmaxが動作中であっても各構成要素を項目別に変更することができます。

項目	説明
ドメイン(Domain)	サービス・タイムアウト時間、トランザクション・タイムアウト時間、ノード(machine)の生存確認(live check)時間などを変更します
ノード(Node)	メッセージ・キューのタイムアウト時間を設定します
サーバー・グループ(Server group)	ノード別の負荷(load)値、負荷調整方式などを変更します
サーバー(Server)	最大キュー数(Max queue count)、キューイング時のサーバーの開始数(start count)、サーバーの再開数(restart count)、対象サーバー数、サーバー優先順位などを変更します
サービス(Service)	サービス別の優先順位、サービス・タイムアウト時間などを変更します

- モニタリングおよび運用(Administration)機能

- 動的環境設定の変更が可能です
- 各種資料を出力し、サーバーのトランザクション処理量、サービス別の処理件数、平均処理時間などの各種統計情報を提供します。

リソース管理(Resource Management)

アプリケーションとデータベースの統合管理によってリソースを効率的に管理することができます。

既存システムでは全体システムに対する管理が行われなくなりリソースの浪費が発生していたため、分散システムではより確実な管理機能が求められました。Tmaxは全体の分散システムに対する中央集中型のモニタリング機能を使ってアプリケーションを管理します。

単一アプリケーションに対して同種のデータベースまたは異種のデータベースを一緒に使用する場合、Tmaxは多様な種類のリソースをアプリケーションレベルで統合管理することができます。

2.3.10. マルチドメインおよび多様なゲートウェイ・サービスの提供

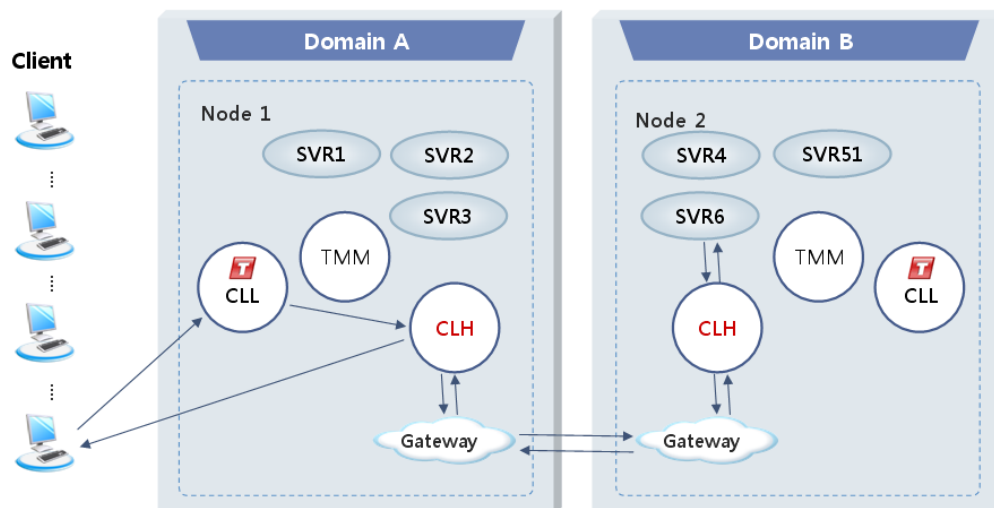
TmaxドメインはTmaxで独立的に管理(start/down)される最上位の単位で地域別または業務別にシステムを複数のドメインで分散管理するときにもドメイン・ゲートウェイを介してドメイン間の連動だけでなくマルチドメインの2フェーズ・コミットなどの機能をサポートします。

Tmaxは長距離分散システムの相互データを交換し、他のプラットフォーム基盤システム間の容易な連動をサポートし、SNA CICS、SNA IMS、TCP CICS、TCP IMS、OSI TPなどのような多様なゲートウェイ・モジュールをサポートします。ドメイン間のトランザクション・サービス処理およびルーティング機能を実行できます。

複数のノードを1つのドメインで管理するときに発生する全体ノード管理の厳しさ、ノード間通信のトラフィック急増などの問題を解決できます。また、いかなるシステムでも要求されたサービス进行处理することができます。マルチドメイン環境におけるサービスは、ドメイン間のサービス処理、ドメイン間のルーティング、ドメイン間のトランザクション方式で実行されます。

以下は、マルチドメイン構成に伴うサービス呼び出しのフローについての説明です。マルチドメインはゲートウェイによって連結されます。ドメイン・ゲートウェイは相互の連結を結ぶとき、サーバーまたはクライアントで動作します。ゲートウェイ間の起動順序はありません。

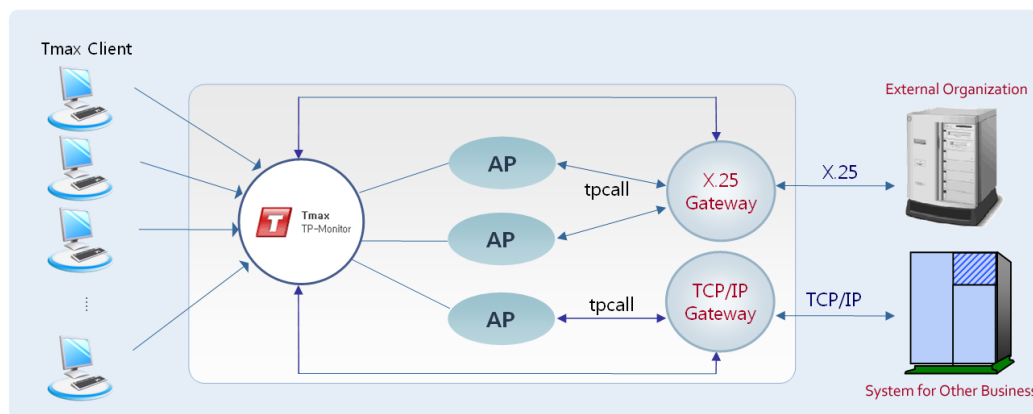
[図 2.19] マルチドメインのサービスフロー



TmaxはTCP/IP、X.25、SNAなどの多様なゲートウェイを提供して他業務システム/対外機関との連動を容易にします。ゲートウェイは効果的な通信を可能にし、業務ロジックと連携モジュールを分離することで管理の利便性を提供します。ゲートウェイはTmaxによって管理されるので自動で障害を復旧することができ、管理者による別途管理が要りません。

以下では、ゲートウェイの動作についての説明で、クライアントがサービスを要求すると当該サービスがあるドメインからその処理結果の応答を受けることを示しています。この際、両側のドメインではゲートウェイを介してトランザクション・サービスや値によってドメイン間にルーティングできます。

【図 2.20】 Tmaxゲートウェイの動作



2.3.11. 多様なクライアント・エージェントの提供

2ティア・システムから3ティア・システムへの変換が容易にできるように多様なエージェントを提供します。

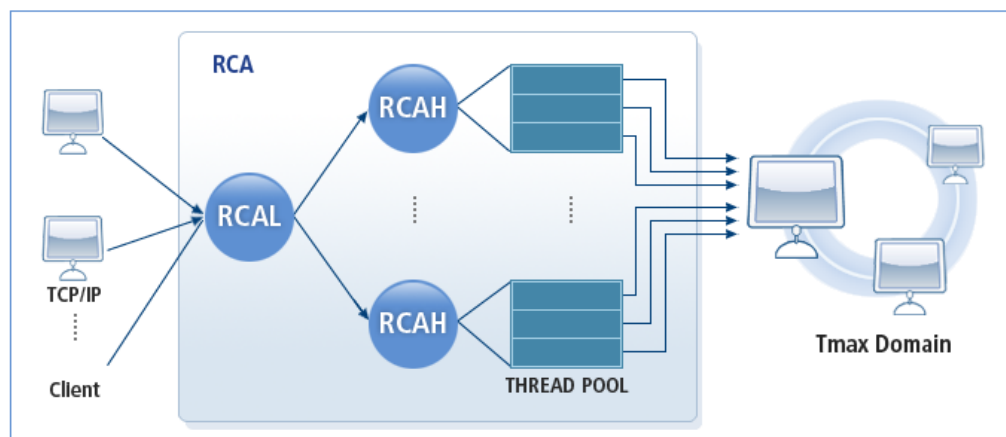
RCA (Raw Client Agent)

RCAはTmaxクライアント・ライブラリーを使用できない既存の通信プログラムとTCP/IPソケットに連結してTmaxシステムで提供するサービスを利用できるようにサポートするエージェントです。RCAはリモート(REMOTE)モードまたはローカル(LOCAL)モードのサービスをサポートします。既存Tmaxクライアント・ライブラリーのように1つのクライアント・ライブラリーは1つのクライアント・プログラムでのみ使用します。ところが、RCAはマルチスレッド方式で作成され、1つのスレッドが1つのTmaxクライアントで動作します。また、多様な形態のクライアントをサポートするために最大32個までのマルチポートを指定できます。

RCAはユーザーの接続を制御するRCAL、ユーザーのロジックと一緒に生成されるRCAHによって安定的な障害対策が可能で、管理ツールのrcastatとrcakillを提供します。

以下は、RCAの構造です。

【図 2.21】 RCAの構造



SCA (Simple Client Agent)

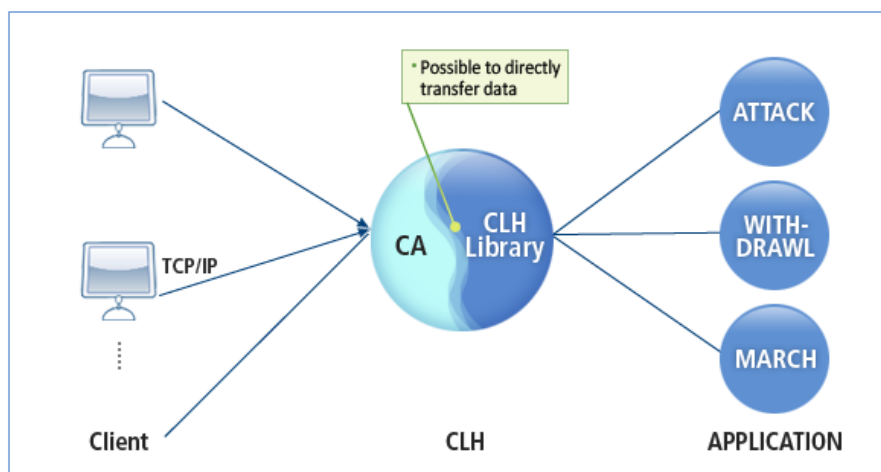
SCAはTmaxクライアント・ライブラリーを使用できない既存の通信プログラムとTCP/IP通信で連結し、Tmaxシステムで提供するサービスを利用できるようにサポートするエージェントです。SCAはカスタマイズ・ルーチン(customizing routine)とCLHライブラリーで構成されます。CLHはCLHライブラリーとリンクされます。

クライアントとの接続と解除、データ送受信に該当するネットワークでの処理部分はシステム内部的に処理されます。開発者は、クライアントや事前定義されたポート番号をTmax環境ファイルに設定してクライアントと接続でき、受信されたデータと送信するデータを開発者の意図に合わせて修正および補完できます。

SCAは多様な形態のクライアントをサポートするために最大8個までポートを指定できます。SCAモジュールはCLHモジュールと直接的なデータのやり取りが可能で、Non-Tmaxクライアント/Tmaxクライアントを同時に収容することができます。SCAのサービス方式はTCP/IPロー(raw)ソケットを利用する方法とTmaxクライアント・ライブラリーを利用する方法に区分できます。

以下は、SCAのうち1つのCA(Client Agent)を利用したサービス呼び出しの形態を示した図です。

[図 2.22] CAを利用したサービス呼び出しの形態



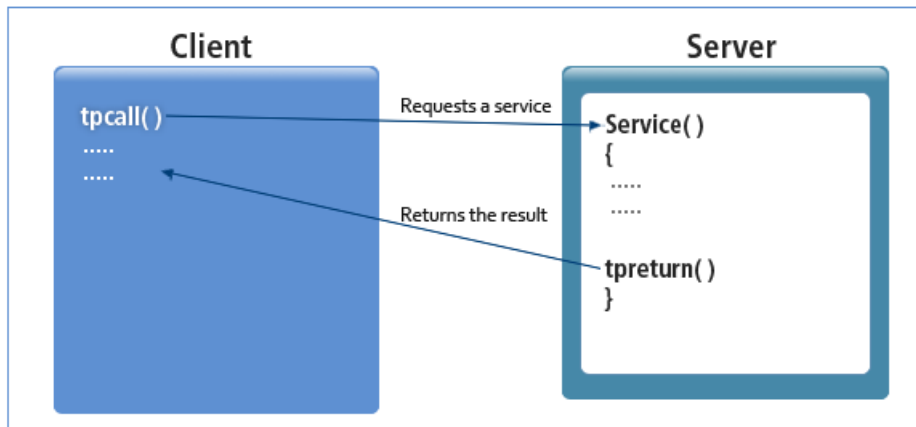
2.3.12. 多様な通信方式のサポート

クライアントからサーバーへの通信を要求する以下のような4つの方式をサポートします。

- 同期通信

クライアントからサーバーに要求を送り、サーバーから応答が来るまでブロッキングされて待機します。

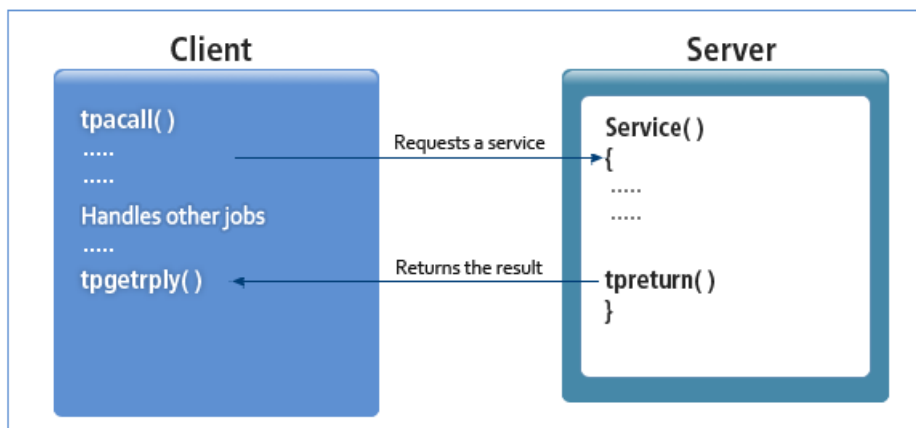
[図 2.23] 同期通信



- 非同期通信

クライアントがサーバーに要求を送った後、サーバーから応答が来るまで待機せずに他の業務を処理できます。応答を受けようとする場合、関数を利用してサーバーから応答を受けます。

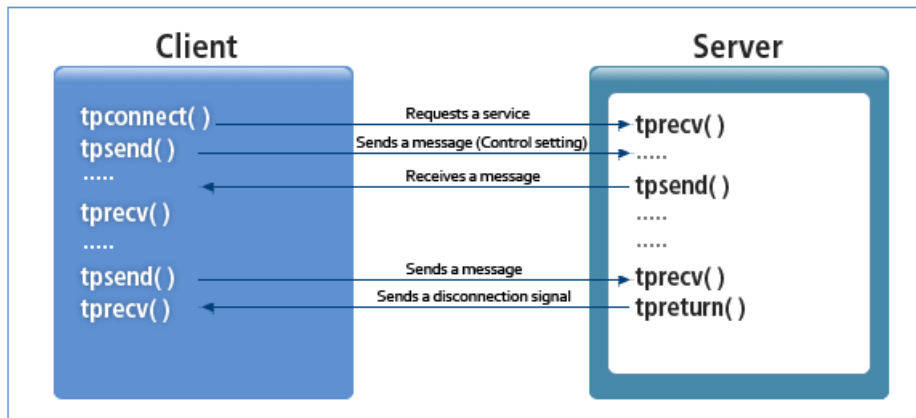
[図 2.24] 非同期通信



- 対話型通信

クライアントがサーバーに要求を受ける場合、サービスを連結し、サービス要求を送ります。メッセージを受信するときは対話型通信関数を利用して受信します。クライアント/サーバー間のローカル通信によってコントロール(control、制御権)をやり取りしつつメッセージを送受信します。コントロールを持っている側がメッセージを送信できます。通信が実行されたとき、接続記述子(connection descriptor)が返され、返却された接続記述子がメッセージの伝達を確認するために使われます。

[図 2.25] 対話型通信

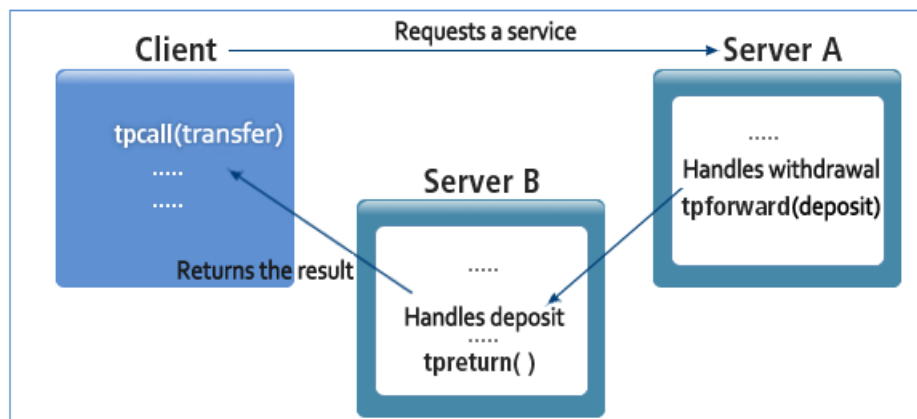


- 伝達型通信

- タイプA

ビジネス・ロジックをモジュール化して段階的なサービス処理が可能なタイプで、各モジュールに対する使用効率を向上させます。問題分析および修正時に体系的なアプローチが可能で、同期と非同期を混用できます。

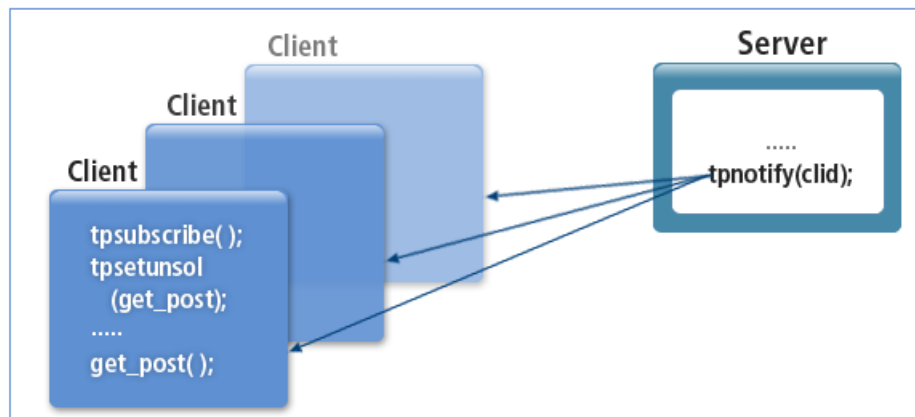
[図 2.26] 伝達型-タイプA



- タイプB

対外機関のような既存のレガシーシステムと連動するタイプで、サーバー・プロセスがブロッキングされずに持続的に渡されるサービス要求を処理します。レガシー・システムから渡される応答を最終的に呼び出し側に伝達できる形態のサービスをサポートします。

[図 2.27] 伝達型-タイプB

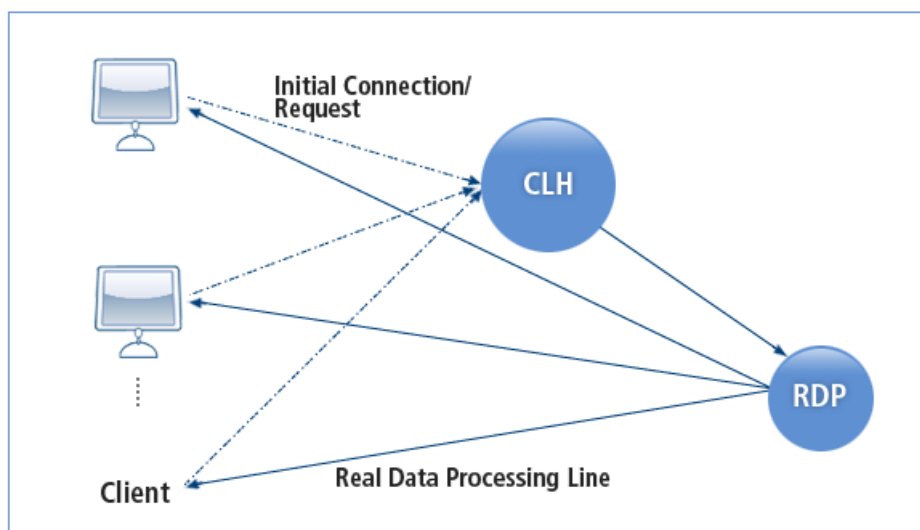


2.3.13. 多様な開発方式のサポート

RDP (Real Data Processor)

リアルタイムで持続的に変わるデータをより迅速に処理するため、CLH(Client Handler)を介さずにRDPからクライアントに直接データを送信できます。RDPを利用すると、CLHの負荷を大幅に減らせるのでTmaxシステムの全体的な性能を向上させることができます。RDPはUDPデータタイプに対してのみサポートしています。RDPはUCSサーバー・プロセスの形態で構成されます。

[図 2.28] クライアントとRDPの直接的なデータ送信



Window制御

Window基盤のクライアント・プログラムで利用できる便利なライブラリーを提供します。WinTmaxライブラリーとtmaxmtライブラリーの2種類があり、基本的にスレッド基盤で動作します。

- WinTmax

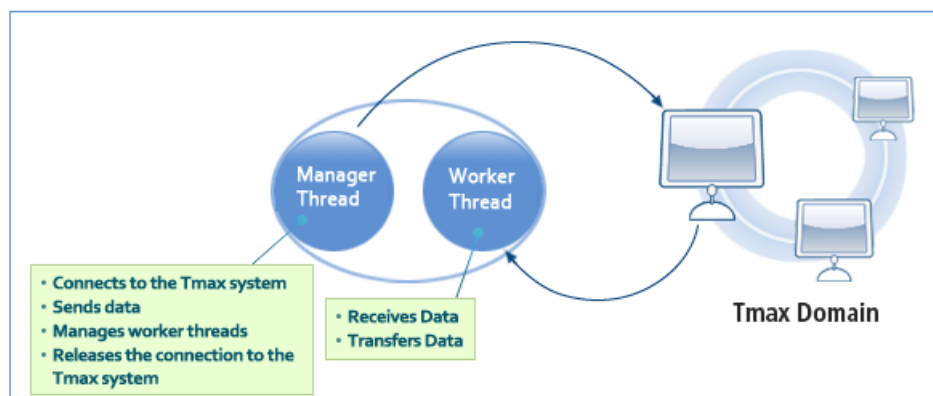
Window基盤のクライアント・ライブラリーで、マルチウィンドウ(Multi Window)を設定できるように設計されています。

WinTmaxライブラリーは内部的に管理スレッドと作業スレッドで構成されます。マルチウィンドウの設定が可能で、256個までのウィンドウ設定ができるので、同時多発的に発生するデータ処理に有効です。

区分	説明
管理スレッド	Tmaxシステムとの接続、データの送信、作業スレッドの管理、Tmaxシステムとの接続解除の機能を実行します
作業スレッド	データの受信と特定ウィンドウにデータ送信機能を実行します

以下は、WinTmaxライブラリーの構造を示した図です。

[図 2.29] WinTmaxライブラリーの構造および機能



- tmaxmt

クライアント・プログラムをスレッド化できるように開発されました。開発者はスレッドを使用してプログラムを作成しなければならず、それぞれのスレッドで指定された関数(WinTmaxAcall()、WinTmaxAcall2())を使用してデータを送受信することができます。それぞれの関数は内部的にスレッドを作成し、サービスを呼び出し、その結果を受けて指定されたウィンドウや関数にデータを渡します。

WinTmaxAcallは指定されたWindowにSendMessageを使用してデータを送信します。一方、WinTmaxAcall2は指定されたコールバック関数にデータを送信します。

2.3.14. メッセージの安定的な伝達

TmaxはHMS(Hybrid Messaging System)を使って安定的なメッセージの送受信をサポートします。Tmax HMSは以下のような特徴を持ちます。

- Hybridアーキテクチャー

TPモニター(Tmax)とMOM(Messaging Oriented Middleware)が共存し、MOMとRM間の2PC機能を提供します。Tmax HMSはTmaxのTPモニター機能に追加でMOM機能を提供するためのモジュールで、モジュール間に柔軟に連携できる構造を提供します。HMSはTMSとTPモニター(Tmaxの基本機能)とMOMとが共存する構造を持ちます。

- 信頼性ある障害復旧

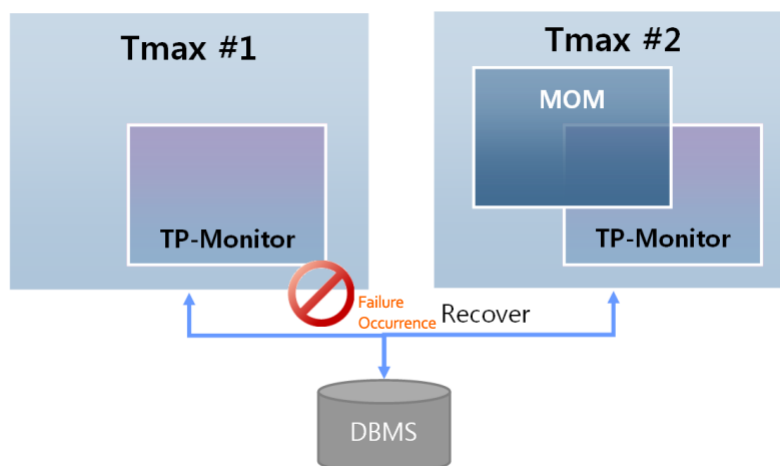
パーシステント(永続、Persistent)メッセージを保存するためにDBMSを使用します。

- 高可用性

- バックアップ・ノードを指定して障害に柔軟に対応します。Tmax HMSはストレージを利用し、障害復旧および信頼性あるメッセージの送受信を保証します。障害復旧の信頼性を保証するためにDBMSを使用し、障害が発生するとパーシステント・メッセージをDBMSから復旧します。

- Active-StandbyのHA(High Availability)構成によってシステム障害に対応します。

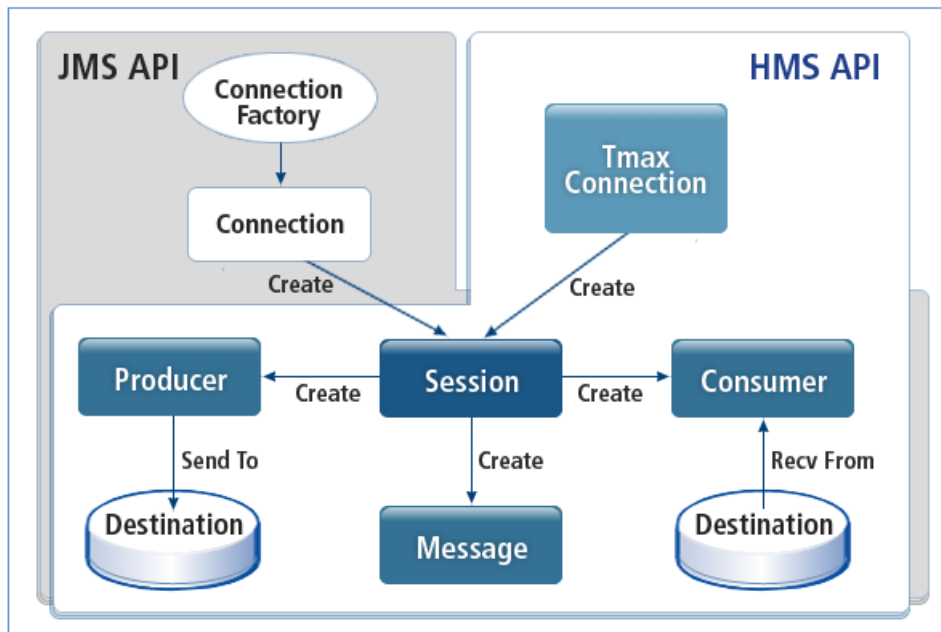
[図 2.30] HMSの高い可用性



- JMSメッセージング・モデルの採用

JMSメッセージング・モデルを採用してP2P、Publish/Subscribeをサポートし、JMSスペックと類似したC APIを提供して使用しやすいです。

[図 2.31] JMSのAPI互換性

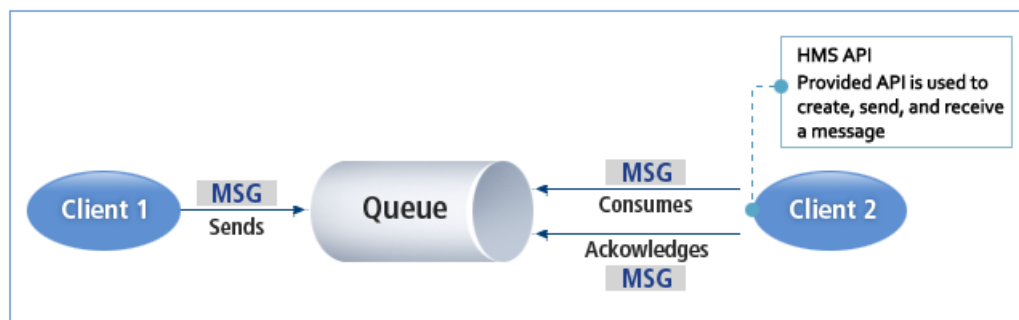


HMSはTmaxの機能で、SenderとReceiverの疎結合(loosely coupled)のための通信媒介であり、キュー(Queue)方式とトピック(Topic)方式をサポートします。

- キュー方式(Point-to-Point)

各メッセージは1つの消費者(consumer)に対してのみ送信され、送/受信間の時間依存性(timing dependency)がありません。

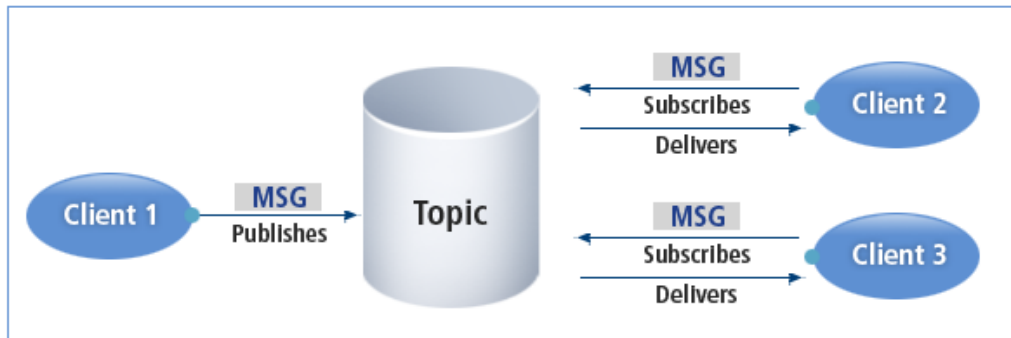
[図 2.32] HMS-キュー方式



- トピック方式(出版-購読型、Publish/Subscribe)

各メッセージは多数の消費者に送信されることができ、永続性のあるサブスクリプション (Durable subscription)によるメッセージの送信で信頼性が保証されます。

[図 2.33] HMS-Topic方式



参考

HMSの詳細については、『Tmax HMS ユーザガイド』を参照してください。

2.4. Tmaxの特徴

Tmaxは既存のマスター/スレーブ(Master/Slave)方式の代わりにPeer-to-Peer構造を採用し、各ノードにRACD(Remote Access Control Daemon)が存在します。また、ストリーム・パイプ(Stream Pipe)通信方式を使用してキューフル(Queue Full)を根本的に防止し、ネットワーク・プロセスでの行きすぎたトランザクションを遮断することで、メッセージ・キュー方式よりプロセス間の通信を安定的に維持することができます。こうしたTmaxの特長はメモリー・リソースの浪費を最小化し、迅速な障害復旧をサポートし、管理者が能動的に障害に対応できるようにし、卓越した安定性を提供します。

以下は、Tmaxの特徴です。

- X/Open、ISO-TPなど各種のDTP国際標準を100%遵守
 - 分散トランザクション・プロセッシングの国際標準であるX/Open DTP(Distributed Transaction Processing)モデルを遵守するので、アプリケーション・プログラム(AP)、トランザクション・マネージャー(TM)、リソース・マネージャー(RM)、通信リソース・マネージャー(CRM)などをベースに、互換性のあるAPIとシステム構造を提供します。
 - 国際標準化機構OSI(Open Systems Interconnection group)のDTPサービスに対する機能的な分散と機能構成要素間のAPI、システム・インターフェースに定義された分散トランザクションをサポートするための機能、そして分離された開放型システムに存在する多数のトランザクションを相互調整できるように基本ツールを提供します。
 - 分散環境で異機種間の透明な業務処理およびOLTP(On-Line Transaction Processing)をサポートします。
- プロテクション、マルチプレクシング(Protection、Multiplexing)の効率性の向上

ストリーム・パイプ方式のIPC(Inter Process Communication)の実装でプロテクション、マルチプレクシングの効率性を向上します。

- 多様なメッセージ・タイプおよび通信タイプの提供

- Integer、Long、Characterなどのメッセージ・タイプをサポートします。
- 同期(Synchronous)通信、非同期(Asynchronous Mode)通信、対話型(Conversational)通信、伝達型通信などの通信タイプを提供します。
- FDL(Field Definition Language)および構造体の配列(Structure Array)をサポートします。

- 障害対策(Fault Tolerance, Fail-Over)

- Peer-to-Peer方式のTPモニター構造をもちます。
- H/WおよびS/Wの障害対策をもちます。
- 多様な障害防止機能をサポートします。

- 拡張性(Scalability)

- クライアントが増加しても対応できるシステム活用性を提供します。
- CA(Client Agent)を利用した2ティア・モデルから3ティア・モデルへの効率的な移行が可能です。
- レガシー・システムに対する多様なプロトコルを提供します。
- WebTを利用したWeb環境にサービスを拡大して提供します。
- TCP/IP、SNA、X.25などレガシー・システムに対する多様なプロトコルを提供します。
- 多様なプロセスの制御方式をサポートします。

- 柔軟性(Flexibility)

- 多様なプロセス制御方式をサポートします。
- 使用環境別の特性を考慮した機能要求があるとき、追加機能をサポートします。

- 優れた性能

- システム・リソースの効率的な活用が可能です。

- 単純、明瞭な開発生産性の向上のためのAPIを提供します。
- ユーザー・フレンドリーで使い易いシステム管理(System Monitoring)機能を提供します。
- 多様なH/Wプラットフォームのサポート
 - IBM OS 390、ほとんどのUNIX系列の運用体制、Linux、Windows NTなどのプラットフォームをサポートします。
- PowerBuilder、Delphi、Visual C/C++、Visual Basic、.NET(C#, VB)などのすべての4GLをサポート

2.5. Tmax導入時の考慮事項

2.5.1. システム環境

サポート環境

以下は、Tmaxシステムの基本環境を示した表です。

区分	説明
プロトコル	Application API : XATMI、TX
	Integrating API : XA
	Network : TCP/IP、X.25、SNA(LU 0/6.2)
OS	サーバー : All UNIX、NT、Linux
	クライアント : All UNIX、Windows、MS-DOSなど
対応プラットフォーム	IBM OS 390、UNIX、LinuxおよびNTをサポートするすべてのH/W
サーバー用の開発言語	C、C++、COBOL
クライアント用の開発言語	C、C++および多様な4GL(Power Builder、Delphi、Visual C/C++、Visual Basic、.NET(C#, VB)など)に対するインターフェースのサポート
DBMS対応	Oracle、Informix、Sybase and DB2 (UDB)

サーバー要件

以下は、Tmaxの導入時に考慮する必要のあるサーバーの要件を示した表です。

区分	説明
H/W	Memory: 0.50MB

区分	説明
	Disk(Tmaxクライアント) : 0.277MB (Include-83KB、DLL-86KB、Type Compiler-108KB)
S/W	IBM OS 390、UNIX、Linux、NT
	CまたはC++、COBOL Compiler
ネットワーク・プロトコル	TCP/IP

クライアント要件

以下は、Tmaxの導入時に考慮する必要のあるクライアントの要件を示した表です。

区分	説明
H/W	Memory : 0.537MB+0.2~0.5MB/application
	Disk(Tmaxクライアント) : 0.277MB (Include-83KB、DLL-86KB、Type Compiler-108KB) ただし、Power Builderの場合、203KB(DLL、PBD)を追加
S/W	Linux、NT、Windows(2000、XP)、MS-DOS、UNIX
	Power Builder、Delphi、Visual Basic、Visual C++、C、.NET(C#、VB)
ネットワーク・プロトコル	TCP/IP

2.5.2. 考慮事項

Tmaxを導入する際、機能、性能および安定性など、多様な側面から考慮すべき事項があります。

以下は、機能面で考慮する必要のある事項についての説明です。

考慮事項	細部事項
TPモニターの基本機能	<ul style="list-style-type: none"> – プロセス管理 – 分散トランザクションのサポート – 負荷分散 – クライアント/サーバー間の多様な通信 – 障害対策(すべてのサーバー障害への対応および防止機能) – 異機種DBMSのサポート
付加機能	<ul style="list-style-type: none"> – セキュリティ機能

考慮事項	細部事項
	<ul style="list-style-type: none"> – システム管理 – ネーミング・サービス(Naming Service) – BPクライアントのマルチプレキシング(Multiplexing)機能 – システム特性に合ったセキュリティー機能 – 構造体の配列通信のサポート – ホストとの連携時の対応能力

以下は、機能以外に考慮する必要のある事項についての説明です。

項目	考慮事項
性能	<ul style="list-style-type: none"> – 平均処理時間または時間当たりの最大処理件数 – リソース使用度合い
安定性(障害対策)	<ul style="list-style-type: none"> – 顧客会社の障害発生頻度 – 障害が発生される場合の対応能力および掛かる時間
教育およびテクニカル・サポート	<ul style="list-style-type: none"> – エンジニアの技術レベル – 教育およびコンサルティングのサポート(システム設計段階のコンサルティング、アプリケーション開発専門教育(OS、ネットワーク、TPモニター))
リスク管理	<ul style="list-style-type: none"> – 新システム構築時の試行錯誤の最小化 – 新技術を利用したシステム構築 – 構築対象業務への理解度
開発および運用利便性	<ul style="list-style-type: none"> – クライアント/サーバー開発ツールのサポート – 開発用ドライバーの提供 – システム統計資料のモニタリング – TPモニター環境の動的変更 – システム運用の利便性 – 報告書機能の提供

項目	考慮事項
ユーザー満足度	<ul style="list-style-type: none"> – 機能への満足度 – テクニカル・サポートおよび教育への満足度 – 製品の柔軟性への満足度
互換性	<ul style="list-style-type: none"> – 国際標準の遵守の有無(X/Open DTP、OSI-TP) – 特定のH/WおよびDBMSとの独立性
会社規模	<ul style="list-style-type: none"> – 資本金の規模 – 従業員数 – 売上高 – 今後の成長性
その他	<ul style="list-style-type: none"> – 技術移転 – バージョン・アップ計画

第3章 WebTの紹介

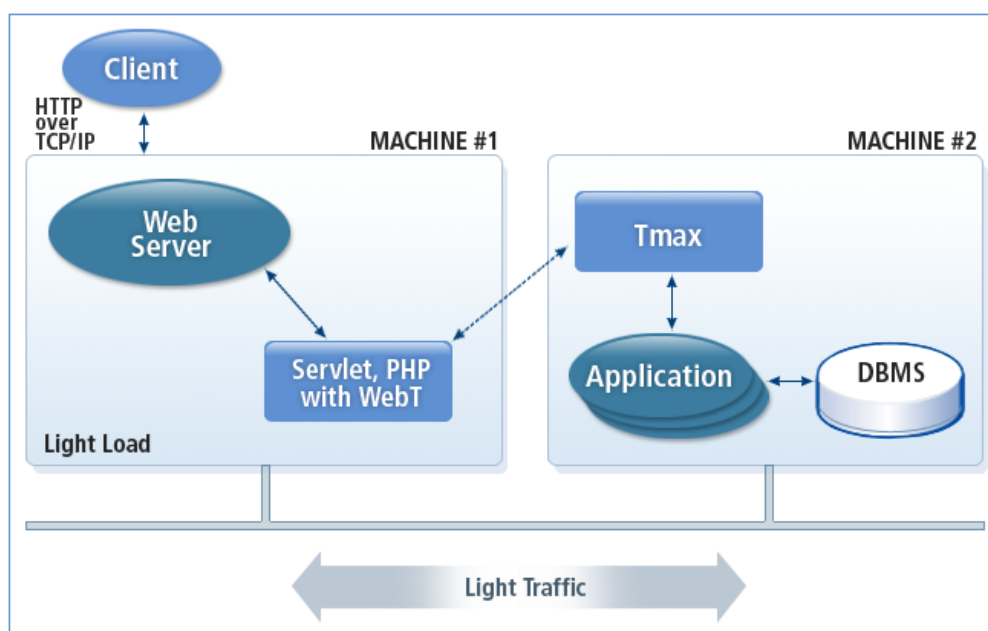
本章は、TmaxとJavaアプリケーション・プログラムとのトランザクション・サービスをサポートするために提供されるWebTについての基本機能と役割について説明します。

3.1. 概要

WebT(Web Transaction)は、クライアント/サーバー環境のミドルウェア製品であるTmaxサーバーとJavaアプリケーション・プログラム間のトランザクション・サービスをサポートするプログラムで、APIライブラリー形式でデPLOYされます。WebTは、JEUSを始めとするWeb基盤環境のWAS(Web Application Server)製品で活用されます。Tmaxのトランザクション処理と負荷調節機能を利用し、Web環境で動的データサービスを提供できるように設計されています。

以下の図は、WebTとTmaxサーバー間におけるサービスの手順を示しています。クライアントから要求を受けると、WebTモジュールを通じてTmaxのサーバー・プログラムが実行され、サービスが行われます。

【図 3.1】 WebTとTmax間のサービス・フロー



3.2. WebTConnectionPool

WebTは、Tmaxの接続を効率的に管理するため、WebTConnectionPoolクラスを提供します。WebTConnectionPoolは、Tmaxサービスを要求する度に接続オブジェクトを新しく作成するのではなく、以

前に使用した接続オブジェクトを再使用します。そのため、Tmaxサーバーにネットワーク接続を設定および終了するのに消費されるリソースおよび時間を節約できます。

WebTConnectionPoolは1個以上のWebTConnectionGroupで構成され、各WebTConnectionGroupは1個のTmaxサーバーと接続されます。クライアント・プログラムでは、WebTConnectionGroupのグループ名を使用してTmaxサーバーに接続できます。

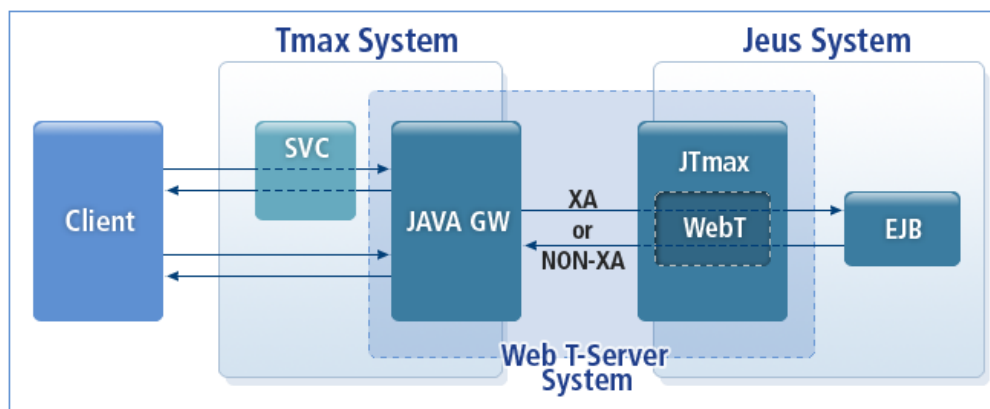
WebTモジュールをJEUSと連動させた場合、JEUSコンテナでWebTConnectionPoolを管理します。JEUSコンテナは、WebT.propertiesの環境設定ファイルまたはJEUSの環境設定ファイルを利用してコネクション・プールを作成します。また、ユーザーがgetConnectionを実行後に返していないコネクションを自動でコネクション・プールに返します。

3.3. WebT-Serverシステム

WebT-ServerシステムはTmaxシステムとJEUSの間に存在し、TmaxクライアントがJEUSのEJBサービスを呼び出せるようにします。

以下の図は、WebT-Serverシステムを通じてEJBサービスが呼び出される手順です。

【図 3.2】 WebT-Serverシステム



WebT-Serverシステムは、以下のモジュールで構成されています。

- JAVA GW
TmaxがJEUSに送信するサービス要求を処理します。
- JTmax
JEUSでTmaxのサービス要求を受信するデーモンです。
- WebT Library
TmaxとJEUSが送受信するデータを処理します。
- その他のユーティリティー

webtutil.jarは、jeus.jarとjeusutil.jarでWebTを使用するためのクラスを別途パッケージングしたファイルで、JEUSがインストールされていないマシンでWebTを起動するためのライブラリーです。

第4章 Tmaxアプリケーション

本章では、Tmaxアプリケーションを開発するために熟知する必要がある基本概念と、クライアント/サーバー・プログラムのフローおよび構成、プログラム開発のためのAPとエラーについて説明します。

4.1. アプリケーションの構成

Tmaxアプリケーションはクライアント・プログラムとサーバー・プログラムで構成されます。クライアント用のプログラムはユーザー・インターフェースを担当(Presentation logic)し、サーバー用のサービスルーチンは業務処理およびデータベースのアクセス・ロジック(Access Logic)を実装します。

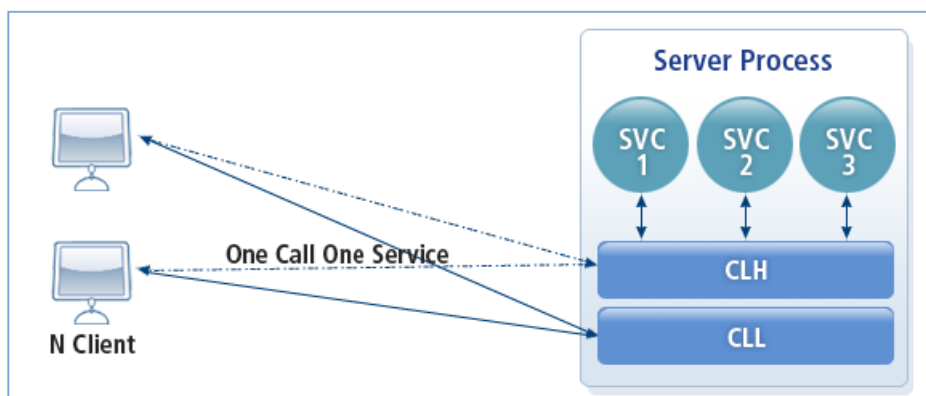
クライアントから要求が発生するとサーバーでその要求を処理します。クライアントがN個存在する場合も1つの呼び出しに対しては1つのサービスが起動します。クライアント要求はCLLに先にアクセスし、要求に対してtpcallが発生します。CLLは当該サービスに該当する要求を実行、実行が完了すると要求を返却する役割をします。クライアント・プログラムはTmaxに連結されるとき、Tmaxのホスト・アドレスとポートなど、連結のための事前情報の設定が必要です。この情報は、.profileまたはtmax.envファイルに保存され、TMAX_HOST_ADDR、TMAX_HOST_PORT項目に設定されます。

サーバー・プログラムはTmaxで提供するmain()と開発者が開発するサービス・ルーチンで構成されます。

Tmaxはクライアントとサーバーの連結データ要求などの処理のために多様なAPI関数を提供します。Tmax APIは分散トランザクション処理の国際標準であるX/Open DTPモデルを遵守します。アプリケーションを開発するとき、サービス・ルーチンのみ開発します。

以下は、Tmaxアプリケーションの構成を示した図です。

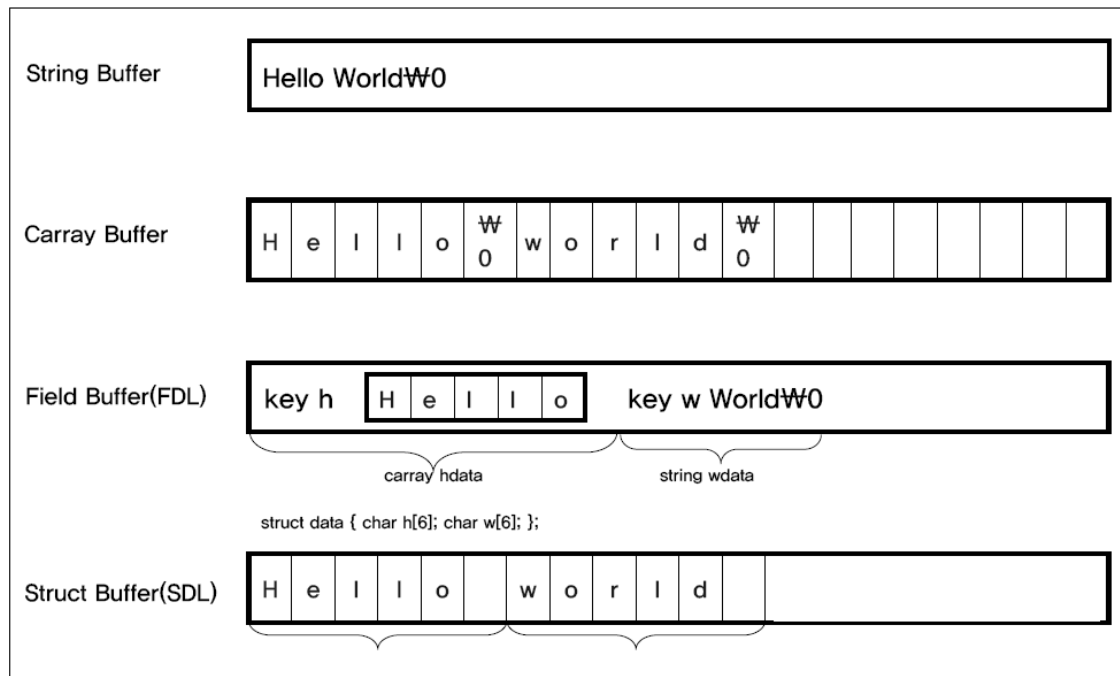
[図 4.1] Tmaxアプリケーションの構成



4.2. バッファ・タイプ

クライアントからサーバーにサービスを要求する場合、通信用として以下の通信バッファを使用します。

[図 4.2] Tmaxの通信バッファのタイプ



- STRINGバッファ

NULL値で終わる文字列で、別途バッファの長さを明示しなくてもいいです。プラットフォームの違いによる問題は発生しません。

- CARRAY、X_OCTETバッファ

長さが指定されているバイト列で構成されたバッファで、通常バイナリタイプのデータを送るときに使われます。データを交換する場合、必ず長さを明示する必要があります。プラットフォームの違いによる問題は発生しません。

- STRUCT、X_C_TYPEバッファ

C言語の構造体をデータ通信に使う場合に使用します。構造体のメンバーとしてはすべての原始タイプを使用でき、宣言された構造体自体もメンバーとして使用できます。また、構造体の配列も使用可能です。

- X_COMMONバッファ

メンバー・タイプがchar、int、longに限定されたC構造体です。

- FIELDバッファ

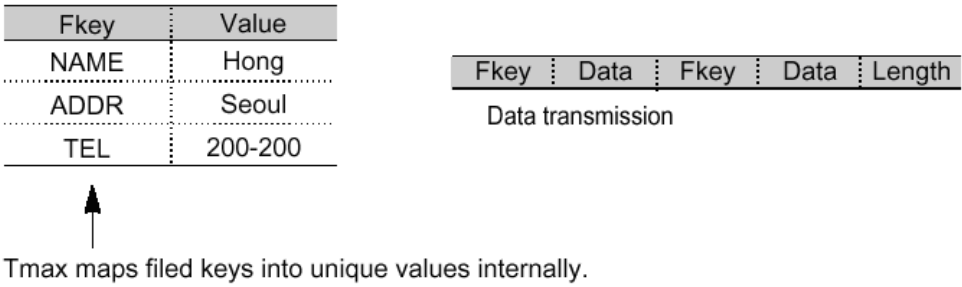
フィールド・キーとデータ値を1組で管理するデータ・バッファで、すべての原始タイプのデータを格納できます。交換されるデータ型が流動的である場合に使用し、多様な方法のデータ・アクセスおよび変換APIを提供します。

FDL (Field Definition Language)

一般的な構造体とは違い、フィールド・キー・バッファ(Field Key Buffer)を使用して必要な情報のデータのみを操作して処理可能です。

各フィールド・キーに対する名前(NAME、ADDR、TELなど)、番号、タイプをフィールド・バッファ・ファイル<XXX.f>に記述します。FDLCを利用して<XXX.f>をコンパイルすると<XXX_fdl.h>にマッピングされたファイルを生成させます。コンパイルされるとき、インクルード(include)された<XXX_fdl.h>を参照してユーザーが選択したフィールド・キーと値のみを操作できます。

[図 4.3] FDLの保存方式



4.3. クライアント/サーバー・プログラム

クライアント用のプログラムはユーザー・インターフェースを担当(Presentation logic、プレゼンテーション・ロジック)し、サーバー用のサービス・ルーチンは業務処理およびデータベースのアクセス・ロジックを実装します。

4.3.1. クライアント・プログラム

クライアント・プログラムはユーザーの入力(input)を受けてサーバーにサービスを要求し、サーバーから応答を受けてユーザーにサービス応答を出力(output)するプログラムです。

クライアント・プログラムのフロー

以下は、クライアント・プログラムのフローです。

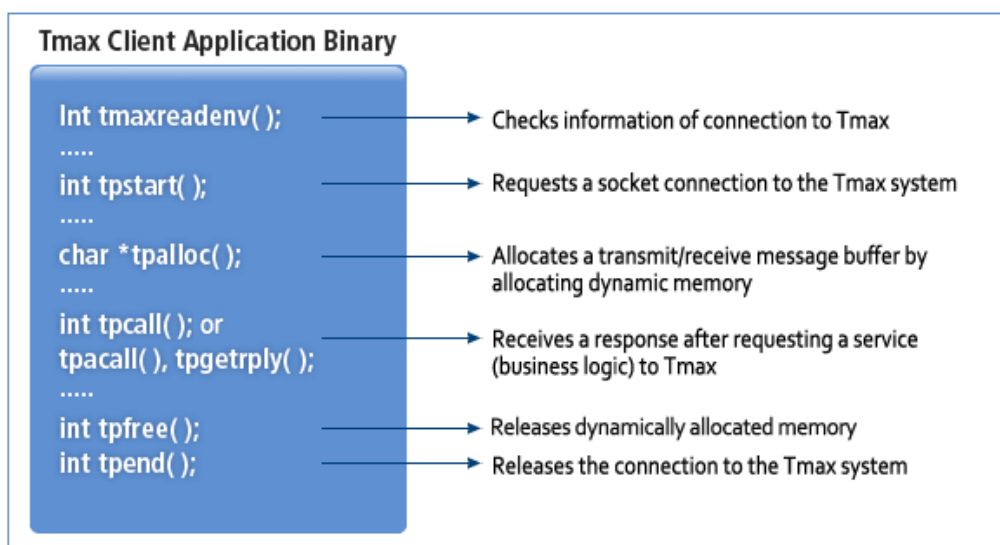
```
main()  
{
```

```

Tmax接続
送受信メッセージ・バッファの割当
クライアント業務ロジックの作成(ユーザー要求事項の入力を受けて送信メッセージ・バッファに保存)
サービス要求および応答(送信メッセージ・バッファをTmax CLHに送信、
    Tmax CLHを使って受信メッセージ・バッファに応答データを受信)
    クライアント業務ロジックの作成(応答データをユーザーに表示)
送受信メッセージ・バッファの解除
Tmax連結の解除
}

```

[図 4.4] クライアント・プログラムのフロー



以下は、クライアント・プログラムの主要関数に対する説明です。

- Tmax通信環境関数

関数	説明
tmaxreadenv()	<p>Tmaxクライアント・プログラムを実行するときに基本的に参照することになる一連のTmax環境変数などを特定ファイルに記述することができます。tmaxreadenv() APIを呼び出すプログラムを作成するとき、特定ファイルのパス名およびファイル名を設定することになると、当該クライアント・プログラムを実行するとき、設定されたファイルのファイルポインターを取得した後、内容をパースします。</p> <p>パースした内容のうちTMAX_HOST_ADDR、TMAX_HOST_PORTなどのTmax環境変数は、クライアント・プログラムが実行されるとき、確保されたメモリー空間内にロードされ、関連API関数の呼び出しに利用されます</p>

- クライアントとサーバー連結関数

関数	説明
tpstart()	CLLプロセスでソケット連結を要求し、取得(accept)された連結をCLHIに伝達します。 tpstart()を呼び出すとき、TMAX_HOST_ADDR、TMAX_HOST_PORTに設定された値を使用します
tpend()	CLLプロセスでソケット連結を終了します

- 通信バッファと解除関数

関数	説明
tpalloc()	Tmaxを使ったクライアントとサーバー間のデータ送受信時に使用するメモリーを動的に割り当てます。データを送受信するとき、必要な長さ情報を予測し、そのサイズ分だけ事前に割り当てる必要があります。 動的に割り当てられたメモリーは必ず明示的に解除しなければなりません
tpfree()	tpalloc()を使って動的割当メモリーを明示的に解除(返却)します。割り当てられたメモリー・バッファは返却しなければならず、そうでなければガーベジ(メモリー漏れ、memory leakage)が発生します

- クライアント同期通信関数

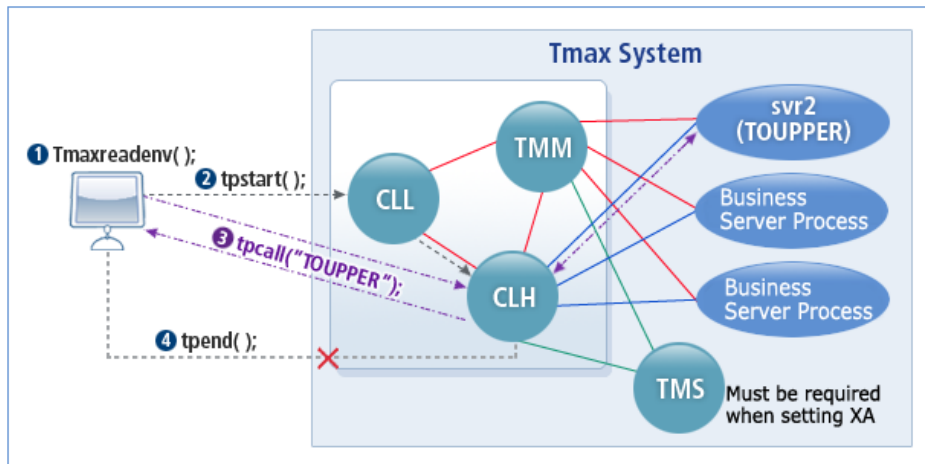
関数	説明
tpcall()	CLHIにサービスを要求して送信データを伝達し、CLHIはクライアントの要求サービスを確認した後、サーバー・アプリケーション・プロセスで伝達します。 クライアントはサービス要求に対する応答が到着するまで待機します

- クライアント非同期通信関数

関数	説明
tpacall()	CLHIにサービスを要求して送信データを伝達し、CLHIはクライアントの要求サービスを確認した後、サーバー・アプリケーション・プロセスで伝達します。 クライアントはサービス要求に対する応答状態とは関係なく、tpacall()の後ロジックを実行します
tpgetrply()	tpacall()に対するパラメーター(cd)値を使ってCLHで応答データを要求します。 CLHIにすでに当該クライアントに送る応答データがあればそれをクライアントに即刻伝達します。 パラメーターのうちフラグ(flags)設定値によって応答データが来るまで待機するかまたは待機せずに即刻応答受信エラーをクライアントに伝達します

以下は、クライアント・プログラムの主要関数プロセスに対する図です。

[図 4.5] クライアント・プログラム関数プロセス



参考

各関数に対する詳細と使用法については、『Tmax アプリケーション開発ガイド』および『Tmax リファレンスガイド』を参照してください。

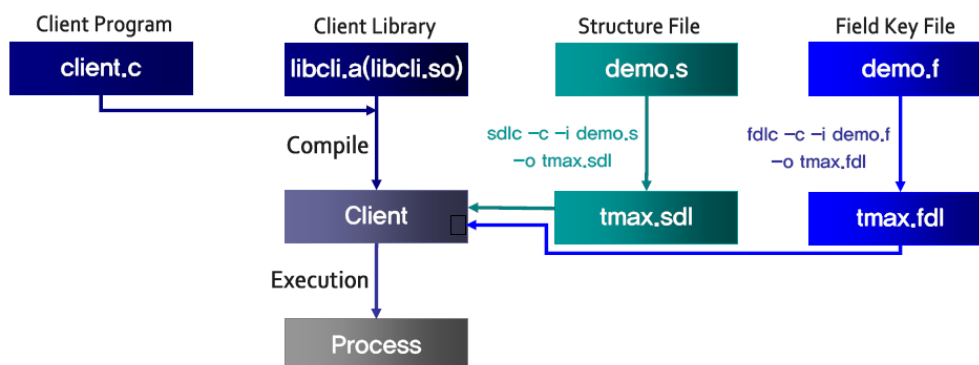
クライアント・プログラムの構成

クライアント・プログラムのコーディングが完了するとコンパイルして実行ファイルを作成します。

クライアント・プログラムをコンパイルするには、開発者が作成したクライアント・プログラム、Tmaxクライアント・ライブラリー、構造体バッファーを使用する場合は構造体ファイル、そしてフィールドテーブルが定義されたフィールド・キー・バッファー・ファイルが定義されている必要があります。

以下は、クライアント・プログラムの構成に対する図です。

[図 4.6] Tmaxクライアント・プログラムの構成



- クライアント・プログラム

開発者が作成したクライアント・プログラムです。

- Tmaxクライアント・ライブラリー(libcli.a / libcli.so)

Tmaxが提供するライブラリーでクライアント・プログラムを開発するとき使用する関数のオブジェクト・コードです。

- 構造体ファイル

クライアント・プログラムで構造体(STRUCT、X_C_TYPE、X_COMMON)を使用した場合、<ファイル名.s>で終わる構造体ファイルが必要です。構造体ファイルを**sdlc**コマンドを利用して事前にコンパイルします。コンパイルすると、構造体の各データを標準通信型に変換するために必要な情報を格納しているバイナリ形態のファイルが生成されます。これはクライアント・プログラムを実行するとき標準通信型でデータを送受信するために使われます。

- フィールド・キー・ファイル

フィールド・キー構造を使用した場合、<.f>で終わるフィールド定義ファイルが必要です。ファイルを**fdlc**コマンドを利用してコンパイルすると、その際フィールド・キー・バッファ・ファイルはキーマッピングを利用して<フィールド・キー・バッファ名_fdl.h>を作成し、プログラム実行時に使うことになります。既存の構造体ファイルとはちがってユーザーが必要とするフィールド・キー値のみを操作して送信でき、リソースの漏洩を削減できます。ただし、キー・マッピングをする過程でオーバーヘッドも発生し得るので注意が必要です。

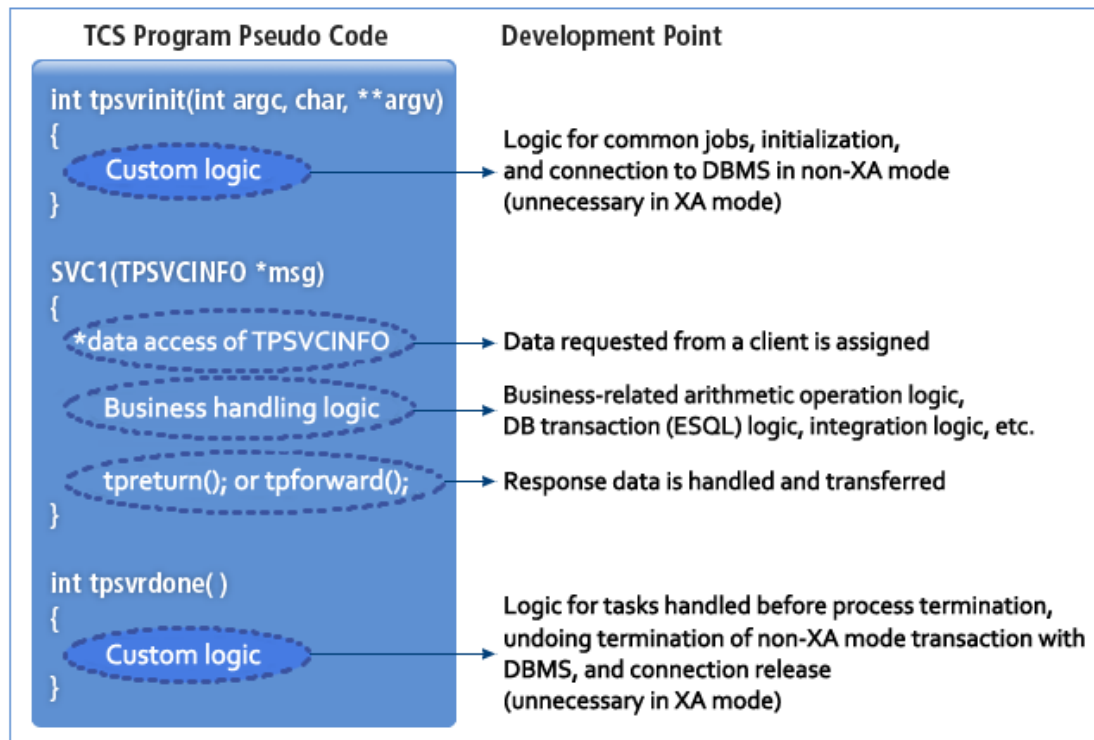
4.3.2. サーバー・プログラム

サーバー・プログラムはユーザーの要求を受けて処理し、その応答をクライアントに返却します。

サーバー・プログラムのフロー

以下は、サーバー・プログラムのフローです。

【図 4.7】 サーバー・プログラムのフロー



```
int tpsvrinit(int argc, char **argv)
{
    サーバー初期化作業 (DBMSとのNon-XAモードの接続)
}

SVC_NAME(TPSVCINFO *msg)
{
    DBトランザクション処理 (ESQL)、業務処理ロジックを含む
    クライアントに応答結果を転送
}

int tpsvrdone()
{
    サーバー終了直前の作業 (DBMSとのNon-XAモードの連結解除)
}

void tpsvctimeout()
{
    SVCTIME項目と関連され、サービス・タイムアウト・ハンドラーによって呼び出される
}
```

以下は、サーバー・プログラムの主要関数についての説明です。

- サーバー初期化関数

関数	説明
tpsvrinit()	<p>サーバー・アプリケーション・プロセスが起動されるとき、初めて呼び出されます。</p> <p>tpsvrinit()の標準入力はTmax環境ファイルの設定によって行われます。tpsvrinit()は開発者が再定義して使用します。再定義しないとTmaxサーバー・ライブラリーにある基本ロジックが適用されたtpsvrinit()が呼び出されます</p>

- サーバー終了時の作業関数

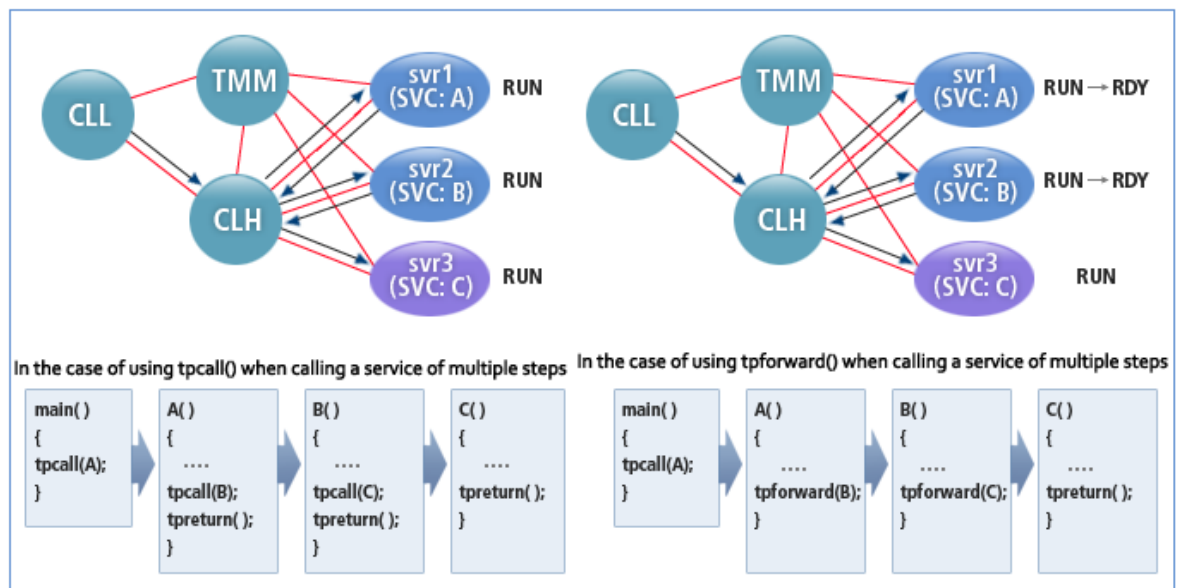
関数	説明
tpsvrdone()	<p>サーバー・アプリケーション・プロセスが正常終了する直前に呼び出されます。</p> <p>tpsvrdone()は開発者が再定義して使用します。再定義しないとTmaxサーバー・ライブラリー内にある基本ロジックが適用されたtpsvrdone()が呼び出されます</p>

- Tmax応答関数

関数	説明
tpreturn()	<p>CLHIにサービス・ロジックの終了を知らせて応答データをわたします。CLHIはサービスを呼び出した側の情報を確認して応答データを即刻送信します。</p> <p>応答データが保存されたバッファ・ポインターが指す動的割当メモリーを返却(free)すると、サービスを含むサーバー・アプリケーションはプロセスの動作を止めて次の応答を受け取る準備をします</p>
tpforward()	<p>CLHIにサービス・ロジックの終了を知らせ、SVCに送りたいデータを渡します。CLHIは渡したいサービスを含むサーバー・プロセスが使用可能かどうかを確認した後、データを即刻渡します。</p> <p>応答データのポインターが指す動的割当メモリーを返却(free)するとサービスを含むサーバーアプリケーションはプロセスの動作を止めて次の応答を受け取る準備をします</p>

以下は、サーバー・プログラムの主要関数のプロセスを示した図です。

[図 4.8] tpcallとtpforwardの比較

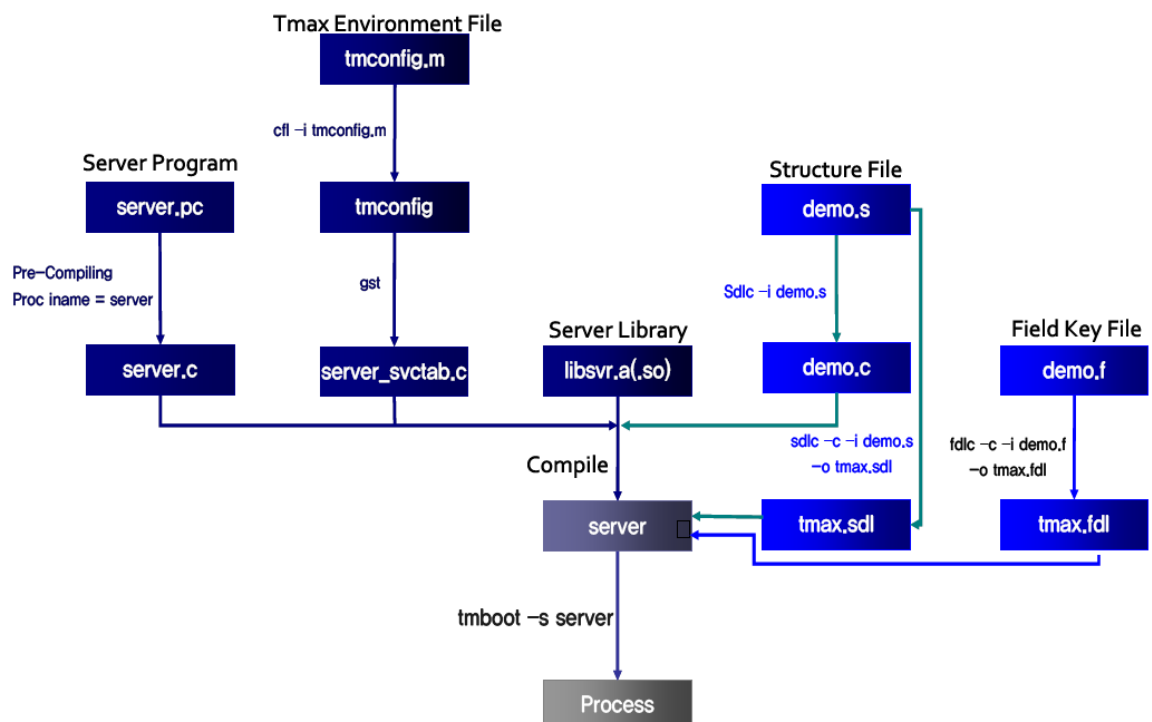


サーバー・プログラムの構成

サーバー・プログラムはTmaxで提供するmain()とアプリケーション開発者が作成したサービス・ルーチン (Service Routine) で構成されます。

以下は、サーバー・プログラムの構成を示した図です。

[図 4.9] Tmaxサーバー・プログラムの構成



- サーバー・プログラム

開発者が作成したサービス・ルーチンです。クライアントの要求を処理します。

SQL文を使用した場合、当該データベース・ベンダーが提供するツールを利用してプリコンパイルを終わらせる必要があります。

- Tmaxサーバー・ライブラリー(libsvr.a / libsvr.so)

Tmaxで提供するサーバー・ライブラリーで、サーバーmain()とtpsvrinit()、tpsvrdone()、そして各種Tmax関数があります。

- サービス・テーブル

サーバーごとに提供されるサービス名が羅列されたファイルで、Tmax環境ファイルを参照して作成します。

サービス・テーブルはサービスを実行するとき実際にサーバー内で当該サービス・ルーチンの位置を探すために使われ、システム管理者が提供します。これに対する詳細については、『Tmax 運用ガイド』を参照してください。

- 構造体バイナリ・テーブル(SDLFILE)

構造体バッファー(STRUCT、X_C_TYPE、X_COMMON)を使用した場合、それを定義する<xxxx.s>形式の構造体ファイルが必要です。

構造体ファイルは標準通信タイプ(構造体ファイル名_sdl.c)と構造体ヘッダー・ファイル(構造体ファイル名_sdl.h)があります。構造体ファイルは**sdlc** -cコマンドを利用してコンパイルし、その結果、構造体のメンバーを標準通信型に変換するために必要なバイナリ・テーブル・ファイルが作成されます。これはサーバー・プログラム実行時に構造体型のデータを標準通信型に変換および逆変換するために使われます。

構造体を使用しない場合には、\$TMAXDIR/lib/sdl.oをアプリケーション・サーバーと一緒にコンパイルする必要があります。

- フィールド・バッファー・バイナリ・テーブル(FDLFILE)

フィールド・バッファーを使用した場合、<xxxx.f>形式で定義されたフィールド・バッファー・ファイルが必要です。このファイルは**fdlc**コマンドを利用してコンパイルし、その結果、バイナリ・テーブル・ファイルとヘッダー・ファイル(xxxx_fdl.h)が作成されます。バイナリ・テーブル・ファイルはフィールド・キーとデータを対応させる機能を行い、ヘッダー・ファイルはフィールド・キーとフィールド・キーの名前を対応させる機能を行います。既存の構造体ファイルとは違ってユーザーが必要とするフィールドのキーのみを操作して送信できるので多様なタイプのデータを使用する場合、リソースの浪費を削減できます。一方、データ値とフィールド・キー値と一緒に管理するので、多くのフィールドを使用しないと逆効果が発生してしまう恐れがあります。

- 構造体-標準バッファーの変換/逆変換プログラム

サーバー・プログラム内で構造体バッファを使用するには**sdlc**コマンドで作成された構造体-標準バッファの変換/逆変換プログラム(xxxx_sdl.c、xxxx_sdl.h)をコンパイルして一緒にリンクする必要があります。構造体を使用しないときは、TMAXDIR/lib/sdl.oをリンク付けるようにします。

4.4. システム環境ファイル

システム環境ファイルはTmaxシステムに必要な情報を格納するファイルです。Tmax環境ファイルはTmaxシステムの構成を設定し、Tmax管理者が作成します。このファイルはサービス・テーブルの作成とTmaxシステムの起動に使われます。

環境ファイルは8つのセクションで構成されます。

セクション	説明	必須状態
DOMAINセクション	1つの独立的なTmaxシステムの全体環境を定義します	必須
NODEセクション	ドメインを構成する各ノードに関係する環境を定義します	必須
SVRGROUPセクション	サーバー・グループおよびデータベースに関連する事項を定義します	必須
SERVERセクション	サーバーに関連する事項を定義します	必須
SERVICEセクション	サービスに関連する事項を定義します	必須
GATEWAYセクション	ドメイン間のゲートウェイに関連する事項を定義します	選択
ROUTINGセクション	データ依存ルーティングに関連する事項を定義します	選択
RQセクション	信頼性キューに関連する事項を定義します	選択

- 各セクションの名前はアスタリスク(*)ではじめます(例: *DOMAIN, *NODEなど)。
- 各セクションの名前とセクションの下位オブジェクト名は必ずラインの最初のコマから始めます。
- 1つの下位オブジェクトに対する定義はカンマ(,)で区分します。

一般にテキスト・ファイルで環境ファイルが作成され、**cfl**コマンドでコンパイルします。

```
cfl -i Tmax環境ファイル名
```

以下は、Tmax環境ファイルの例です。<>に積載されている部分は適切な値に修正する必要があります。

```
*DOMAIN
<resrc_name> SHMKEY = <UNIQUE IPCKEY>,
               MAXUSER = <256>,
               TPORTNO = <8999>
```

```

*NODE
<uname>      TMAXDIR = <TMAX installed directory>
              APPDIR  = <APPLICATION directory>
              PATHDIR = <PATH directory>

*SVRGROUP
<svg_name>    NODENAME = <uname>,
              DBNAME  = <ORACLE>,
              OPENINFO = "ORACLE_XA+Acc=P/tmaxsoft/tmaxsoft+SesTm=60"

*SERVER
<svr_name>    SVGNAME = <svg_name>,
              MIN     = <5>,
              MAX     = <10>

*SERVICE
<svc_name>    SVRNAME = <svr_name>

```

参考

各セクションに対する詳細については、『Tmax 運用ガイド』を参照してください。

4.5. API

プログラム開発時にはAPIの原型が定義されているヘッダー・ファイルをインクルード(include)して使用する必要があります。APIはクライアント/サーバー・ライブラリーに実装されています。APIの詳細については、『Tmax アプリケーション開発ガイド』および『Tmax リファレンスガイド』を参照してください。

4.5.1. Tmaxの標準API

X/Open ATMI

X/Open ATMI(X/Open Application Transaction Monitor Interface) APIはX/Open DTPモデルで標準として規定したATMIインターフェースで、アプリケーション・プログラムとTPモニター間の通信方式の1つとして使用します。

関数はatmi.hに定義されており、バッファ関連関数、サービス要求および応答関連関数、対話型モード関連関数、サービス終了関連関数に区分できます。

- バッファ割り当ておよび解除関連関数

関数	説明
tpalloc()	データを送受信するバッファを割り当てる関数です
tprealloc()	バッファのサイズを変更する関数です
tpfree()	割り当てられたバッファを解除する関数です
tpypes()	バッファのサイズと形式に対する情報を提供する関数です

- サービス要求および応答関連関数

関数	説明
tpcall()	サービスを要求して応答が来るまで待機する関数です
tpacall()	サービスを要求し、ほかの処理を実行する中で、tpgetrply()関数が呼び出されると、処理結果を受信する関数です
tpcancel()	サービス要求に対する応答を取り消す関数です
tpgetrply()	tpacall()呼び出しに対する応答を受信する関数です

- 対話型モード関連関数

関数	説明
tpconnect()	対話型モードでメッセージ送受信のための連結関数です
tpdiscon()	対話型モードでサービスとの連結を異常終了する関数です
tprecv()	対話型モードでメッセージを受信する関数です
tpsend()	対話型モードでメッセージを送信する関数です

- サービス終了関連関数

関数	説明
tpreturn()	サービス要求に対する応答をクライアントに送り、サービス・ルーチンを終了する関数です

X/Open TX API

X/Open TX APIはアプリケーション・プログラムとTPモニター間のトランザクションに関連して通信する方式を提供します。関数はtx.hに定義されており、トランザクション管理関数で構成されます。

以下は、TX API一覧です。

- トランザクション関連関数

関数	説明
tx_begin()	トランザクションを開始する関数です
tx_commit()	トランザクションをコミットする関数で、結果を保存します
tx_rollback()	トランザクションを元の状態に復旧する関数です
tx_open()	内部的に作動する関数で、リソース・マネージャーを開始する関数です
tx_close()	内部的に作動する関数で、リソース・マネージャーを終了する関数です
tx_set_transaction_timeout()	トランザクションが終了される必要のある時間制限を設定する関数です
tx_info()	グローバル・トランザクションの情報を返却する関数です
tx_set_commit_return()	グローバル・トランザクションを許可する時点を設定する関数です
tx_set_transaction_control()	トランザクション完了後に自動で次のトランザクションを開始する関数です

4.5.2. 非標準API

Tmax ATMI

Tmax ATMI関数は非要求メッセージ処理、RQ関連、エラー設定、タイムアウト設定などに使われる関数で、tmaxapi.hに定義されています。定義されたAPIは非標準インターフェースで開発者の開発生産性を向上させるために開発され、開発者によって作成されるアプリケーション・プログラムとTPモニター間の通信方式の1つとして使用可能です。

以下は、tmaxapi.hに定義されている非標準API一覧です。

- 非要求データ関連関数

関数	説明
tpbroadcast()	非要求データをシステムに登録されたクライアントに一方向的に伝達する関数です
tpsetunsol()	非要求メッセージを処理する関数を指定する関数です
tpgetunsol()	非要求データを受信する関数です
tpsetunsol_flag()	非要求データ受信フラグを設定する関数です
tpchkunsol()	非要求データの到着を確認する関数です

- エラー関連関数

関数	説明
tpsterror()	エラーの内容を文字列(String)形式で出力する関数です
Userlog()	先にエラーをバッファに記録(Log)する関数です

関数	説明
ulogsync()	ディスクのメモリー・バッファに「ulog」内容を保存する関数です
UserLog()	userlog()とulogsync()の機能が複合された関数です
gettperrno()	Tmaxシステムを呼び出すときに発生したエラー番号を返却する関数です
gettpurcode()	開発者が設定したurcodeを返却する関数です
tperrordetail()	Tmaxシステムを呼び出すときに発生したエラーに関連する情報を返却する関数です

- ソケット情報関連関数

関数	説明
tpgetpeer_ipaddr()	連結されたクライアントのIPアドレスを返却する関数です
tpgetpeername()	連結されたクライアント名を返却する関数です
tpgetsockname()	連結されたクライアントのソケット名を返却する関数です

- ブロック・タイムアウト設定関数

関数	説明
tpset_timeout()	ブロック・タイムアウト時間を設定する関数です

- 障害対策関連関数

関数	説明
tpbackup()	バックアップ・マシンに連結を結ぶ関数です

- 連結関数

関数	説明
tpstart()	Tmaxシステムと連結を開始する関数です
tpend()	Tmaxシステムと連結を終了する関数です

- RQ関連関数

関数	説明
tpenq()	RQにクライアントから送られた要求を保存する関数です
tpdeq()	RQにあるデータを読み込む関数です
tpqstat()	RQに保存されたデータの統計を要求する関数です
tpextsvcname()	RQに保存されたデータからサービス名を要求する関数です

- 環境変数関連関数

関数	説明
tmaxreadenv()	ファイルで環境変数を読み込む関数です
tpputenv()	環境変数を設定する関数です
tpgetenv()	環境変数値を返却する関数です

- Window操作関連関数

関数	説明
WinTmaxStart()	Tmaxシステムと連結する関数です
WinTmaxEnd()	Tmaxシステムとの連結を解除する関数です
WinTmaxSetContext()	Windowハンドルを指定する関数です
WinTmaxSend()	データを送信する関数です
WinTmaxAcall()	Window用の非同期関数です
WinTmaxAcall2()	データの受信をコールバック関数で処理するWindow用の非同期関数です

- その他の関数

関数	説明
tpscmt()	環境ファイルのトランザクション制御関連の設定を無効化する関数です
tpgetlev()	トランザクション・モードを確認する関数です
tpchkauth()	認証が必要かどうかを確認する関数です
tpgprio()	サービス要求の優先順位を確認する関数です
tpsprio()	サービス要求の優先順位を設定する関数です
tpsleap()	指定された時間内にメッセージ受信に待機する関数です
tp_sleep()	秒単位でデータ受信に待機する関数です
tp_usleep()	マイクロ秒単位でデータ受信に待機する関数です
tpschedule()	キューに積載されている業務を取り出してUCSに処理を割り当てる関数です
tpuschedule()	UCSサーバー・プロセスでデータの到着を入力した時間の間待機する関数です
tpsvrinit()	Tmaxサーバー・プロセスを初期化する関数です
tpsvrdone()	Tmaxサーバー・プロセス終了ルーチンを呼び出す関数です
tpsvctimeout()	UCSサーバー・プロセスをダウンする関数です
tmaxadmin()	サービス呼び出しの形でシステムを管理する関数です

以下は、atmi.hに定義されている非標準API一覧です。

- サービス終了関連関数

関数	説明
tpforward()	自分のサービス処理を終了し、クライアントの要求をほかのサービス・ルーチンに伝達します

- クライアント連結関連関数

関数	説明
tpstart()	クライアント・アプリケーションとTmaxを連結します
tpend()	クライアント・アプリケーションとTmaxの連結を解除します

FDL API

FDL(Field Definition Language)は非標準APIで、開発者の開発生産性を向上させるために開発されました。FDLはフィールド・キーと呼ばれるインデックスとデータが一緒に管理されるassociative-typed dataに該当します。FIELDバッファに関連するAPIはfbuf.hに定義されています。

こうしたデータはTmaxシステムで提供するバッファの一種であるフィールド・キー・バッファに乗せられ、そのバッファを操作できるように、以下のような関数を提供します。

- フィールド・キー関連関数

関数	説明
fbget_fldkey()	フィールド名に対するフィールド・キー値を返却します
fbget_fldname()	フィールド・キーの名前を返却します
fbget_fldno()	フィールド・キーからフィールド番号を読み込みます
fbget_fldtype()	フィールド・キーからフィールドタイプ(type)を読み込みます(整数値の返却)
fbget_strfldtype()	フィールド・キーからタイプに対するポインター値を読み込みます

- バッファ割当関連関数

関数	説明
fbisfbuf()	指定されたバッファがフィールド化されているかどうかを確認します
fbinit()	フィールド・キー・バッファで割り当てられたメモリー空間を初期化します
fbcalcsz()	フィールド・バッファのサイズを計算します
fballoc()	フィールド・キー・バッファを動的に割り当てます
fbfree()	フィールド・バッファを解除します

関数	説明
fbget_fsize()	バイト単位でフィールド・キー・バッファ・サイズを返却します
fbget_unused()	使用されないフィールド・バッファの空間を確認します
fbget_used()	使用中のフィールド・キー・バッファ空間をバイト数単位で返却します
fbrealloc()	バッファ・サイズを調整します

● フィールドアクセスおよび修正関数

関数	説明
fbput()	フィールド・バッファにフィールド・キー値を追加します
fbinsert()	フィールド・キーと位置を指定し、フィールド・バッファにフィールド値を保存します
fbchg_tu()	データを送信する前に指定されたフィールド・バッファを移動させます
fbdelete()	バッファのフィールド・データを削除します
fbdelall()	フィールドのすべての値を削除します
fbdelall_tu()	フィールド・キーの配列(fieldkey[])に羅列されたすべてのフィールドのデータを削除します
fbget()	バッファにあるフィールドの内容を探します
fbgetf()	フィールド・バッファにある指定されたフィールド・キー値を取得します
fbget_tu()	指定された位置にある特定フィールド・キーの値を取得します
fbnext_tu()	フィールド・バッファの特定フィールド・キーのフィールド値を順序どおり取得します
fbgetalloc_tu()	返却されたデータを保存するために他のバッファを内部的に割り当ててそのポインターだけをバッファに返します
fbgetval_last_tu()	フィールド・バッファの特定フィールド・キーのoccurrenceと最近のデータ値を取得します
fbgetlast_tu()	フィールド・バッファに指定されたフィールドの最近のエントリ・データを取得します
fbgetnth()	特定フィールド値を検索します
fbfldcount()	特定バッファに含まれたフィールドの数を返します
fbkeyoccur()	フィールド・キーに指定されたフィールド番号を返します
fbispres()	要求したデータがフィールド・バッファに存在するかを確認します
fbgetval()	要求したデータの長さとその位置のポインターを返します
fbgetvall_tu()	フィールドの実際値をlong型(type)で返します
fbupdate()	指定された位置のフィールド・バッファ内のフィールド・キーのフィールド値を更新します

関数	説明
fbgetlen()	フィールド・バッファ内の指定したフィールド・キーに該当される一番目の occurrence の値を返します

- 変換関数

関数	説明
fbtypecvrt()	データ型を変換します
fbputt()	新しいデータ値とデータ型をフィールド・バッファで付けます
fbget_tut()	指定された位置のフィールドデータを取得し、フィールド・キーの形式を指定します
fbgetalloc_tut()	変換されたデータを定義されたデータ型に変換して保存するため内部的に他のバッファを割り当てます
fbgetvalt()	変換された値のポインターを返します
fbgetvali()	integer 型のフィールド・データを返します
fbgetvals()	string 型のフィールド・データを返します
fbgetvals_tu()	指定された位置の string 型のフィールド・データを返します
fbgetntht()	変換された値を返します
fbchg_tut()	フィールド・バッファの特定開始時点のフィールド・キーの値を変更します

- バッファ演算関連関数

関数	説明
fbbufop()	1つのフィールド・バッファの内容を比較、コピー、移動、変更します

- 関数

関数	説明
fbbufop_proj()	フィールド・キーに該当されるバッファを変更します

- I/O 関連関数

関数	説明
fbbufop_proj()	フィールド・キーに該当されるバッファを変更します
fbread()	標準入出力ライブラリーと一緒に使用する関数で、ファイルからフィールド・バッファを読み込みます
fbwrite()	標準入出力ライブラリーと一緒に使用する関数で、ファイルに書き込みをします
fbprint()	標準入出力でバッファの内容を出力します

関数	説明
fbfprint()	フィールド・バッファの可能nデータをファイルStringで出力します

- エラー関連関数

関数	説明
fbsterror()	フィールド・バッファ操作時に発生したエラーのメッセージをString型で取得します
getfberno()	エラーが発生する場合、エラー番号を返します

- その他関数

関数	説明
fbmake_fidkey()	FDLFILEに記録しないものの、新しいフィールド・キーを自動的に生成します
fbftos()	フィールド・バッファに保存されたデータをC構造体(stname)に移動させます
fbstof()	C構造体で保存されたデータを構造体ファイルにマッピングされるフィールド・バッファに移動させます
fbsnull()	C構造体の指定されたフィールド・キーoccurrenceの構造体メンバー変数とフィールド・バッファがマッピングされるのがNULLかどうかを確認します
fbstelinit()	フィールド・バッファとマッピングされるC構造体メンバー変数をNULLに初期化させます
fbstinit()	フィールド・バッファとマッピングされるC構造体をNULLで初期化させます

4.6. エラー・メッセージ

4.6.1. X/Open DTP関連エラー

X/Open DTPで提供するインターフェースとTmaxシステムで提供する非標準インターフェースを使用する場合にエラーが発生すると当該状況に適切なエラー値がtperrnoと呼ばれるグローバル変数に設定されます。したがって開発者はエラーが発生するとtperrnoを確認して適切な後継措置をすることができます。

以下は、tperrnoエラー・メッセージ一覧です。tperrnoはエラー状況が発生する場合に設定されるグローバル変数です。

Error Message (tperrno)	説明
TPEBADDESC(2)	非同期型や対話型タイプで間違った記述子が使われる場合に発生します
TPEBLOCK(3)	ネットワーク・エラーの場合に発生します

Error Message (tperrno)	説明
TPEINVAL(4)	適切でないパラメータが入力された場合に発生します
TPELIMIT(5)	システムで提供する各種の限界値を超えた場合に発生します
TPENOENT(6)	サービスが提供されない場合に発生します
TPEOS(7)	システム的な問題で連結が不可能な場合に発生します
TPEPROTO(9)	プロトコルのエラーの場合に発生します
TPESVCERR(10)	アプリケーション・プログラミングの失敗によるTmaxシステム・バッファが損傷を受けた場合に発生します
TPESVCFail(11)	アプリケーション・プログラムのレベル・サービス・エラーの場合に発生します
TPESYSTEM(12)	Tmax内部エラー(ログ・メッセージの確認)の場合に発生します
TPETIME(13)	処理時間が超過(BLOCKTIME)された場合に発生します
TPETRAN(14)	トランザクションの失敗でトランザクションが取り消される場合に発生します
TPGOTSIG(15)	シグナルが発生された場合です
TPEITYPE(17)	登録されていない構造体の形式やフィールド・キーが使われた場合です
TPEOTYPE(18)	バッファ使用エラーまたはバッファ形式エラーが発生した場合です
TPEEVENT(22)	対話型モードでイベントが発生した場合です
TPEMATCH(23)	RQのtpdeq()関数を呼び出すとき、当該サービスに対する結果が存在しない場合に発生します
TPENOREADY(24)	サーバー・プロセスが動作が準備されていない場合に発生します
TPESECURITY(25)	セキュリティ上のエラーの場合に発生します
TPEQFULL(26)	サーバー・プロセスのキュー待機時間が超過された場合に発生します
TPEQPURGE(27)	キュー・ページによってキューから除去される場合に発生します
TPECLOSE(28)	Tmaxシステムとの接続が解除される場合に発生します
TPESVRDOWN(29)	アプリケーション・プログラムのエラーでサーバー・プロセスがダウンされた場合に発生します
TPEPRESVC(30)	以前のサービスの処置途中にエラーが発生した場合です
TPEMAXNO(31)	同時ユーザー数が限界値に達した場合に発生します

参考

エラーと対応方法についての詳細は、『Tmax アプリケーション開発ガイド』および『Tmax メッセージリファレンスガイド』を参照してください。

4.6.2. FDL関連エラー

FDLインターフェースに対するエラーはグローバル変数のfbernoに設定されます。そのため、開発者はエラーが発生するとfbernoを確認して適切な後継措置を取ることができます。本節ではFDL関連のエラー・メッセージについて説明します。

以下は、fberrorエラー・メッセージ一覧です。fberrorはエラー状況が発生する場合に設定されるグローバル変数です。

Error Message (fberror)	説明
FBEBADFB(3)	適切でないバッファを使用(フィールド・キー・バッファではない)した場合に発生します
FBEINVAL(4)	適切でないパラメータ値が使われた場合に発生します
FBELIMIT(5)	システムで提供する限界値を超えた場合に発生します
FBENOENT(6)	当該フィールド・キーがバッファに存在しない場合に発生します
FBEOS(7)	運用体制にエラーが発生した場合です。
FBEBADFLD(8)	適切でないフィールド・キーが使われた場合です
FBEPROTO(9)	プロトコルにエラーが発生した場合です
FBENOSPACE(10)	バッファ空間が足りない場合に発生します
FBEMALLOC(11)	メモリー割当にエラーが発生した場合です
FBESYSTEM(12)	システムにエラーが発生した場合です
FBETYPE(13)	タイプにエラーが発生した場合です
FBEMATCH(14)	一致する値が存在しない場合に発生します
FBEBADSTRUCT(15)	登録されていない構造体が使われた場合に発生します
FBEMAXNO(19)	存在しないエラー番号が使われた場合に発生します

参考

エラーと対応方法についての詳細は、『Tmax FDL リファレンスガイド』および『Tmax メッセージリファレンスガイド』を参照してください。

第5章 例

本章では、Tmaxシステムが提供する固有機能を使用するための例について説明します。

5.1. 通信タイプの例

本節では、Tmaxで使用する3種類の通信タイプである同期型、非同期型、会話型アプリケーションの簡単な例を通して、全体的な流れについて説明します。

5.1.1. 同期型通信

クライアントはSTRINGバッファに文字列をコピーしてサービスを呼び出し、サーバーのサービス・ルーチンはこの文字列を受信し、大文字列に変えて返すプログラムです。

プログラム構成

- 共通プログラム

プログラム・ファイル	説明
sample.m	Tmax環境設定ファイルです

- クライアント・プログラム

プログラム・ファイル	説明
sync_cli.c	クライアント・プログラムです

- サーバー・プログラム

プログラム・ファイル	説明
syncsvc.c	大文字に変えるサービス・プログラムです
Makefile	Tmaxが提供するMakefileを修正する必要があります

プログラムの特徴

- クライアント・プログラム

機能	説明
Tmax接続	基本接続(クライアント情報なし)
バッファ・タイプ	STRING
通信タイプ	tpcall()を利用した同期型通信

- サーバー・プログラム

機能	説明
サービス	TOUPPERSTR
データベース接続	なし

Tmax環境ファイル

以下は、Tmax環境ファイルの例です。

<sample.m>

```
*DOMAIN
resrc          SHMKEY = 77990, MAXUSER = 256

*NODE
tmax           TMAXDIR = "/home/tmax",
               APPDIR  = "/home/tmax/appbin",
               PATHDIR = "/home/tmax/path",
               TLOGDIR = "/home/tmax/log/tlog",
               ULOGDIR = "/home/tmax/log/slog",
               SLOGDIR = "/home/tmax/log/ulog"

*SVRGROUP
svg1           NODENAME = tmax

*SERVER
syncsvc       SVGNAME = svg1,
               MIN = 1, MAX = 5,
               CLOPT = " -e $(SVR).err -o $(SVR).out "

*SERVICE
TOUPPERSTR     SVRNAME = syncsvc
```

クライアント・プログラム

以下は、クライアント・プログラムの例です。

<sync_cli.c>

```
#include <stdio.h>
#include <string.h>
#include <usrinc/atmi.h>

main(int argc, char *argv[])
{
    char *sendbuf, *recvbuf;
    long rlen;

    if (argc != 2)
    {
        fprintf(stderr, "Usage: $ %s string \n", argv[0]);
        exit(1);
    }

    if (tpstart((TPSTART_T*)NULL) == -1)
    {
        fprintf(stderr, "Tpstart failed\n");
        exit(1);
    }

    if ((sendbuf = tpalloc("STRING", NULL, 0)) == NULL) {
        fprintf(stderr, "Error allocation send buffer\n");
        tpend();
        exit(1);
    }

    if ((recvbuf = tpalloc("STRING", NULL, 0)) == NULL) {
        fprintf(stderr, "Error allocation recv buffer\n");
        tpend();
        exit(1);
    }

    strcpy(sendbuf, argv[1]);

    if (tpcall("TOUPPERSTR", sendbuf, 0, &sendbuf, &rlen, TPNOFLAGS) == -1)
    {
        fprintf(stderr, "Can't send request to service TOUPPER->%s!\n",
            tpstrerror(tperrno));
        tpfree(sendbuf);
        tpfree(recvbuf);
        tpend();
        exit(1);
    }

    printf("Sent value:%s\n", sendbuf);
}
```

```
printf("Returned value:%s\n ",recvbuf );
tpfree(sendbuf);
tpfree(recvbuf);
tpend( );
```

サーバー・プログラム

以下は、サーバー・プログラムの例です。

<syncsvc.c>

```
#include <stdio.h>
#include <usrinc/atmi.h>

TOUPPERSTR(TPSVCINFO *msg)
{
    int i;

    for (i = 0; i < msg->len ; i++)
        msg->data[i] = toupper(msg->data[i]);
    msg->data[i] = '\0';

    tpreturn(TPSUCCESS, 0, msg->data, 0, TPNOFLAGS);
}
```

5.1.2. 非同期型通信

クライアントはSTRUCTバッファのメンバーに文字列をコピーしてサービス呼び出し、サーバーのサービス・ルーチンはこの文字列を受信して小文字列あるいは大文字列に変えて返すプログラムです。クライアントは非同期型通信でTOUPPERサービスを要求し、再度同期型でTOLOWERサービスを呼び出して結果を受信した後、先に要求したTOUPPERサービスの実行結果を受信します。

プログラム構成

- 共通プログラム

プログラム・ファイル	説明
demo.s	構造体バッファを定義します
sample.m	Tmax環境設定ファイルです

- クライアント・プログラム

プログラム・ファイル	説明
async_cli.c	クライアント・プログラムです

- サーバー・プログラム

プログラム・ファイル	説明
asynsvc.c	大文字/小文字に変えるサービス・プログラムです
Makefile	Tmaxが提供するMakefileを修正する必要があります

プログラムの特徴

- クライアント・プログラム

機能	説明
Tmax接続	基本接続
バッファ・タイプ	STRUCT
通信タイプ	同期型及び非同期型

- サーバー・プログラム

機能	説明
サービス	TOUPPER、TOLOWER
データベース接続	なし
通信タイプ	同期型及び非同期型

構造体バッファ

以下は、非同期型通信で使用する構造体バッファです。

<demo.s>

```
struct strdata {
    int flag;
    char sdata[20];
};
```

Tmax環境ファイル

以下は、Tmax環境ファイルの例です。

<sample.m>

```

*DOMAIN
resrc          SHMKEY = 77990, MAXUSER = 256

*NODE
tmax           TMAXDIR = "/home/tmax",
               APPDIR = "/home/tmax/appbin",
               PATHDIR = "/home/tmax/path"

*SVRGROUP
svg1           NODENAME = tmax

*SERVER
asynccsvc      SVGNAME = svg1, MIN = 1, MAX = 5

*SERVICE
TOUPPER        SVRNAME = asynccsvc
TOLOWER        SVRNAME = asynccsvc

```

クライアント・プログラム

以下は、クライアント・プログラムの例です。

<async_cli.c>

```

#include <stdio.h>
#include <string.h>
#include <usrinc/atmi.h>
#include "../sdl/demo.s"

main(int argc, char *argv[ ])
{
    struct strdata *sendbuf, *sendbuf1;
    long dlen, clen;
    int cd;

    if (argc != 3) {
        fprintf(stderr, "Usage: $ %s string STRING\n", argv[0], argv[1]);
        exit(1) ;
    }

    if (tpstart((TPSTART_T *)NULL) == -1) {
        fprintf(stderr, "TPSTART_T failed\n");
        exit(1) ;
    }

    sendbuf = (struct strdata *)tpalloc("STRUCT", "strdata", 0);

```

```

    if ( sendbuf == NULL) {
        fprintf(stderr, "Error allocation send buffer\n");
        tpend () ;
        exit(1) ;
    }

    sendbuf1 = (struct strdata *)tpalloc("STRUCT", "strdata", 0);
    if (sendbuf1 == NULL) {
        fprintf(stderr, "Error allocation send1 buffer\n");
        tpend();
        exit(1) ;
    }

    strcpy(sendbuf->sdata, argv[1]);
    strcpy(sendbuf1->sdata, argv[2]);

    if ((cd = tpacall("TOUPPER", (char *)sendbuf, 0, TPNOFLAGS)) == -1)
    {
        fprintf(stderr, "Toupper error -> %s", tpstrerror(tperrno));
        tpfree((char *)sendbuf);
        tpend();
        exit(1) ;
    }
    if (tpcall("TOLOWER", (char *)sendbuf1, 0, (char **)&sendbuf1, &dlen,
               TPSIGRSTRT) == -1) {
        fprintf(stderr, "Tolower error -> %s", tpstrerror(tperrno));
        tpfree((char *)sendbuf);
        tpend();
        exit(1) ;
    }
    if (tpgetrply(&cd, (char **)&sendbuf, &crlen, TPSIGRSTRT) == -1) {
        fprintf(stderr, "Toupper getrply error -> %s", tpstrerror(tperrno));

        tpfree((char *)sendbuf);
        tpend();
        exit(1) ;
    }
    printf("Return value %s\n %s\n", sendbuf -> sdata, sendbuf1 -> sdata);
    tpfree((char *)sendbuf);
    tpfree((char *)sendbuf1);
    tpend() ;
}

```

サーバー・プログラム

以下は、サーバー・プログラムの例です。

<asynsvc.c>

```
#include <stdio.h>
#include <usrinc/atmi.h>
#include "../sdl/demo.s"

TOUPPER(TPSVCINFO *msg)
{
    int i = 0;
    struct strdata *stdata;

    stdata = (struct strdata *)msg -> data;

    while (stdata->sdata[ i ] != '\\0') {
        stdata->sdata[ i ] = toupper(stdata->sdata[ i ]);
        i++ ;
    }

    tpreturn(TPSUCCESS, 0, (char *)stdata, 0, TPNOFLAGS);
}

TOLOWER(TPSVCINFO *msg)
{
    int i = 0;
    struct strdata *stdata;

    stdata = (struct strdata *)msg -> data;

    while ( stdata->sdata[ i ] != '\\0') {
        stdata->sdata[ i ] = tolower(stdata->sdata[ i ]);
        i++;
    }

    tpreturn(TPSUCCESS, 0, (char *)stdata, 0, TPNOFLAGS);
}
```

5.1.3. 会話型通信

クライアントはユーザーからの入力を受け、STRINGバッファを通じて固有番号を送ります。サーバーのサービス・ルーチンは、データベースに格納されたテーブルから該当固有番号より大きい番号をもつ顧客情報を、構造体を通じて返します。

クライアントは、会話型モードを設定して固有番号を送り、会話主導権をサーバーに渡します。サーバーは、条件を満たすデータベースの全データをカーソルにて読み取り、クライアントに送ります。クライアントは、TPEVSVCSUCCを通じて、正常に全データを読み取ったことを確認できます。

プログラム構成

- 共通プログラム

プログラム・ファイル	説明
demo.s	構造体を定義したファイルです
sample.m	Tmax環境設定ファイルです
mktable.sql	テーブルを作成するスクリプトです
sel.sql	テーブル及びデータを出力するスクリプトです

- クライアント・プログラム

プログラム・ファイル	説明
conv_cli.c	クライアント・プログラムです

- サーバー・プログラム

プログラム・ファイル	説明
convsvc.pc	サーバー・プログラムです
Makefile	Tmaxが提供するMakefileを修正する必要があります

プログラムの特徴

- クライアント・プログラム

機能	説明
Tmax接続	基本接続
バッファ・タイプ	送信時はSTRING、受信時はSTRUCT
通信タイプ	会話型通信STRUCT

- サーバー・プログラム

機能	説明
サービス	MULTI
データベース接続	Oracle使用

構造体バッファ

以下は、会話型通信で使用する構造体バッファです。

<demo.s>

```
struct sel_o {
    char seqno[10];
    char corpno[10];
    char compdate[8];
    int totmon;
    float guarat;
    float guamon;
} ;
```

Tmax環境ファイル

以下は、Tmax環境ファイルの例です。

<sample.m>

```
* DOMAIN
resrc          SHMKEY = 77990, MAXUSER = 256

*NODE
tmax           TMAXDIR = "/home/tmax",
               APPDIR  = "/home/tmax/appbin",
               PATHDIR = "/home/tmax/path"

* SVRGROUP
svg1           NODENAME = tmax,
               DBNAME  = ORACLE,
               OPENINFO = "ORACLE_XA+Acc=P/scott/tiger+SesTm=60",
               TMSNAME = svg1_tms

*SERVER
convsvc        SVGNAME = svg1, CONV = Y

*SERVICE
MULTI          SVRNAME = convsvc
```

以下は、環境設定に追加された項目についての説明です。

項目	説明
DBNAME	使用するデータベース名を定義します
OPENINFO	Oracleデータベースとの連動のための接続情報を設定します

項目	説明
TMSNAME	グローバル・トランザクション処理を行うプロセス名を設定します
CONV	会話型モードのサーバーを指定します

データベース・スクリプト

以下は、Oracleテーブルを作成するスクリプトの例です。

<mktable.sql>

```
sqlplus scott/tiger << EOF
create table multi_sel
(
    seqno      VARCHAR(10),
    corpno     VARCHAR(10),
    compdate   VARCHAR(8),
    totmon     NUMERIC(38),
    guarat     FLOAT,
    guamon     FLOAT
);
create unique index idx_tdb on multi_sel(seqno);
EOF
```

以下は、Oracleテーブル及びデータを出力するスクリプトの例です。

<sel.sql >

```
sqlplus scott/tiger << EOF
Desc multi_sel;
select * from multi_sel;
EOF
```

クライアント・プログラム

以下は、クライアント・プログラムの例です。

<conv_cli.c>

```
#include <stdio.h>
#include <usrinc/atmi.h>
#include "../sdl/demo.s"

main(int argc, char *argv[])
{
    struct sel_o *rcvbuf;
```

```

char *sndbuf;
long sndlen, rcvlen, revent;
int cd;

if (argc != 2) {
    printf("Usage: client string\n");
    exit(1);
}

/* tpstart()と一緒にTmaxに接続. */
if (tpstart((TPSTART_T *) NULL) == -1) {
    printf("tpstart failed\n");
    exit(1);
}

if ((sndbuf = tpalloc("STRING", NULL, 12)) == NULL) {
    printf("tpalloc failed:sndbuf\n");
    tpend();
    exit(1);
}

if ((rcvbuf = (struct sel_o *)tpalloc("STRUCT", "sel_o", 0)) == NULL) {
    printf("tpalloc failed:rcvbuf\n");
    tpfree(sndbuf);
    tpend();
    exit(1);
}

strcpy(sndbuf, argv[1]);

if ((cd = tpconnect ("MULTI", sndbuf, 0, TPRECVOONLY)) == -1){
    printf("tpconnect failed:CONVER service, tperrno=%d\n", tperrno);
    tpfree(sndbuf);
    tpfree((char *)rcvbuf);
    tpend();
    exit(1);
}

/* 会話型通信接続、会話主導権はサーバー側に渡す */
printf("tpconnect SUCESS \"MULTI\" service\n");
while ( 1 ) {    /* 多重データの受信. */
    printf("tprecv strat\n");
    if( tprecv(cd, (char **)&rcvbuf, &rcvlen, TPNOTIME, &revent) < 0 )
    {
        /* サーバーでtpreturn()で終わった場合 */
        if (revent == TPEV_SVCSUCC){
            printf("all is completed\n");
            break;

```

```

    }
    printf("tprecv failed, tperrno=%s, revent=%x\n",
           tpstrerror(tperrno), revent );
    tpfree(sndbuf);
    tpfree((char *)rcvbuf);
    tpend();
    exit(1);
}
printf("seqno = %s\t\t corpno =%s\n", rcvbuf->seqno, rcvbuf->corpno);
printf("compdate = %s\t\t totmon =%d\n", rcvbuf->compdate, rcvbuf->totmon);

printf("guarat = %f\t\t guamon =%f\n\n", rcvbuf->guarat, rcvbuf->guamon)
;
}

tpfree(sndbuf);
tpfree((char *)rcvbuf);
tpend();
printf("FINISH\n");
}

```

サーバー・プログラム

以下は、サーバー・プログラムの例です。

<convsvc.pc>

```

#include <stdio.h>
#include <string.h>
#include <usrinc/atmi.h>
#include "../sdl/demo.s"

EXEC SQL begin declare section; /* Oracleグローバル変数の宣言 */
    char seq[10];
    struct sel_o *sndbuf;
EXEC SQL end declare section;
EXEC SQL include sqlca;

MULTI(TPSVCINFO *msg)
{
    int i, cd;
    long sndlen, revent;

    memset(seq, 0, 10);
    strcpy(seq, msg->data);

    if ((sndbuf = (struct sel_o *) tmalloc ("STRUCT", "sel_o", 0)) == NULL) {

```

```

        printf("tpalloc failed:\n");
        tpreturn (TPFAIL, -1, NULL, 0, TPNOFLAGS);
    }

    /* 多量のデータのためにカーソル宣言 */
    EXEC SQL declare democursor cursor for
    select *
    from corp
    where seqno > :seq;

    EXEC SQL open democursor;
    EXEC SQL whenever not found goto end_of_fetch;
    if (sqlca.sqlcode != 0){
        printf("oracle sqlerror=%s", sqlca.sqlerrm.sqlerrmc);
        tpreturn (TPFAIL, sqlca.sqlcode, NULL, 0, TPNOFLAGS);
    }

    /* Oracleエラーがない間データ転送 */
    while ( sqlca.sqlcode == 0 ){
        EXEC SQL fetch democursor into :sndbuf;

        if (tpsend (msg->cd, (char *)sndbuf, 0, TPNOTIME, &revent) == -1){
            printf("tpsend failed, tperrno=%d, revent=%x\n", tperrno,
                revent );
            tpfree ((char *)sndbuf);
            tpreturn (TPFAIL, -1, NULL, 0, TPNOFLAGS);
        }
    }

    tpreturn (TPFAIL, sqlca.sqlcode, NULL, 0, TPNOFLAGS);

end_of_fetch:
exec sql close democursor;
printf("tpreturn before");
tpreturn (TPSUCCESS, 0, NULL, 0, TPNOFLAGS);
}

```

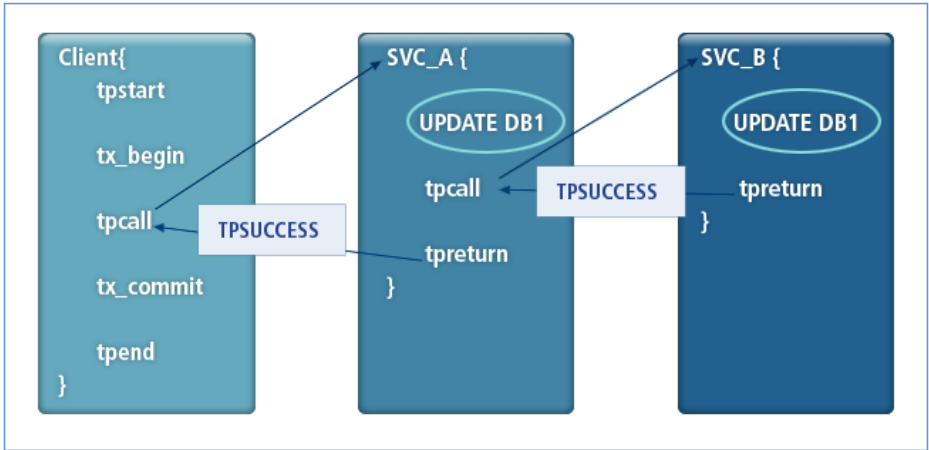
5.2. グローバル・トランザクション・プログラムの例

グローバル・トランザクション(Global Transaction Processing)とは、1つ以上のリソース管理者(データベース)と1つ以上の物理的なサイトが、1つの論理的な単位で参与するトランザクションです。Tmaxシステムでは、すべてのトランザクションをグローバル・トランザクションと見なし、データの整合性のために2PC(2 Phase Commit)を使用します。

クライアントは、ユーザーからの入力を受け、構造体バッファを通じて固有番号とデータを送ります。サーバーは該当固有番号のデータを更新し、このデータで他のデータベースを使用するサービスを呼び出し、

テーブルに挿入します。クライアントは、このすべての過程を1つのトランザクションとして指定し、エラーが発生した場合、2個のデータベースを同時にロールバックできるようにします。

[図 5.1] 2つのデータベースの接続



プログラム構成

● 共通プログラム

プログラム・ファイル	説明
demo.s	構造体バッファの設定ファイルです
sample.m	Tmax環境設定ファイルです
mktable.sql	データベース・テーブルを作成するSQLスクリプトです

● クライアント・プログラム

プログラム・ファイル	説明
client.c	クライアント・プログラムです

● サーバー・プログラム

プログラム・ファイル	説明
update.pc	データベースに更新するサーバー・プログラムです
insert.pc	データベースに挿入するサーバー・プログラムです
Makefile	Tmaxが提供するMakefileを修正する必要があります

プログラムの特徴

- クライアント部分

機能	説明
Tmax接続	基本接続
バッファ・タイプ	STRUCT
通信タイプ	tpcall()による同期通信
トランザクション処理	クライアントでトランザクション範囲を指定

- サーバー部分

機能	説明
サーバー・プログラム	互いに異なるデータベースを使用する2個のサーバー・プログラム
サービス	UPDATE、INSERT
データベース接続	2種類のOracleデータベース

構造体バッファ

以下は、グローバル・トランザクションで使用する構造体バッファです。

<demo.s>

```
struct input {
    int account_id;
    int branch_id;
    char phone[15];
    char address[61];
};
```

Tmax環境ファイル

以下は、Tmax環境ファイルの例です。

<sample.m>

```

*DOMAIN
res          SHMKEY=88000, MINCLH=1, MAXCLH=5, TPORTNO=8880, BLOCKTIME=60

*NODE
tmax1        TMAXDIR = "/user/ tmax ",
              APPDIR  = "/user/ tmax /appbin",
              PATHDIR = "/user/ tmax /path",
              TLOGDIR = "/user/ tmax /log/tlog",
              ULOGDIR="/user/ tmax /log/ulog",
              SLOGDIR="/user/ tmax /log/slog"

tmax2        TMAXDIR = "/user/ tmax ",
              APPDIR  = "/user/ tmax /appbin",
              PATHDIR = "/user/ tmax /path",
              TLOGDIR = "/user/ tmax /log/tlog",
              ULOGDIR="/user/ tmax /log/ulog",
              SLOGDIR="/user/ tmax /log/slog"

*SVRGROUP
svg1         NODENAME = tmax1, DBNAME = ORACLE,
              OPENINFO = "ORACLE_XA+Acc=P/scott/tiger+SesTm=60",
              TMSNAME = svg1_tms

svg2         NODENAME = tmax2, DBNAME = ORACLE,
              OPENINFO = "ORACLE_XA+Acc=P/scott/tiger+SesTm=60",
              TMSNAME = svg2_tms

*SERVER
update       SVGNAME=svg1
insert       SVGNAME=svg2

*SERVICE
UPDATE       SVRNAME=update
INSERT       SVRNAME=insert

```

データベース・スクリプト

以下は、Oracleテーブルを作成するスクリプトの例です。

<mktable.sql>

```

sqlplus scott/tiger << EOF
drop table ACCOUNT;

create table ACCOUNT (

```

```

ACCOUNT_ID  integer,
BRANCH_ID   integer not null,
SSN         char(13) not null,
BALANCE     number,
ACCT_TYPE   char(1),
LAST_NAME   char(21),
FIRST_NAME  char(21),
MID_INIT    char(1),
PHONE       char(15),
ADDRESS     char(61),
CONSTRAINT ACCOUNT_PK PRIMARY KEY(ACCOUNT_ID)
);
quit
EOF

```

クライアント・プログラム

以下は、クライアント・プログラムの例です。

<client.c>

```

#include <stdio.h>
#include <usrinc/atmi.h>
#include "../sdl/demo.s"

#define TEMP_PHONE "6283-2114"
#define TEMP_ADDRESS "Korea"

int main(int argc, char *argv[])
{
    struct input *sndbuf;
    char *rcvbuf;
    int acnt_id, n, timeout;
    long len;

    if (argc != 2) {
        fprintf(stderr, "Usage:%s account_id \n", argv[0]);
        exit(1);
    }

    acnt_id = atoi(argv[1]);
    timeout = 5;

    n = tmaxreadenv("tmax.env", "tmax");
    if (n < 0) {

```



```

        fprintf(stderr, "tmaxreadenv fail! tperrno = %d\n", tperrno);
        exit(1);
    }

    n = tpstart((TPSTART_T *)NULL);
    if (n < 0) {
        fprintf(stderr, "tpstart fail! tperrno = %s\n", tperrno);
        exit(1);
    }
    sndbuf = (struct input *)tpalloc("STRUCT", "input", sizeof(struct input));

    if (sndbuf == NULL) {
        fprintf(stderr, "tpalloc fail: sndbuf tperrno = %d\n", tperrno);
        tpend();
        exit(1);
    }
    rcvbuf = (char *)tpalloc("STRING", NULL, 0);
    if (rcvbuf == NULL) {
        fprintf(stderr, "tpalloc fail: rcvbuf tperrno = %d\n", tperrno);
        tpend();
        exit(1);
    }
    sndbuf->account_id = acct_id;
    sndbuf->branch_id = acct_id;
    strcpy(sndbuf->phone, TEMP_PHONE);
    strcpy(sndbuf->address, TEMP_ADDRESS);

    tx_set_transaction_timeout(timeout);
    n = tx_begin();
    if (n < 0)
        fprintf(stderr, "tx begin fail! tperrno = %d\n", tperrno);

    n = tpcall("UPDATE", (char *)sndbuf, sizeof(struct input), (char **)&rcvbuf,

        (long *)&len, TPNOFLAGS);
    if (n < 0) {
        fprintf(stderr, "tpcall fail! tperrno = %d\n", tperrno);
        tpend();
        exit(1);
    }

    n = tx_commit();
    if (n < 0) {
        fprintf(stderr, "tx commit fail! tx error = %d \n", n);
        tx_rollback();
        tpend();
        exit(1);
    }

```

```

    }
    printf("rtn msg = %s\n", rcvbuf);
    tpfree((char *)sndbuf);
    tpfree((char *)rcvbuf);
    tpend();
}

```

サーバー・プログラム

以下は、データベースに更新するサーバー・プログラムの例です。

<update.pc>

```

#include <stdio.h>
#include <ctype.h>
#include <usrinc/atmi.h>
#include <usrinc/sdl.h>
#include "../sdl/demo.s"

#define OKMSG "YOU COMPLETE THE TRANSACTION"

EXEC SQL include sqlca.h;
EXEC SQL BEGIN DECLARE SECTION;
int account_id;
    int      branch_id;
    char     ssn[15];
    char     phone[15];
    char     address[61];
EXEC SQL END DECLARE SECTION;

UPDATE(TPSVCINFO *msg)
{
    struct input *rcvbuf;
    int      ret;
    long     acnt_id, rcvlen;
    char     *send;

    rcvbuf = (struct input *) (msg->data);
    send = (char *) tppalloc("STRING", NULL, 0);
    if (send == NULL) {
        fprintf(stderr, "tpalloc fail errno = %s\n", strerror(tperrno));
        tpreturn(TPFAIL, 0, (char *)NULL, 0, 0);
    }
    account_id = rcvbuf->account_id;
    branch_id = rcvbuf->branch_id;

```

```

strcpy(phone, rcvbuf->phone);
strcpy(address, rcvbuf->address);
strcpy(ssn, "1234567");

EXEC SQL UPDATE ACCOUNT
        SET BRANCH_ID = :branch_id,
          PHONE = :phone,
          ADDRESS = :address,
          SSN = :ssn
        WHERE ACCOUNT_ID = :account_id;
if (sqlca.sqlcode != 0 && sqlca.sqlcode != 1403 ) {
    fprintf(stderr, "update failed sqlcode = %d\n", sqlca.sqlcode);
    tpreturn(TPFAIL, -1, (char *)NULL, 0, 0);
}
rcvbuf->account_id++;
ret = tpcall("INSERT", (char *)rcvbuf, 0, (char **)&send, (long *)&rcvlen,
            TPNOFLAGS);
if (ret < 0) {
    fprintf(stderr, "tpcall fail tperrno = %d\n", tperrno);
    tpreturn(TPFAIL, -1, (char *)NULL, 0, 0);
}
strcpy(send, OKMSG);
tpreturn(TPSUCCESS, 1, (char *)send, strlen(send), TPNOFLAGS);
}

```

以下は、データベースに挿入するサーバー・プログラムの例です。

<insert.pc>

```

#include <stdio.h>
#include <ctype.h>
#include <usrinc/atmi.h>
#include <usrinc/sdl.h>
#include "../sdl/demo.s"

#define OKMSG "YOU COMPLETE THE TRANSACTION"

EXEC SQL include sqlca.h;
EXEC SQL BEGIN DECLARE SECTION;
    int      account_id;
    int      branch_id;
    char      ssn[15];
    char      phone[15];
    char      address[61];
EXEC SQL END DECLARE SECTION;

INSERT(msg)

```

```

TPSVCINFO  *msg;
{
    struct input *rcvbuf;
    int      ret;
    long     acnt_id;
    char     *send;

    rcvbuf = (struct input *)(msg->data);
    send = (char *)tpalloc("STRING", NULL, 0);
    if (send == NULL) {
        fprintf(stderr, "tpalloc fail errno = %s\n", tpstrerror(tperrno));
        tpreturn(TPFAIL, 0, (char *)NULL, 0, TPNOFLAGS);
    }

    account_id = rcvbuf->account_id;
    branch_id = rcvbuf->branch_id;
    strcpy(phone, rcvbuf->phone);
    strcpy(address, rcvbuf->address);
    strcpy(ssn, "1234567");

    /* Declare && Open Cursor for Fetch */
    EXEC SQL INSERT INTO ACCOUNT (
    ACCOUNT_ID,
    BRANCH_ID,
    SSN,
    PHONE,
    ADDRESS )
    VALUES (
    :account_id, :branch_id, :ssn, :phone, :address);

    if (sqlca.sqlcode != 0 && sqlca.sqlcode != 1403 )
    {
        printf("insert failed sqlcode = %d\n", sqlca.sqlcode);
        tpreturn(TPFAIL, -1, (char *)NULL, 0, TPNOFLAGS);
    }
    strcpy(send, OKMSG);
    tpreturn(TPSUCCESS, 1, (char *)send, strlen(send), TPNOFLAGS);
}

```

5.3. データベース・プログラム

代表的なデータベースであるOracleとInformixを使用した例について説明します。

5.3.1. Oracle Insertプログラム

クライアントはユーザーからの入力を受け、構造体バッファーに入れてサービスを呼び出します。サーバーはこれを受け取り、該当テーブルに追加します。クライアントはトランザクションを指定し、エラーが発生した場合はロールバックできるようにします。

プログラム構成

- 共通プログラム

プログラム・ファイル	説明
demo.s	構造体バッファーの設定ファイルです
sample.m	Tmax環境設定ファイルです
mktable.sql	データベース・テーブルを作成するSQLスクリプトです
sel.sql	データベーステーブルの内容を出力するSQLスクリプトです

- クライアント・プログラム

プログラム・ファイル	説明
oins_cli.c	クライアント・プログラムです

- サーバー・プログラム

プログラム・ファイル	説明
oinssvc.pc	サービス・プログラムのOracleソースです
Makefile	Tmaxが提供するMakefileを修正する必要があります

プログラムの特徴

- クライアント・プログラム

機能	説明
Tmax接続	基本接続
バッファー・タイプ	STRUCT
通信タイプ	tpcall()による同期通信
トランザクション処理	クライアントでトランザクション範囲を指定

- サーバー・プログラム

区分	説明
サービス	ORAINS
データベース接続	Oracleデータベース

構造体バッファ

以下は、構造体バッファの例です。

<demo.s>

```
struct ktran {
    int no;
    char name[20];
};
```

Tmax環境ファイル

以下は、Tmax環境ファイルの例です。

<sample.m>

```
*DOMAIN
resrc          SHMKEY = 77990, MAXUSER = 256

*NODE
tmax           TMAXDIR = /home/tmax,
               APPDIR  = /home/tmax/appbin,
               PATHDIR = /home/tmax/path

*SVRGROUP
svg1           NODENAME = tmax,
               DBNAME  = ORACLE,
               OPENINFO = "Oracle_XA+Acc=P/scott/tiger+SesTm=60",
               TMSNAME = svg1_tms

*SERVER
oinssvc        SVGNAME = svg1, MIN = 1, MAX = 5

*SERVICE
ORAINS         SVRNAME = oinssvc
```

以下は、環境設定に追加された項目についての説明です。

項目	説明
DBNAME	使用するデータベースを定義します

項目	説明
OPENINFO	Oracleデータベース接続情報を設定する項目です。Oracleデータベースの場合、CLOSEINFOは指定しなくても結構です。tpsvrinfo()で呼び出します
TMSNAME	トランザクション処理を行うプロセス名を指定する項目です。OPENINFOの指定による自動トランザクションを処理します。svg1に属する該当サービスを自動トランザクション状態で処理します

データベース・スクリプト

以下は、Oracleテーブルを作成するスクリプトの例です。

<mktable.sql>

```
sqlplus scott/tiger << EOF
    create table testdb1 (
        no number(7),
        name char(30)
    ) ;
EOF
```

以下は、Oracleテーブル及びデータを出力するスクリプトの例です。

<sel.sql>

```
sqlpus scott/tiger << EOF
desc testdb1;
select * from testdb1;
select count (*) from testdb1;
EOF
```

クライアント・プログラム

以下は、クライアント・プログラムの例です。

<oins_cli.c>

```
#include <stdio.h>
#include <usrinc/atmi.h>
#include "../sdl/demo.s"

main(int argc, char *argv[])
{
    struct ktran *sndbuf, *rcvbuf;
    long sndlen, rcvlen;
    int cd;
```

```

    if (argc != 3) {
        printf("Usage: client no name\n");
        exit(1);
    }

    printf("tpstart-start \n");
    if(tpstart ((TPSTART_T *) NULL) == -1) {
        printf("Tpstart failed\n");
        exit(1);
    }
    printf("tpstart-ok \n");

    if((sndbuf=(struct ktran *) tmalloc("STRUCT","ktran",0))==NULL) {
        printf("tpalloc failed:sndbuf, tperrno=%d\n", tperrno);
        tpend();
        exit(1) ;
    }

    if((rcvbuf = (struct ktran *) tmalloc("STRUCT", "ktran", 0))== NULL) {
        printf("tpalloc failed:rcvbuf, tperrno=%d\n", tperrno);
        tpfree((char *)sndbuf);
        tpend();
        exit(1);
    }

    sndbuf->no = atoi(argv[1]);
    strcpy(sndbuf->name, argv[2]);
    printf("tpcall-start \n");
    tx_begin();

    if(tpcall("ORAINS",(char *)sndbuf,0,(char **)&rcvbuf,&rcvlen,TPNOFLAGS)==-1){

        printf("tpcall failed:ORA service, tperrno=%d", tperrno);
        printf("sql code=%d\n", tpurcode);
        tx_rollback();
        tpfree ((char *)sndbuf);
        tpfree ((char *)rcvbuf);
        tpend();
        exit(1);
    }

    printf("tpcall-success \n");
    tx_commit();

    tpfree ((char *)sndbuf);
    tpfree ((char *)rcvbuf);

```



```

        tpend();
    }

```

サーバー・プログラム

以下は、サーバー・プログラムの例です。

<oinssvc.pc>

```

#include <stdio.h>
#include <usrinc/atmi.h>
#include "../sdl/demo.s"

EXEC SQL begin declare section;
char name[20];
int no;

EXEC SQL end declare section;
EXEC SQL include sqlca;

ORAINS(TPSVCINFO *msg)
{
    struct ktran *stdata;
    stdata = (struct ktran *)msg->data;
    strcpy(name, stdata->name);
    no = stdata->no;
    printf("Ora service started\n");

    /* データベースに挿入 */
    EXEC SQL insert into testdb1(no, name) values(:no, :name);

    if (sqlca.sqlcode != 0){
        printf("oracle sqlerror=%s",sqlca.sqlerrm.sqlerrmc);
        tpreturn (TPFAIL, sqlca.sqlcode, NULL, 0, TPNOFLAGS);
    }

    tpreturn (TPSUCCESS, sqlca.sqlcode, stdata, 0, TPNOFLAGS);
}

```

5.3.2. Oracle Selectプログラム

クライアントはユーザーからの入力を受け、構造体バッファーに入れてサービスを呼び出します。サーバーはこれに該当する全データを受け取り、構造体配列を使用して結果を返します。クライアントはトランザクションを指定し、エラーが発生した場合はロールバックできるようにします。

プログラム構成

- 共通プログラム

構成	説明
demo.s	構造体バッファ設定ファイルです
sample.m	Tmaxシステム環境ファイルです
mktable.sql	データベース・テーブルを作成するSQLスクリプトです
sel.sql	データベーステーブルの内容を出力するSQLスクリプトです

- クライアント・プログラム

構成	説明
oins_cli.c	クライアント・プログラムです
cdata.c	クライアント使用の関数モジュールです

- サーバー・プログラム

構成	説明
oselsvc.pc	サービス・プログラムのOracleソースです
Makefile	Tmaxが提供するMakefileを修正する必要があります

プログラムの特徴

- クライアント・プログラム

機能	説明
Tmax接続	基本接続
サービス	ORASEL
バッファ・タイプ	STRUCT
通信タイプ	tpcall()による同期通信
トランザクション処理	クライアントでトランザクション範囲を指定

- サーバー・プログラム

機能	説明
サービス	ORASEL

機能	説明
データベース接続	Oracleデータベース
バッファ使用	必要に応じてバッファサイズを再調整

構造体バッファ

以下は、構造体バッファの例です。

<demo.s>

```
struct stru_his{
    long    ACCOUNT_ID ;
    long    TELLER_ID ;
    long    BRANCH_ID ;
    long    AMOUNT ;
} ;
```

Tmax環境ファイル

以下は、Tmax環境ファイルの例です。

<sample.m>

```
*DOMAIN
resrc          SHMKEY = 77990, MAXUSER = 256

*NODE
tmax           TMAXDIR = "/home/tmax",
               APPDIR  = "/home/tmax/appbin",
               PATHDIR = "/home/tmax/path"

*SVRGROUP
svg1           NODENAME= tmax,
               DBNAME  = ORACLE,
               OPENINFO = "Oracle_XA+Acc=P/scott/tiger+SesTm=600",
               TMSNAME = svg1_tms

*SERVER
oselsvc        SVGNAME = svg1, MIN = 1, MAX = 5

*SERVICE
ORASEL         SVRNAME = oselsvc
```

以下は、環境設定に追加された項目についての説明です。

項目	説明
DBNAME	使用するデータベース名を設定します
OPENINFO	Oracleのデータベース接続情報を設定する項目です。OracleではCLOSEINFOは設定しなくても結構です。使用できるオプションは、以下で説明します
TMSNAME	トランザクション処理を管理するプロセス名を指定します
AUTOTRAN	該当サービスを処理する際、自動的にトランザクション状態で処理します

以下は、OPENINFO項目で使用できるオプションで、LogDirとDbgFlを使用できます。

オプション	説明
LogDir	LogDirを指定した場合、指定された位置にXAに関連したログを残すことができます LogDirを指定しなかった場合、\$ORACLE_HOME/rdbms/logまたは現在のディレクトリに<xa_NULL日付.trc>ファイルが作成されます
DbgFl	何回目の段階でデバッグ・フラグを設定するかを決定します。基本的な段階の0x01、またはOCI段階の0x04などを使用します

以下は、OPENINFO項目のオプションでLogDirとDbgFlを使用する例です。

```
OPENINFO="Oracle_XA+Acc=P/アカウント/パスワード +SesTm=60+LogDir=/tmp+DbgFl=0x01"
```

注

開発時にデバッグモードを解除して使用すると、後にディスクがフル状態になることを避けることができます。

データベース・スクリプト

以下は、Oracleテーブルを作成するスクリプトの例です。

<mktable.sql>

```
sqlplus scott/tiger << EOF
    create table sel_his(
        account_id number(6),
        teller_id number(6),
        branch_id number(6),
        amount number(6)
    );
create unique index idx_tdbl on sel_his(account_id);
EOF
```

以下は、Oracleテーブル及びデータを出力するスクリプトの例です。

<sel.sql>

```
sqlplus scott/tiger << EOF
desc sel_his;
select * from sel_his;
EOF
```

クライアント・プログラム

以下は、クライアント・プログラムの例です。

<oins_cli.c>

```
#include <stdio.h>
#include <usrinc/atmi.h>
#include <usrinc/tx.h>
#include "../sdl/demo.s"
#define NARRAY 10
#define NOTFOUND 1403

main(int argc, char *argv[])
{
    struct stru_his *transf;
    int i, j;
    long urcode, nrecv, narray = NARRAY;
    long account_id, teller_id, branch_id, amount;

    if (argc != 2) {
        fprintf(stderr, "Usage: %s ACCOUNT_ID !\n", argv[0]);
        exit(0);
    }

    if (tpstart((TPSTART_T *) NULL) == -1) { /* Tmaxに接続 */
        fprintf(stderr, "TPSTART_T(tpinfo) failed -> %s!\n",
            tpstrerror(tperrno)) ;
        exit(1) ;
    }

    /* c structure構造でバッファ作成 */
    transf = (struct stru_his *) tpalloc("STRUCT", "stru_his", 0);
    if (transf == (struct stru_his *) NULL) {
        fprintf(stderr, "Tpalloc failed->%s!\n",
            tpstrerror(tperrno)) ;
        tpend();
        exit(1);
    }
}
```

```

memset(transf, 0x00, sizeof(struct stru_his));

account_id = atoi(argv[1]);
transf->ACCOUNT_ID = account_id;

/* トランザクション・タイムアウトの設定 */
tx_set_transaction_timeout(30);

/* グローバル・トランザクションの開始 */
if (tx_begin() < 0) {
    fprintf(stderr, "tx_begin() failed ->%s!\n",
        tpstrerror(tperrno));
    tpfree((char*)transf);
    tpend();
    exit(0) ;
}

if (tpcall("ORASEL", (char *)transf, 0, (char **)&transf, &nrecv,
    TPNOFLAGS)== -1){
    /* 同期通信で"ORASEL"サービス要求 */
    fprintf(stderr, "Tpcall(SELECT...)error->%s ! ",
        tpstrerror(tperrno)) ;
    tpfree((char *)transf);
    /* 失敗時、トランザクションをキャンセル */
    tx_rollback();
    tpend();
    exit(0) ;
}
/* 成功時、トランザクションをコミット */
if (tx_commit() == -1) {
    fprintf(stderr, "tx_commit() failed ->%s!\n",
        tpstrerror(tperrno)) ;
    tpfree((char *)transf);
    tpend();
    exit(0) ;
}
/* 受け取ったデータは構造体の配列です。*/
for (j =0 ; j < tpurcode ; j++) {
    /* Oracleで問い合わせたデータ結果を印刷 */
    if (j == 0)
        printf("%-12s%-10s%-10s%-10s\n",
            "ACCOUNT_ID", "TELLER_ID", "BRANCH_ID", "AMOUNT");
    account_id=transf[j].ACCOUNT_ID;
    teller_id=transf[j].TELLER_ID;
    branch_id=(*(transf+j)).BRANCH_ID;
    amount=transf[j].AMOUNT;
    printf("%-12d %-10d %-10d %-10d\n", account_id, teller_id,branch_id, amount);
}

```

```

    }
    /* 問い合わせたデータがないか、最後の場合 */
    if (urcode == NOTFOUND) {
        printf("No records selected!\n");
        tpfree((char *)transf);
        tpend();
        return 0;
    }

    tpfree((char *)transf);
    tpend();
}

```

サーバー・プログラム

以下は、サーバー・プログラムの例です。

<oselsvc.pc>

```

#include <stdio.h>
#include <usrinc/atmi.h>
#include "../sdl/demo.s"
#define NARRAY 10
#define TOOMANY 2112
#define NOTFOUND 1403
EXEC SQL include sqlca.h;

EXEC SQL begin declare section;
    long key, rowno= NARRAY;
    long account_id[NARRAY],teller_id[NARRAY], branch_id[NARRAY],
        amount[NARRAY] ;
EXEC SQL end declare section;

ORASEL(TPSVCINFO *msg)
{
    struct stru_his *transf;
    int i , lastno;
    transf=(struct stru_his *) msg->data;

    /* msgバッファの内容をプログラム変数に渡す*/
    key = transf->ACCOUNT_ID;

    /* transfバッファのサイズを再調整 */
    if((transf=(struct stru_his *) tprealloc((char*)transf,
        sizeof(struct stru_his) * NARRAY ))==(struct stru_his*)NULL){
        fprintf(stderr, "tprealloc error ->%s\n",

```

```

        tpstrerror(tperrno));
        tpreturn(TPFAIL, tperrno, NULL, 0, TPNOFLAGS);
    }
EXEC SQL select account_id, teller_id, branch_id, amount
        into :account_id, :teller_id, :branch_id, :amount from sel_his

        where account_id > :key
/* account_idがクライアントで送ったkey値より大きいものを */
order by account_id;          /* グローバル変数に入れる */

/* sqlエラーチェック(問い合わせたものがない場合、あるいは多過ぎる場合は除外) */
if (sqlca.sqlcode!=0 && sqlca.sqlcode!=NOTFOUND && sqlca.sqlcode!=TOOMANY)
{
    fprintf(stderr, "SQL ERROR ->NO(%d):%s\n", sqlca.sqlcode,
        sqlca.sqlerrm.sqlerrmc) ;
    tpreturn(TPFAIL, sqlca.sqlcode, NULL, 0, TPNOFLAGS);
}

/* アクセスした数をlastnoに入れる */
lastno = sqlca.sqlerrd[2];

/* 問い合わせたデータが多過ぎる */
if (sqlca.sqlcode == TOOMANY)
    lastno =rowno;

/* No records */
if (lastno == 0)
    transf->ACCOUNT_ID = 0;

/* Oracleで問い合わせたデータを転送するバッファーに入れる */
for ( i = 0 ; i < lastno; i++) {
    transf[i].ACCOUNT_ID = account_id[i];
    transf[i].TELLER_ID = teller_id[i];
    transf[i].BRANCH_ID = branch_id[i];
    transf[i].AMOUNT = amount[i];
}
tpreturn(TPSUCCESS, lastno, transf, i * sizeof(struct stru_his),
TPNOFLAGS );
}

```

5.3.3. Informix Insertプログラム

クライアントはユーザーからの入力を受け、構造体バッファーに入れてサービスを呼び出します。サーバーはこれを受け取り、該当テーブルに追加します。クライアントはトランザクションを指定し、エラーが発生した場合はロールバックします。

Informixアプリケーションをコンパイルする前に、以下の事項を確認してください。

1. UNIX環境を確認します。(profile、.login、.cshrc)

以下の値を設定します。

```
INFORMIXDIR=/home/informix
INFORMIXSERVER=tmax
ONCONFIG=onconfig
PATH=$INFORMIXDIR/bin: ...
LD_LIBRARY_PATH=/home/informix/lib:/home/informix/lib/esql:
...
```

2. Makefileを確認します。

以下のオペレーションと設定を確認します。

```
# Server esql makefile

TARGET = <target filename>
APOBJS = $(TARGET).o
SDLFILE = info.s

LIBS = -lsvr -linfs
# solarisの場合、-lnsl -lsocketを追加

OBJS = $(APOBJS) $(SDLOBJ) $(SVCTOBJ)
SDLOBJ = ${SDLFILE:.s=_sdl.o}
SDLC = ${SDLFILE:.s=_sdl.c}
SVCTOBJ = $(TARGET)_svctab.o

CFLAGS = -O -I$(INFORMIXDIR)/incl/esql -I$(TMAXDIR)
# Solaris 32bit, Compaq, Linux: CFLAGS = -O -I$(INFORMIXDIR)/incl/esql
#                                     -I$(TMAXDIR)
# Solaris 64bit: CFLAGS = -xarch=v9 -O -I$(INFORMIXDIR)/incl/esql -I$(TMAXDIR)
# HP 32bit: CFLAGS = -Ae -O -I$(INFORMIXDIR)/incl/esql -I$(TMAXDIR)
# HP 64bit: CFLAGS = -Ae +DA2.0W +DD64 +DS2.0 -O -I$(INFORMIXDIR)/incl/esql
#                                     -I$(TMAXDIR)
# IBM 32bit: CFLAGS = -q32 -brtl -O -I$(INFORMIXDIR)/incl/esql -I$(TMAXDIR)
# IBM 64bit: CFLAGS = -q64 -brtl -O -I$(INFORMIXDIR)/incl/esql -I$(TMAXDIR)

INFLIBD = $(INFORMIXDIR)/lib/esql
INFLIBDD = $(INFORMIXDIR)/lib
INFLIBS = -lifsql -lifasf -lifgen -lifos -lifgls -lm -ldl -lcrypt
#                                     $(INFORMIXDIR)/lib/esql/
checkapi.o -lifglx -lifxa

APPDIR = $(TMAXDIR)/appbin
```

```

SVCTDIR = $(TMAXDIR)/svct
TMAXLIBDIR = $(TMAXDIR)/lib

#
.SUFFIXES : .ec .s .c .o

.ec.c :
    esql -e $*.ec

#
# server compile
#
all: $(TARGET)

$(TARGET):$(OBS)
    $(CC) $(CFLAGS) -L$(TMAXLIBDIR) -L$(INFLIBD) -L$(INFLIBDD) -o $(TARGET)

    $(OBS) $(LIBS) $(INFLIBS)
    mv $(TARGET) $(APPDIR)/.
    rm -f $(OBS)

$(APOBJS): $(TARGET).ec
    esql -e -I$(TMAXDIR)/usrinc $(TARGET).ec
    $(CC) $(CFLAGS) -c $(TARGET).c

$(SVCTOBJ):
    touch $(SVCTDIR)/$(TARGET)_svctab.c
    $(CC) $(CFLAGS) -c $(SVCTDIR)/$(TARGET)_svctab.c

$(SDLOBJ):
    $(TMAXDIR)/bin/sdlc -i ../sdl/$(SDLFILE)
    $(CC) $(CFLAGS) -c ../sdl/$(SDLC)

#
clean:
    -rm -f *.o core $(TARGET) $(TARGET).lis

```

<TMS Makefile>

```

#
TARGET = info_tms

INFOLIBDIR = ${INFORMIXDIR}/lib
INFOELIBDIR = ${INFORMIXDIR}/esql
INFOLIBD = ${INFORMIXDIR}/lib/esql
INFOLIBS = -lifsq1 -lifasf -lifgen -lifos -lifgls -lm -ldl -lcrypt
           /opt/informix/lib/esql/checkapi.o

```

```

-lifglx -lifxa
# solarisはライブラリーに-lnsl -lsocket -laio -lelfを追加

CFLAGS =-O -I$(INFORMIXDIR)/incl/esql -I$(TMAXDIR)
# Solaris 32bit, Compaq, Linux: CFLAGS = -O -I$(INFORMIXDIR)/incl/esql
-I$(TMAXDIR)
# Solaris 64bit: CFLAGS = -xarch=v9 -O -I$(INFORMIXDIR)/incl/esql -I$(TMAXDIR)
# HP 32bit: CFLAGS = -Ae -O -I$(INFORMIXDIR)/incl/esql -I$(TMAXDIR)
# HP 64bit: CFLAGS = -Ae +DA2.0W +DD64 +DS2.0 -O -I$(INFORMIXDIR)/incl/esql
-I$(TMAXDIR)
# IBM 32bit: CFLAGS = -q32 -brtl -O -I$(INFORMIXDIR)/incl/esql -I$(TMAXDIR)
# IBM 64bit: CFLAGS = -q64 -brtl -O -I$(INFORMIXDIR)/incl/esql -I$(TMAXDIR)

TMAXLIBDIR = $(TMAXDIR)/lib
TMAXLIBS = -ltms -linfs
# CC = /opt/SUNWspro/bin/cc : solaris only

$(TARGET): $(APOBJ)
    $(CC) $(CFLAGS) -o $(TARGET) -L$(TMAXLIBDIR) -L$(INFOLIBD)
-L$(INFOLIBDIR) -L
$(INFOELIBDIR) $(INFOLIBS) $(TMAXLIBS)
    mv $(TARGET) $(TMAXDIR)/appbin

#
clean:
    -rm -f *.o core $(TARGET)

```

プログラム構成

- 共通プログラム

プログラム・ファイル	説明
demo.s	構造体バッファの設定ファイルです
sample.m	Tmaxシステム環境ファイルです
mkdb.sql	データベースを作成するSQLスクリプトです。XAモードはデータベースがロギングモードで作成される際にサポートします
mktable.sql	データベース・テーブルを作成するSQLスクリプトです
sel.sql	データベーステーブルの内容を出力するSQLスクリプトです
info.s	SDLFILEです

- クライアント・プログラム

プログラム・ファイル	説明
client.c	クライアント・プログラムです

- サーバー・プログラム

構成	説明
tdbsvr.ec	サーバー・プログラムです
Makefile	Tmaxが提供するMakefileを修正します

プログラムの特徴

- クライアント・プログラム

機能	説明
Tmax接続	基本接続
バッファ・タイプ	STRUCT
通信タイプ	tpcall()による同期通信
トランザクション処理	クライアントでトランザクション範囲を指定

- サーバー・プログラム

機能	説明
サービス	INSERT
データベース接続	Informixデータベース

構造体バッファ

以下は、構造体バッファの例です。

<demo.s>

```
struct info {
    char seq[8];
    char data01[128];
    char data02[128];
    char data03[128];
};
```

Tmax環境ファイル

以下は、Tmax環境ファイルの例です。

<sample.m>

```
*DOMAIN
resrc          SHMKEY = 77990, MAXUSER = 256

*NODE
tmax           TMAXDIR = "/home/tmax", A
               APPDIR = "/home/tmax/appbin",
               PATHDIR = "/home/tmax/path"

*SVRGROUP
svg1           NODENAME = tmax,
               DBNAME = INFORMIX,
               OPENINFO = "stores7",
               CLOSEINFO = "",
               TMSNAME = info_tms

*SERVER
tdbsvr         SVGNAME = svg1, MIN = 1, MAX = 5

*SERVICE
INSERT        SVRNAME = tdbsvr
```

以下は、環境設定に追加された項目についての説明です。

項目	説明
DBNAME	使用するデータベース名を設定します
OPENINFO, CLOSEINFO	Informixデータベース接続及び解除のための情報を、tpsvrinfo()、tpsvrdone()で呼び出します
TMSNAME	トランザクション処理を行うプロセス名を指定する項目です。OPENINFOの指定による自動トランザクションを処理します。svg1に属する該当サービスを自動トランザクション状態で処理します

データベース・スクリプト

以下は、Informixテーブルを作成するスクリプトの例です。

<mktable.sql>

```
dbaccess << EOF
create database stores7 with buffered log;
grant connect to public;

database stores7;
drop table testdb1;
```

```

create table testdb1 (
    seq          VARCHAR(8) ,
    data01       VARCHAR(120) ,
    data02       VARCHAR(120) ,
    data03       VARCHAR(120)
) lock mode row;

create unique index idx_tdb1 on testdb1(seq);
EOF

```

以下は、Informixテーブル及びデータを出力するスクリプトの例です。

<sel.sql>

```

dbaccess << EOF
database stores7;
select * from testdb1;
EOF

```

クライアント・プログラム

以下は、クライアント・プログラムの例です。

<client.c>

```

#include <stdio.h>
#include <usrinc/atmi.h>
#include <usrinc/tx.h>
#include "info.s"
main(int argc, char **argv)
{
    struct info *transf;
    char data[256];
    long nrecv;

    /* Tmaxに接続 */
    if ((tpstart((TPSTART_T *)NULL) == -1) {
        fprintf(stderr, "tpstart(TPINFO...) failed ->%s!\n",
            tpstrerror(tperrno)) ;
        exit(1) ;
    }

    /* アプリケーション・プログラムで使用するバッファ・メモリの割り当て */
    if ((transf=(struct info *)tpalloc ("STRUCT","info",0))==(struct info *)NULL){

        fprintf(stderr, "tpalloc(struct info, ...) failed ->%s!\n",
            tpstrerror(tperrno)) ;
        tpend();
    }
}

```

```

        exit(1) ;
    }

    /* 転送するデータフィールドの設定 */
    strcpy(transf->seq, "000001");
    strcpy(transf->data01, "Hello");
    strcpy(transf->data02, "World");
    strcpy(transf->data03, "1234");

    /* トランザクション・タイムアウトの設定 * /
    tx_set_transaction_timeout (30);

    /* トランザクションの開始を通知 */
    if (tx_begin() < 0) {
        fprintf(stderr, "tx_begin() failed ->%s!\n",
            tpstrerror(tperrno)) ;
        tpfree ((char *)transf);
        tpend( ); exit(0);
    }

    /* サービスの呼び出し */
    if (tpcall("INSERT", (char*) transf, 0, (char **)&transf, &nrecv, TPNOFLAGS)==-1){

        fprintf(stderr, "tpcall(struct info, ...)
        failed ->%s!\n", tpstrerror(tperrno)) ;
        tx_rollback ();
        tpfree ((char *)transf),
        tpend();
        exit(0);
    }

    /* トランザクションが成功 */
    if (tx_commit () < 0) {
        fprintf(stderr, "tx_commit() failed ->%s!\n", tpstrerror(tperrno)) ;
        tpfree ((char *)transf);
        tpend( );
        exit(0);
    }

    tpfree ((char *)transf );
    /* Tmax接続を終了 */
    tpend( ) ;
}

```

サーバー・プログラム

以下は、サーバー・プログラムの例です。

<tdbsvr.ec>

```
#include <stdio.h>
#include <ctype.h>
#include <usrinc/atmi.h>
#include "info.s"

EXEC SQL include sqlca.h;

/* サービス名 */
INSERT(TPSVCINFO *msg)
{
    /* プログラム上のバッファ・タイプの宣言 */
    struct info *INFO;

    /* SQL文内のバッファ・タイプの宣言 */
    EXEC SQL begin declare section;
    varchar seq[8],buf01[128],buf02[128],buf03[128];
    EXEC SQL end declare section;

    /* メッセージ・バッファから構造体型でデータを受け取る */
    INFO = (struct info *)msg -> data;

    /* 構造体形式で受け取ったデータをデータベース・バッファにコピー */
    strcpy(seq, INFO->seq);
    strcpy(buf01, INFO->data01);
    strcpy(buf02, INFO->data02);
    strcpy(buf03, INFO->data03);

    /* SQL文を挿入 */
    EXEC SQL insert into testdb1 (seq,data01,data02,data03)
    values(:seq, :buf01, :buf02, :buf03);

    /* エラーが発生した場合 */
    if ( sqlca.sqlcode != 0) {
        /* 挿入失敗を通知 */
        printf("SQL error => %d !" ,sqlca.sqlcode);
        tpreturn (TPFAIL, sqlca.sqlcode, NULL, 0, TPNOFLAGS);
    }

    /* 挿入が正常に終了した場合 */
}
```



```

        tpreturn (TPSUCCESS, 0, NULL, 0, TPNOFLAGS);
    }

```

5.3.4. Informix Selectプログラム

クライアントはユーザーからの入力を受け、構造体バッファに入れてサービスを呼び出します。サーバーはこれに該当する全データを受け取り、構造体配列を使用して結果を返します。クライアントはトランザクションを指定し、エラーが発生した場合はロールバックできるようにします。

プログラム構成

- 共通プログラム

プログラム・ファイル	説明
acct.s	SDLFILEです
sample.m	Tmax環境ファイルです
mkdb.sql	データベースを作成するSQLスクリプトです
mktables.sql	データベース・テーブルを作成するSQLスクリプトです
sel.sql	データベース・テーブルの内容を出力するSQLスクリプトです

- クライアント・プログラム

構成	説明
client.c	クライアント・プログラムです
cdate.c	client.cで使用された関数モジュールです

- サーバー・プログラム

構成	説明
sel_acct.ec	サービス・プログラムのInformixソースです
Makefile	Tmaxが提供するMakefileを修正します

プログラムの特徴

- クライアント・プログラム

区分	説明
Tmax接続	基本接続
バッファ・タイプ	STRUCT
通信タイプ	tpcall()による同期通信
トランザクション処理	クライアントでトランザクション範囲を指定

- サーバー・プログラム

区分	説明
サービス	SEL_ACCT
データベース接続	Informixデータベース
バッファ使用	必要に応じてバッファサイズを再調整

構造体バッファ

以下は、構造体バッファの例です。

<acct.s>

```
struct stru_acct {
    int    ACCOUNT_ID;
    char   PHONE[20];
    char   ADDRESS[80];
};
```

Tmax環境ファイル

以下は、Tmax環境ファイルの例です。

<sample.m>

```
*DOMAIN
resrc          SHMKEY = 77990, MAXUSER = 256

*NODE
tmax           TMAXDIR = "/home/tmax",
               APPDIR  = "/home/tmax/appbin",
               PATHDIR = "/home/tmax/path"

*SVRGROUP
svg1           NODENAME = tmax,
               DBNAME  = INFORMIX,
               OPENINFO = "stores7",
               CLOSEINFO = "",
```

```

TMSNAME = info_tms

*SERVER
SEL_ACCT      SVGNAME = svg1, MIN = 1, MAX = 5

*SERVICE
SEL_ACCT      SVRNAME = sel_acct

```

以下は、環境設定に追加された項目についての説明です。

項目	説明
DBNAME	使用するデータベースを設定します
OPENINFO, CLOSEINFO	Informixデータベースへの接続及び解除のための情報を、tpsvrinit()、tpsvrdone()で呼び出します
TMSNAME	トランザクション処理を行うプロセス名を指定する項目です。OPENINFOの指定による自動トランザクションを処理します。svg1に属する該当サービスを自動トランザクション状態で処理します

データベース・スクリプト

以下は、Informixテーブルを作成するスクリプトの例です。

<mkdb.sql>

```

dbaccess << EOF
create database stores7 with buffered log;
grant connect to public;

database stores7;
drop table ACCOUNT;

create table ACCOUNT (
    account_id INTEGER,
    phone VARCHAR(20),
    address VARCHAR(80)
) lock mode row;

create unique index idx_tdb1 on ACCOUNT(account_id);
EOF

```

以下は、Informixテーブル及びデータを出力するスクリプトの例です。

<sel.sql>

```

dbaccess << EOF
database stores7;

```

```
select * from ACCOUNT;
EOF
```

クライアント・プログラム

以下は、クライアント・プログラムの例です。

<client.c>

```
#include <stdio.h>
#include <time.h>
#include <usrinc/atmi.h>
#include <usrinc/tx.h>
#include "acct.s"
#define NOTFOUND 1403

void htime(char *, int *);

main(int argc, char *argv[ ])
{
    struct stru_acct *transf;
    float tps;
    int i, j, loop, cnt_data = 0, sec1, sec2;
    long urcode, nrecv, narray;
    char ts[30], te[30], phone[20], address[80];
    int account_id, key;

    if(argc != 2) {
        fprintf(stderr, "Usage: %s LOOP (NARRAY = 30) !\n", argv[0]);
        exit(0) ;
    }

    /* ユーザーが実行したい回数分ループを行う */
    loop = atoi(argv[1]);

    /* Tmaxに接続 */
    if (tpstart((TPSTART_T *)NULL) == -1) {
        fprintf(stderr, "tpstart(tpinfo) failed ->%s!\n",
            tpstrerror(tperrno));
        exit(1) ;
    }

    /* sec1 = 開始時間 */
    htime(ts, &sec1); key=0;

    /* メッセージ・バッファの割り当て */
    for( i = 0; i < loop; i++) {
```

```

    if ((transf=(struct stru_acct *)tpalloc("STRUCT","stru_acct",0))
        ==(struct stru_acct *)NULL) {
        fprintf(stderr,"Tpalloc(STRUCT.. )failed->%s!\n" ,
            tpstrerror(tperrno)) ;
        tpend(); exit(1);
    }
    transf -> ACCOUNT_ID = key;

    /* time-out value= 30 */
    tx_set_transaction_timeout(30) ;

    if (tx_begin() < 0) { /* トランザクション開始 */
        fprintf(stderr, "tx_begin() failed ->%s!\n", tpstrerror(tperrno)) ;

        tpfree((char*)transf);
        tpend();
        exit(0);
    }

    /* select service呼び出し */
    if (tpcall("SEL_ACCT", (char *)transf, 0, (char **)&transf, &nrecv,
        TPNOFLAGS)== -1) {
        /* service error : message buffer free & トランザクションのキャンセル & 接
続終了 */
        fprintf(stderr,"Tpcall(SELECT...)error->%s! " ,
            tpstrerror(tperrno)) ;
        tpfree ((char *)transf);
        tx_rollback ( ) ;
        tpend( ) ;
        exit ( 1 ) ;
    }
    urcode = tpurcode;

    /*正常にサービス完了 : トランザクション結果で実際にリソースを変化させる*/
    if (tx_commit() < 0 ) {
        fprintf(stderr, "tx_commit() failed ->%s!\n", tpstrerror(tperrno))
;

        tpfree((char *)transf);
        tpend();
        exit(0);
    }

    /*データを問い合わせた場合*/
    if ( urcode != NOTFOUND) {
        narray =urcode;
        /* 問い合わせたデータの最後のレコード */
        key=transf[narray-1].ACCOUNT_ID;

```

```

/* 問い合わせた個数分の結果をユーザーに出力 */
for ( j = 0 ; j < narray ; j++ ) {
    if ( j == 0 )
        printf("%-10s%-14s%s\n", "ACCOUNT_ID", "PHONE", "ADDRESS") ;
        account_id = transf[j].ACCOUNT_ID;
        strcpy(phone, transf[j].PHONE);
        strcpy(address, transf[j].ADDRESS);
        printf("%-10d %-14s %s\n", account_id, phone, address);
    }/* for2 end */

/* 全結果数の増加 */
cnt_data += j;

/* message buffer free */
tpfree ((char *)transf);
if(urcode == NOTFOUND) {
    printf("No records selected!\n");
    break ;
}
}/* for1 end */

/* message buffer free */
tpfree ((char *)transf);
/* Tmax接続の終了 */
tpend ();

/* sec2 = 終了時間 */
htime(te,&sec2);

/* データ1つあたりの処理時間を計算 */
printf("TOT.COUNT = %d\n", cnt_data);
printf("Start time = %s\n", ts);
printf("End time = %s\n", te);
if ((sec2-sec1) != 0)
    tps = (float) cnt_data / (sec2 - sec1);
else
    tps = cnt_data;
printf("Interval = %d secs ==> %10.2f T/S\n", sec2-sec1,tps);
}

htime(char *cdate, int *sec)
{
    long time(), timef, pt;
    char ct[20], *ap;
    struct tm *localtime(), *tmp;

    pt = time(&timef);

```

```

        *sec = pt;
        tmp = localtime(&timef);
        ap = asctime(tmp);

        sscanf(ap, "%*s%*s%*s%*s", ct);
        sprintf( cdate, "%02d. %02d. %02d %s", tmp->tm_year, ++tmp->tm_mon,
                tmp->tm_mday, ct);
    }

```

サーバー・プログラム

以下は、サーバー・プログラムの例です。

<sel_acct.pc>

```

#include <stdio.h>
#include <usrinc/atmi.h>
#include <usrinc/tx.h>
#include "acct.s"
#define NFETCH 5
#define NOTFOUND 100

EXEC SQL include sqlca.h;
EXEC SQL begin declare section;
long account_id, key;
varchar phone[20], address[80];
EXEC SQL end declare section;

SEL_ACCT(TPSVCINFO *msg)
{
    int i , j , nfetch;
    int return_code;
    struct stru_acct *ACCT_V;

    /*クライアントのデータを受け取る */
    ACCT_V = (struct stru_acct *) msg->data;

    /* 問い合わせたいaccount idをkeyに移す */
    key = ACCT_V->ACCOUNT_ID;

    /* クライアント・バッファサイズのリ割り当て */
    if ((ACCT_V = (struct stru_acct *) tprealloc((char *) ACCT_V,
        sizeof(struct stru_acct)*NFETCH )) == (struct stru_acct *) NULL) {
        fprintf(stderr, "tprealloc error = %s\n", tpstrerror(tperrno));
        tpreturn (TPFAIL, tperrno, NULL, 0, TPNOFLAGS);
    }
}

```

```

/*バッファの初期化*/
ACCT_V->ACCOUNT_ID = 0;
strcpy(ACCT_V->PHONE," " );
strcpy(ACCT_V->ADDRESS," " );

/* ACCOUNTテーブルからphone、 addressフィールドを問い合わせ */
EXEC SQL declare CUR_1 cursor for
        select account_id,phone,address
        into :account_id, :phone, :address
        from ACCOUNT
        where account_id > :key; /* フィールド・キーよりaccount idが大きいとき */

/* cursor open */
EXEC SQL open CUR_1;
/* cursor open error */
if (sqlca.sqlcode != 0) {
        printf("open cursor error !\n");
        tpreturn(TPFAIL, sqlca.sqlcode, NULL, 0, TPNOFLAGS);
}

nfetch=0 ;
return_code = NOTFOUND;

/* cursor open success */
while (sqlca.sqlcode == 0) {
        /* カーソルからデータを1つずつフェッチ */
        EXEC SQL fetch CUR_1;

        /* fetch error */
        if (sqlca.sqlcode != 0) {
                if (sqlca.sqlcode == NOTFOUND)
                        break ;
                break ;
        }

        ACCT_V[nfetch].ACCOUNT_ID = account_id;
        strcpy(ACCT_V[nfetch].PHONE, phone );
        strcpy(ACCT_V[nfetch].ADDRESS, address );

        /* 問い合わせたデータ数の増加 */
        nfetch++;
        return_code = nfetch

        /* NFETCHの数の分だけ問い合わせた場合、while文から抜ける */
        if (nfetch > NFETCH) {

```



```

        nfetch = NFETCH;
        return_code = nfetch;
        break ;
    }
}
/* cursor close */
EXEC SQL close CUR_1;

/* 問い合わせ結果とそのデータをクライアントに返す */
tpreturn ( TPSUCCESS, return_code, (char *)ACCT_V,
          sizeof(struct stru_acct)*nfetch, TPNOFLAGS);
}

```

5.3.5. DB2プログラム

クライアントはユーザーの入力を受け、STRINGバッファにEMPNOを入れてサービスを呼び出し、サーバーはこれを受けて該当するテーブルに追加します。クライアントはトランザクションを指定し、エラーが発生した場合にロールバックできるようにします。

プログラム構成

- 共通プログラム

プログラム・ファイル	説明
sample.m	Tmax環境設定ファイルです
create.ers	データベース・テーブルを作成するSQLスクリプトです

- クライアント・プログラム

プログラム・ファイル	説明
clidb2tx.c	クライアント・プログラムです

- サーバー・プログラム

プログラム・ファイル	説明
svr_db2.sqc	サービス・プログラムのOracleソースです
Makefile	TMSおよびサーバー・プログラムをコンパイルするメイクファイルです

プログラムの特徴

- クライアント・プログラム

機能	説明
Tmax接続	基本接続
バッファ・タイプ	STRING
通信タイプ	tpcall()による同期型通信
トランザクション処理	クライアントでトランザクション範囲を指定

● サーバー・プログラム

区分	説明
サービス	XASERVICE2
データベース接続	DB2データベース

DB2連携の前の確認事項

以下は、DB2連携時の考慮事項です。

1. DB2クライアント・エンジンのビット(32あるいは64)を確認します。
2. DB2クライアントのバージョンが8.0以下なのか、9.0以上なのかを確認します。
3. Tmax環境設定ファイルのSVRGROUPセクションのXAOPTION項目を1、2項の結果に基づいて設定します。

– DB2クライアントのバージョンが8.0以下の場合

- DB2クライアント・エンジンが32ビットの場合

```
XAOPTION = "DYNAMIC"
```

- DB2クライアント・エンジンが64ビットであり、Linux/UNIX系の場合

```
XAOPTION = "DYNAMIC XASWITCH32"
```

- DB2クライアント・エンジンが64ビットであり、Windows系の場合

```
XAOPTION = "DYNAMIC"
```

– DB2クライアントのバージョンが9.0以上の場合 (DB2クライアント・エンジンのビットおよびLinux/UNIX/Windowsを問わない)

```
XAOPTION = 未設定
```

4. TMSまたはSVRのコンパイル時、1、2、3項の結果に基づいて適切なライブラリーをリンクします。

- DB2クライアントのバージョンが8.0以下の場合

- DB2クライアント・エンジンが32ビットの場合

```
-ldb2s or $(TMAXDIR)/lib/libdb2s.a
```

- DB2クライアント・エンジンが64ビットであり、Linux/UNIX系の場合

```
-ldb2_64s or $(TMAXDIR)/lib64/libdb2_64s.a
```

- DB2クライアント・エンジンが64ビットであり、Windows系の場合

```
-ldb2s or $(TMAXDIR)/lib/libdb2s.a
```

- DB2クライアントのバージョンが9.0以上の場合 (DB2クライアント・エンジンのビットおよびLinux/UNIX/Windowsを問わない)

```
-ldb2s_static or $(TMAXDIR)/lib/libdb2s_static.a
```

参考

バージョン9.0以上のDB2クライアントの動的登録を使用する必要がある場合は、上記の8.0の場合と同様に設定することができますが、使用を推奨しません。

Tmax環境ファイル

以下は、Tmax環境ファイルの例です。

<sample.m>

```
*DOMAIN
tmax1      SHMKEY = 71990, MINCLH = 1, MAXCLH = 3,
           TPORTNO = 7789, BLOCKTIME = 30,
           MAXCPC = 150

*NODE
phk        TMAXDIR = "/home/tmaxha/tmax",
           APPDIR  = "/home/tmaxha/tmax/appbin",
           PATHDIR = "/home/tmaxha/tmax/path",
           TLOGDIR = "/home/tmaxha/tmax/log/tlog",
           ULOGDIR = "/home/tmaxha/tmax/log/ulog",
           SLOGDIR = "/home/tmaxha/tmax/log/slog"

*SVRGROUP
xa_svg_db2  NODENAME = "phk",
           DBNAME  = IBMDB2,
```

```
#
XAOPTION = "DYNAMIC XASWITCH32",
XAOPTION = "DYNAMIC",
OPENINFO = "db=test,uid=tmaxha,pwd=ha0115",
TMSNAME = tms_db2,
RESTART=N

*SERVER
svr_db2          SVGNAME = xa_svg_db2

*SERVICE
XASERVICE2      SVRNAME = svr_db2
```

以下は、環境設定の追加項目についての説明です。

項目	説明
DBNAME	使用するデータベースを定義します
OPENINFO	DB2データベースの接続情報を設定します
TMSNAME	トランザクション処理を管理するプロセス名を指定します。OPENINFO指定による自動トランザクションを処理します

データベース・スクリプト

以下は、DB2テーブル作成の例です。

```
# 'db2start' を実行
$ db2start

# 'TPTEST' というデータベースを作成
$ db2 "CREATE DATABASE TPTEST"

# TPTESTデータベースに接続
$ db2 "CONNECT TO TPTEST"

# EMPテーブルの作成
$ db2 -vf create.ers -t

<create.ers>
CREATE TABLE EMP (
    EMPNO          DECIMAL(8) NOT NULL,
    ENAME          VARCHAR(16),
    JOB            VARCHAR(16),
    SAL            DECIMAL(8),
    HIREDATE        DECIMAL(8),
    XID            CHAR(32)
);
```

```
# EMPテーブルが作成されたことを確認
$ db2 "LIST TABLES"
```

以下は、DB2テーブルおよびデータ出力の例です。

```
$ db2 "DESCRIBE TABLE EMP"

Column          Type      Type      Length  Scale Nulls
name            schema   name                               -----
-----
EMPNO           SYSIBM   DECIMAL      8        0  No
ENAME           SYSIBM   VARCHAR     16        0  Yes
JOB             SYSIBM   VARCHAR     16        0  Yes
SAL             SYSIBM   DECIMAL      8        0  Yes
HIREDATE        SYSIBM   DECIMAL      8        0  Yes
XID             SYSIBM   CHARACTER    32        0  Yes

6 record(s) selected.
```

クライアント・プログラム

以下は、クライアント・プログラムの例です。

<clidb2tx.c>

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <usrinc/atmi.h>

int main(int argc, char *argv[])
{
    char      *sndbuf, *rcvbuf;
    long      rcvlen;
    int       ret;

    if ( (ret = tmaxreadenv( "tmax.env", "TMAX" )) == -1 ){
        printf( "tmax read env failed. [%s]\n", tpstrerror(tperrno));
        exit(1);
    }

    if (tpstart((TPSTART_T *)NULL) == -1){
        printf("tpstart failed. [%s]\n", tpstrerror(tperrno));
        exit(1);
    }
}
```

```

if ((sndbuf = (char *)tpalloc("STRING", NULL, 0)) == NULL) {
    printf("sndbuf alloc failed! [%s]\n", tpstrerror(tperrno));
    tpend();
    exit(1);
}

if ((rcvbuf = (char *)tpalloc("STRING", NULL, 0)) == NULL) {
    printf("rcvbuf alloc failed! [%s]\n", tpstrerror(tperrno));
    tpfree(sndbuf);
    tpend();
    exit(1);
}

strcpy(sndbuf, argv[1]);

ret = tx_begin();
if (ret < 0) {
    printf("tx_begin is failed. [%s]\n", tpstrerror(tperrno));
    resource_free(sndbuf, rcvbuf);
    exit(1);
} else
    printf("tx_begin success.\n");

if (tpcall("XASERVICE2", sndbuf, strlen(sndbuf), &rcvbuf, &rcvlen, 0) == -1){

    printf("Can't send request to service XASERVICE2. [%s]\n",
tpstrerror(tperrno));
    ret = tx_rollback();
    if (ret < 0)
        printf("tx_rollback is failed. [%s]\n", tpstrerror(tperrno));

    resource_free(&sndbuf, &rcvbuf);
    exit(1);
} else
    printf("XASERVCE2 success.\n");

ret = tx_commit();
if (ret < 0) {
    printf("tx_commit is failed. [%s]\n", tpstrerror(tperrno));
    ret = tx_rollback();
    if (ret < 0)
        printf("tx_rollback is failed. [%s]\n", tpstrerror(tperrno));
} else
    printf("tx_commit success.\n");
resource_free(sndbuf, rcvbuf);

```

```

    return 0;
}

resource_free(char* sndbuf, char *rcvbuf)
{
    if (rcvbuf != NULL)
        tpfree((char*)rcvbuf);

    if (sndbuf != NULL)
        tpfree((char*)sndbuf);

    tpend();
}

```

サーバー・プログラム

以下は、サーバー・プログラムの例です。

<svr_db2.sqc>

```

#include <stdio.h>
#include <usrinc/atmi.h>

EXEC SQL INCLUDE SQLCA;

EXEC SQL begin declare section;
    sqlint32 h_empno;
    sqlint32 h_count;
EXEC SQL end declare section;

XASERVICE2(TPSVCINFO *msg)
{
    char          *res_msg;
    int   h_count=0;

    h_empno = atoi(msg->data);
    printf("h_empno = %d \n", h_empno);

    EXEC SQL
        INSERT into EMP (EMPNO) VALUES (:h_empno);
    if (sqlca.sqlcode != 0) {
        printf("insertion is failed : sqlcode[%d]%d\n", sqlca.sqlcode,
sqlca.sqlstate);
        tpreturn(TPFAIL, -1, 0, 0, 0);
    } else

    EXEC SQL SELECT COUNT(*)

```

```

        INTO      :h_count
        FROM      emp
        WHERE     empno = :h_empno;

        if (sqlca.sqlcode != 0) {
            printf("insertion is failed : sqlcode[%d]%d\n", sqlca.sqlcode,
sqlca.sqlstate);
            tpreturn(TPFAIL, -1, 0, 0, 0);
        } else
            printf("insertion is success. selcnt(%d) \n", h_count);

        tpreturn(TPSUCCESS, 0, 0, 0, 0);
    }

```

サーバー・メイクファイル

以下は、サーバー・メイクファイルの例です。

```

# Server makefile for DB2
# Linux 64bit

DB2LIBDIR = $(DB2_HOME)/lib
DB2LIBS    = -ldb2

DB          = TEST
DB2USER     = tmaxha
DB2PASS     = ha0115

TARGET      = $(COMP_TARGET)
APOBJS      = $(TARGET).o
APOBJS2     = utilemb.o
NSDLOBJ     = $(TMAXDIR)/lib64/sdl.o

#OBS        = $(APOBJS) $(APOBJS2) $(SVCTOBJ)
OBS         = $(APOBJS) $(SVCTOBJ)
SVCTOBJ     = $(TARGET)_svctab.o

#CFLAGS     = -m64 -O -I$(TMAXDIR) -I$(DB2_HOME)/include
CFLAGS      = -O -I$(TMAXDIR) -I$(DB2_HOME)/include
LDFLAGS     =

TMAXAPPPDIR = $(TMAXDIR)/appbin
TMAXSVCTDIR = $(TMAXDIR)/svct
TMAXLIBDIR  = $(TMAXDIR)/lib64

TMAXLIBS    = -lsvr -ldb2s          # dynmic XAOPTION=DYNAMIC (DB2 Client
v8.0(below) 32bit or DB2 Client 64bit & Windows)

```



```

#TMAXLIBS      = -lsvr -ldb2_64s      # dynmic XAOPTION=DYNAMIC (DB2 Client
v8.0(below) 64bit & Linux/Unix)
#TMAXLIBS      = -lsvr -ldb2s_static      #static XAOPTION=none (DB2 Client
v9.0(above))
#
.SUFFIXES : .c

.c.o:
    $(CC) $(CFLAGS) $(LDFLAGS) -c $<

#
# server compile
#
all: $(TARGET)

$(TARGET): $(OBSJS)
    $(CC) $(CFLAGS) $(LDFLAGS) -L$(TMAXLIBDIR) -o $(TARGET) -L$(DB2LIBDIR)
$(DB2LIBS) $(OBSJS) $(TMAXLIBS) $(NSDLOBJ)
    mv $(TARGET) $(TMAXAPPDIR)
    rm -f $(OBSJS)

$(APOBJS): $(TARGET).sqc
    db2 connect to $(DB) user $(DB2USER) using $(DB2PASS)
    db2 prep $(TARGET).sqc bindfile
    db2 bind $(TARGET).bnd
    db2 connect reset
    db2 terminate
    $(CC) $(CFLAGS) $(LDFLAGS) -c $(TARGET).c

$(SVCTOBJ):
    cp -f $(TMAXSVCTDIR)/$(TARGET)_svctab.c .
    touch ./$(TARGET)_svctab.c
    $(CC) $(CFLAGS) -c ./$(TARGET)_svctab.c

#
clean:
    :-rm -f *.o core $(TMAXAPPDIR)/$(TARGET) $(TARGET).bnd

```

TMSメイクファイル

以下は、TMSメイクファイルの例です。

```

# TMS Makefile for DB2
# Linux 64bit

TARGET    = tms_db2
APOBJ     = dummy.o

```

```

APPDIR = $(TMAXDIR)/appbin
TMAXLIBD= $(TMAXDIR)/lib64
TMAXLIBS = -lsvr -ldb2s          # dynmic XAOPTION=DYNAMIC (DB2 Client
v8.0(below) 32bit or DB2 Client 64bit & Windows)
#TMAXLIBS = -lsvr -ldb2_64s      # dynmic XAOPTION=DYNAMIC (DB2 Client
v8.0(below) 64bit & Linux/Unix)
#TMAXLIBS = -lsvr -ldb2s_static   #static XAOPTION=none (DB2 Client
v9.0(above))

DB2PATH = $(DB2_HOME)
DB2LIBDIR= $(DB2PATH)/lib
DB2LIB = -ldb2

CFLAGS =
LDFLAGS =
SYSLIBS =

all: $(TARGET)

$(TARGET): $(APOBJ)
    $(CC) $(CFLAGS) $(LDFLAGS) -o $(TARGET) -L$(TMAXLIBD) $(TMAXLIBS) $(APOBJ)
    -L$(DB2LIBDIR) $(DB2LIB) $(SYSLIBS)
    mv $(TARGET) $(APPDIR)/.

$(APOBJ):
    $(CC) $(CFLAGS) -c dummy.c
#
clean:
    -rm -f *.o core $(APPDIR)/$(TARGET)

```

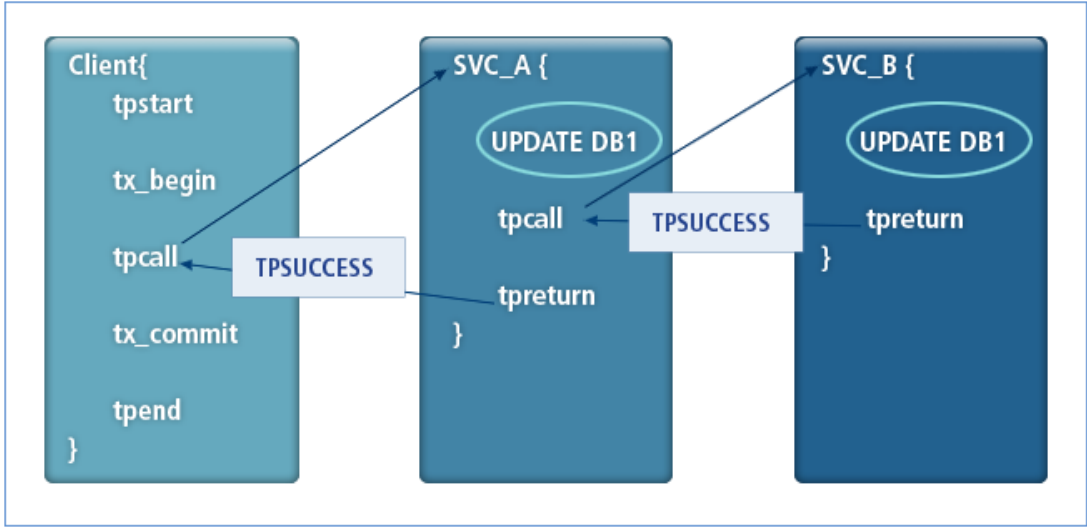
5.4. データベース連携プログラム

アプリケーションの開発時に応用して使用できる例を通して、実際のプログラミングの手法を説明します。データベースとの連携は、同一機種のデータベースと異機種データベースに分かれます。

5.4.1. 同期型モード(同機種)

以下は、同期型モードで同機種のデータベースに接続する場合のプログラムのフローを表した図です。

[図 5.2] 同期型モードのフロー図(同機種のデータベース接続)



プログラム構成

● 共通プログラム

プログラム・ファイル	説明
demo.s	SDLFILEです
sample.m	Tmax環境ファイルです
tmax.env	環境ファイルです
mktable.sql	データベース・テーブルを作成するSQLスクリプトです

● クライアント・プログラム

プログラム・ファイル	説明
client.c	クライアント・プログラムです

● サーバー・プログラム

プログラム・ファイル	説明
update.pc, insert.pc	サーバー・プログラムです
Makefile	Tmaxが提供するMakefileを修正する必要があります

プログラムの特徴

● クライアント・プログラム

区分	説明
Tmax接続	NULLパラメータで接続
バッファ・タイプ	STRUCT
サブタイプ	input構造体ファイルをsdlcでコンパイルし、SDLファイルの作成が必要
トランザクション可否	クライアントでトランザクションを指定

- サーバー・プログラム

区分	説明
サービス数	UPDATEサービスでINSERTサービスを要求
データベース接続	Oracleデータベースを使用し、Tmax環境ファイルのSVRGROUPセクションにデータベース情報を指定

プログラム環境

区分	説明
システム	SunOS 5.7 32bit
データベース	Oracle 8.0.5

構造体バッファ

以下は、構造体バッファの例です。

<demo.s>

```
struct input {
    int     account_id;
    int     branch_id;
    char    phone[15];
    char    address[61];
};
```

Tmax環境ファイル

以下は、Tmax環境ファイルの例です。

<sample.m>

```
*DOMAIN
res    SHMKEY=88000, MINCLH=1, MAXCLH=5, TPORTNO=8880, BLOCKTIME=60
```

```

*NODE
tmax          TMAXDIR = "/user/ tmax ",
              APPDIR  = "/user/ tmax /appbin",
              PATHDIR = "/user/ tmax /path",
              TLOGDIR = "/user/ tmax /log/tlog",
              ULOGDIR="/user/ tmax /log/ulog",
              SLOGDIR="/user/ tmax /log/slog"

*SVRGROUP
svg1          NODENAME = tmax, DBNAME = ORACLE,
              OPENINFO = "ORACLE_XA+Acc=P/bmt/bmt+SesTm=60",
              TMSNAME  = svg1_tms

*SERVER
update        SVGNAME=svg1
insert        SVGNAME=svg1

*SERVICE
UPDATE        SVRNAME=update
INSERT        SVRNAME=insert

```

以下は、環境ファイルの例です。

<tmax.env>

```

[tmax]
TMAX_HOST_ADDR=192.168.1.39
TMAX_HOST_PORT=8880
SDLFILE=/user/tmax/sample/sdl/tmax.sdl
TMAX_CONNECT_TIMEOUT=5

```

データベース・スクリプト

以下は、データベース・テーブルを作成するSQLスクリプトの例です。

<mktable.sql>

```

$ORACLE_HOME/bin/sqlplus bmt/bmt <<!
drop table ACCOUNT;
create table ACCOUNT (
    ACCOUNT_ID integer,
    BRANCH_ID integer not null,
    SSN        char(13) not null,
    BALANCE     number,
    ACCT_TYPE   char(1),
    LAST_NAME   char(21),
    FIRST_NAME  char(21),
    MID_INIT    char(1),

```

```

        PHONE      char(15),
        ADDRESS    char(61),
        CONSTRAINT ACCOUNT_PK PRIMARY KEY(ACCOUNT_ID)
);
quit
!
```

クライアント・プログラム

以下は、クライアント・プログラムの例です。

<client.c>

```

#include <stdio.h>
#include <usrinc/atmi.h>
#include "../sdl/demo.s"

#define TEMP_PHONE "6283-2114"
#define TEMP_ADDRESS "Korea"

int main(int argc, char *argv[])
{
    struct input *sndbuf;
    char      *rcvbuf;
    int acct_id, n, timeout;
    long      len;

    if (argc != 2) {
        fprintf(stderr, "Usage:%s account_id \n", argv[0]);
        exit(1);
    }

    acct_id = atoi(argv[1]);
    timeout = 5;

    n = tmaxreadenv("tmax.env", "tmax");
    if (n < 0) {
        fprintf(stderr, "tmaxreadenv fail! tperrno = %d\n", tperrno);
        exit(1);
    }

    n = tpstart((TPSTART_T *)NULL);
    if (n < 0) {
        fprintf(stderr, "tpstart fail! tperrno = %s\n", tperrno);
        exit(1);
    }
}
```

```

sndbuf = (struct input *)tpalloc("STRUCT", "input", `
    sizeof(struct input));
if (sndbuf == NULL) {
    fprintf(stderr, "tpalloc fail: sndbuf tperrno = %d\n", tperrno);
    tpend();
    exit(1);
}

rcvbuf = (char *)tpalloc("STRING", NULL, 0);
if (rcvbuf == NULL) {
    fprintf(stderr, "tpalloc fail: rcvbuf tperrno = %d\n", tperrno);
    tpend();
    exit(1);
}

sndbuf->account_id = acct_id;
sndbuf->branch_id = acct_id;
strcpy(sndbuf ->phone, TEMP_PHONE);
strcpy(sndbuf ->address, TEMP_ADDRESS);

tx_set_transaction_timeout(timeout);
n = tx_begin();
if (n < 0)
    fprintf(stderr, "tx begin fail! tperrno = %d\n", tperrno);

n = tpcall("UPDATE", (char *)sndbuf, sizeof(struct input),
    (char **)&rcvbuf, (long *)&len, TPNOFLAGS);
if (n < 0) {
    fprintf(stderr, "tpcall fail! tperrno = %d\n", tperrno);
    tpend();
    exit(1);
}

n = tx_commit();
if (n < 0) {
    fprintf(stderr, "tx commit fail! tx error = %d \n", n);
    tx_rollback();
    tpend();
    exit(1);
}

printf("rtn msg = %s\n", rcvbuf);
tpfree((char *)sndbuf);
tpfree((char *)rcvbuf);
tpend();
}

```

サーバー・プログラム

以下は、データベースに更新するサーバー・プログラムの例です。

<update.pc>

```
#include <stdio.h>
#include <ctype.h>
#include <usrinc/atmi.h>
#include <usrinc/sdl.h>
#include "../sdl/demo.s"

#define OKMSG "YOU COMPLETE THE TRANSACTION"

EXEC SQL include sqlca.h;
EXEC SQL BEGIN DECLARE SECTION;
    int      account_id;
    int      branch_id;
    char      ssn[15];
    char      phone[15];
    char      address[61];
EXEC SQL END DECLARE SECTION;

UPDATE(TPSVCINFO *msg)
{
    struct input *rcvbuf;
    int      ret;
    long      acnt_id, rcvlen;
    char      *send;

    rcvbuf = (struct input *) (msg->data);
    send = (char *) tmalloc("STRING", NULL, 0);
    if (send == NULL) {
        fprintf(stderr, "tpalloc fail errno = %s\n", strerror(tperrno));
        tpreturn(TPFAIL, 0, (char *) NULL, 0, 0);
    }
    account_id = rcvbuf->account_id;
    branch_id = rcvbuf->branch_id;
}
```

以下は、データベースに挿入するサーバー・プログラムの例です。

<insert.pc>

```
#include <stdio.h>
#include <ctype.h>
#include <usrinc/atmi.h>
```



```

#include <usrinc/sdl.h>
#include "../sdl/demo.s"

#define OKMSG "YOU COMPLETE THE TRANSACTION"

EXEC SQL include sqlca.h;
EXEC SQL BEGIN DECLARE SECTION;
    int      account_id;
    int      branch_id;
    char      ssn[15];
    char      phone[15];
    char      address[61];
EXEC SQL END DECLARE SECTION;

INSERT(msg)
TPSVCINFO *msg;
{
    struct input *rcvbuf;
    int      ret;
    long      acnt_id;
    char      *send;

    rcvbuf = (struct input *) (msg->data);
    send = (char *) tmalloc("STRING", NULL, 0);
    if (send == NULL) {
        fprintf(stderr, "tpalloc fail errno = %s\n", tpstrerror(tperrno));
        tpreturn(TPFAIL, 0, (char *) NULL, 0, TPNOFLAGS);
    }

    account_id = rcvbuf->account_id;
    branch_id = rcvbuf->branch_id;
    strcpy(phone, rcvbuf->phone);
    strcpy(address, rcvbuf->address);
    strcpy(ssn, "1234567");

    /* Declare && Open Cursor for Fetch */
    EXEC SQL INSERT INTO ACCOUNT (
        ACCOUNT_ID,
        BRANCH_ID,
        SSN,
        PHONE,
        ADDRESS )
    VALUES (:account_id, :branch_id, :ssn, :phone, :address);

    if (sqlca.sqlcode != 0 && sqlca.sqlcode != 1403 )
    {
        printf("insert failed sqlcode = %d\n", sqlca.sqlcode);
    }
}

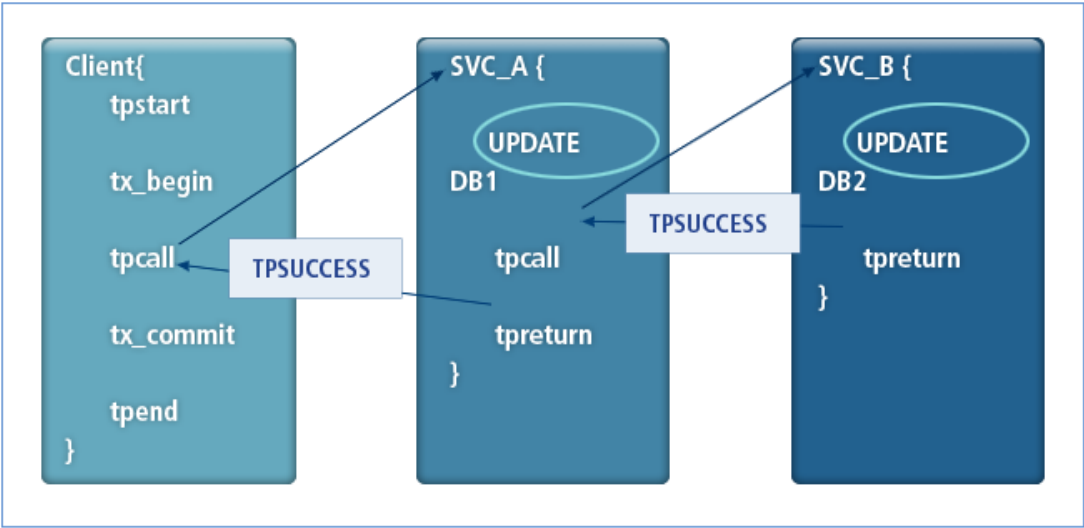
```

```
        tpreturn(TPFAIL, -1, (char *)NULL, 0, TPNOFLAGS);
    }
    strcpy(send, OKMSG);
    tpreturn(TPSUCCESS, 1, (char *)send, strlen(send), TPNOFLAGS);
}
```

5.4.2. 同期型モード(異機種)

以下は、同期型モードで異機種のデータベースに接続する場合のプログラムのフローを表した図です。

[図 5.3] 同期型モードのフロー図(異機種のデータベース接続)



プログラム構成

- 共通プログラム

プログラム・ファイル	説明
demo.s	SDLFILEです
sample.m	Tmax環境ファイルです
tmax.env	環境ファイルです
mktable.sql	データベース・テーブルを作成するSQLスクリプトです

- クライアント・プログラム

構成	説明
client.c	クライアント・プログラムです

- サーバー・プログラム

構成	説明
update.pc, insert.pc	サーバー・プログラムです
Makefile	Tmaxが提供するメイクファイルを変更する必要があります

参考

クライアントとサーバー・プログラムは、「[5.4.1. 同期型モード\(同機種\)](#)」の場合と同じです。マルチノード環境設定についての詳細内容は、『Tmax 運用ガイド』を参照してください。

プログラムの特徴

- クライアント・プログラム

区分	説明
Tmax接続	NULLパラメータで接続
バッファ・タイプ	STRUCT
サブタイプ	input構造体ファイルをsdlcでコンパイルし、SDLファイルの作成が必要
トランザクション可否	クライアントで明示的にトランザクション範囲を指定

- サーバー・プログラム

区分	説明
サービス数	UPDATEサービスでINSERTサービスを要求
データベース接続	Oracleデータベースを使用し、システム構成ファイルのSVRGROUPセクションにデータベース情報を指定

プログラム環境

区分	説明
システム	SunOS 5.7 32bit, SunOS 5.8 32bit
データベース	Oracle 8.0.5

Tmax環境ファイル

以下は、Tmax環境ファイルの例です。

<sample.m>

```
*DOMAIN
res          SHMKEY=88000, MINCLH=1, MAXCLH=5, TPORTNO=8880, BLOCKTIME=60

*NODE
tmax1        TMAXDIR="/user/ tmax ",
              APPDIR="/user/ tmax /appbin",
              PATHDIR = "/user/ tmax /path",
              TLOGDIR = "/user/ tmax /log/tlog",
              ULOGDIR="/user/ tmax /log/ulog",
              SLOGDIR="/user/ tmax /log/slog"

tmax2        TMAXDIR="/user/ tmax ",
              APPDIR="/user/ tmax /appbin",
              PATHDIR = "/user/ tmax /path",
              TLOGDIR = "/user/ tmax /log/tlog",
              ULOGDIR="/user/ tmax /log/ulog",
              SLOGDIR="/user/ tmax /log/slog"

*SVRGROUP
svg1         NODENAME = tmax1, DBNAME = ORACLE,
              OPENINFO = "ORACLE_XA+Acc=P/bmt/bmt+SesTm=60",
              TMSNAME = svg1_tms

svg2         NODENAME = tmax2, DBNAME = ORACLE,
              OPENINFO = "ORACLE_XA+Acc=P/bmt/bmt+SesTm=60",
              TMSNAME = svg2_tms

*SERVER
update       SVGNAME=svg1
insert       SVGNAME=svg2

*SERVICE
UPDATE       SVRNAME=update
INSERT       SVRNAME=insert
```

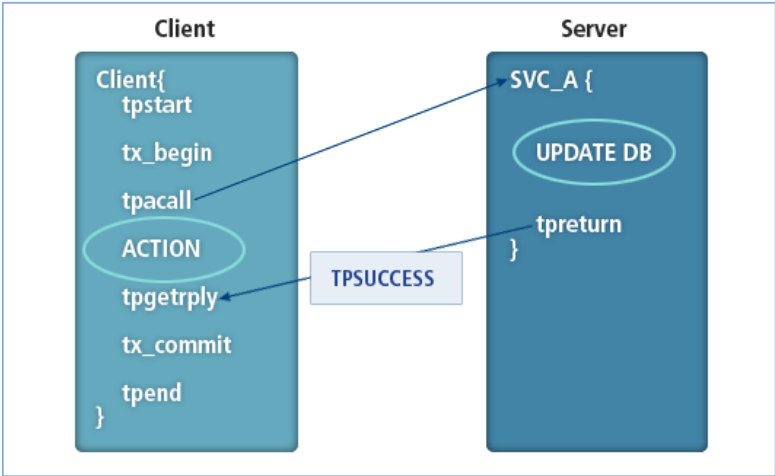
以下は、環境ファイルの例です。

```
[tmax1]
TMAX_HOST_ADDR=192.168.1.39
TMAX_HOST_PORT=8880
SDLFILE=/user/tmax/sample/sdl/tmax.sdl
TMAX_CONNECT_TIMEOUT=5
```

5.4.3. 非同期型モード(同機種)

以下は、非同期型モードで同機種のデータベースに接続する場合のプログラムのフローを表した図です。

[図 5.4] 非同期型モードのフロー図(同機種のデータベース接続)



プログラム構成

● 共通プログラム

プログラム・ファイル	説明
demo.s	SDLFILEです
sample.m	Tmax環境ファイルです
tmax.env	環境ファイルです
mktable.sql	データベース・テーブルを作成するSQLスクリプトです

● クライアント・プログラム

プログラム・ファイル	説明
client.c	クライアント・プログラムです

● サーバー・プログラム

プログラム・ファイル	説明
update.pc	サーバー・プログラムです
Makefile	Tmaxが提供するメイクファイルを変更する必要があります

プログラムの特徴

- クライアント・プログラム

機能	説明
Tmax接続	NULLパラメータで接続
バッファ・タイプ	STRUCT
サブタイプ	input構造体ファイルをsdhcでコンパイルし、SDLファイルの作成が必要
トランザクション可否	クライアントでトランザクションを指定

- サーバー・プログラム

機能	説明
サービス数	UPDATEサービスを要求
データベース接続	Oracleデータベースを使用し、システム構成ファイルのSVRGROUPセクションにデータベース情報を指定

プログラム環境

区分	説明
システム	SunOS 5.7 32bit
データベース	Oracle 8.0.5

構造体バッファ

以下は、構造体バッファの例です。

<demo.s>

```
struct input {
    int    account_id;
    int    branch_id;
    char   phone[15];
    char   address[61];
};
```

Tmax環境ファイル

以下は、Tmax環境ファイルの例です。

<sample.m>

```
*DOMAIN
res          SHMKEY=88000, MINCLH=1, MAXCLH=5, TPORTNO=8880, BLOCKTIME=60

*NODE
tmax         TMAXDIR="/user/ tmax ",
             APPDIR="/user/ tmax /appbin",
             PATHDIR = "/user/ tmax /path",
             TLOGDIR = "/user/ tmax /log/tlog",
             ULOGDIR="/user/ tmax /log/ulog",
             SLOGDIR="/user/ tmax /log/slog"

*SVRGROUP
svg1         NODENAME = tmax, DBNAME = ORACLE,
             OPENINFO = "ORACLE_XA+Acc=P/bmt/bmt+SesTm=60",
             TMSNAME = svg1_tms

*SERVER
update       SVGNAME=svg1

*SERVICE
UPDATE       SVRNAME=update
```

以下は、環境ファイルの例です。

<tmax.env>

```
[tmax]
TMAX_HOST_ADDR=192.168.1.39
TMAX_HOST_PORT=8880
SDLFILE=/user/tmax/sample/sdl/tmax.sdl
TMAX_CONNECT_TIMEOUT=5
```

データベース・スクリプト

以下は、データベースにテーブルを作成するSQLスクリプトの例です。

<mktable.sql>

```
$ORACLE_HOME/bin/sqlplus bmt/bmt <<!
drop table ACCOUNT;
create table ACCOUNT (
    ACCOUNT_ID integer,
```

```

        BRANCH_ID    integer not null,
        SSN          char(13) not null,
        BALANCE       number,
        ACCT_TYPE     char(1),
        LAST_NAME     char(21),
        FIRST_NAME    char(21),
        MID_INIT      char(1),
        PHONE         char(15),
        ADDRESS       char(61),
        CONSTRAINT ACCOUNT_PK PRIMARY KEY(ACCOUNT_ID)
);
quit
!
```

クライアント・プログラム

以下は、クライアント・プログラムの例です。

<client.c>

```

#include <stdio.h>
#include <usrinc/atmi.h>
#include "../sdl/demo.s"

#define TEMP_PHONE "6283-2114"
#define TEMP_ADDRESS "Korea"

int main(int argc, char *argv[])
{
    struct input *sndbuf;
    char *rcvbuf;
    int acct_id, n, cd, timeout;
    long len;

    if (argc != 2) {
        fprintf(stderr, "Usage:%s account_id \n", argv[0]);
        exit(1);
    }

    acct_id = atoi(argv[1]);
    timeout = 5;

    n = tmaxreadenv("tmax.env", "tmax");
    if (n < 0) {
        fprintf(stderr, "tmaxreadenv fail! tperrno = %d\n", tperrno);
        exit(1);
    }

    n = tpstart((TPSTART_T *)NULL);
    if (n < 0) {
```



```

        fprintf(stderr, "tpstart fail! tperrno = %s\n", tperrno);
        exit(1);
    }

    sndbuf = (struct input *)tpalloc("STRUCT", "input", sizeof(struct input));
    if (sndbuf == NULL) {
        fprintf(stderr, "tpalloc fail: sndbuf tperrno = %d\n", tperrno);
        tpend();
        exit(1);
    }

    rcvbuf = (char *)tpalloc("STRING", NULL, 0);
    if (rcvbuf == NULL) {
        fprintf(stderr, "tpalloc fail: rcvbuf tperrno = %d\n", tperrno);
        tpend();
        exit(1);
    }

    sndbuf->account_id = acnt_id;
    sndbuf->branch_id = acnt_id;
    strcpy(sndbuf->phone, TEMP_PHONE);
    strcpy(sndbuf->address, TEMP_ADDRESS);
    tx_set_transaction_timeout(timeout);
    n = tx_begin();
    if (n < 0)
        fprintf(stderr, "tx begin fail! tperrno = %d\n", tperrno);

    cd = tpacall("UPDATE", (char *)sndbuf, sizeof(struct input), TPNOFLAGS);
    if (cd < 0) {
        fprintf(stderr, "tpacall fail! tperrno = %d\n", tperrno);
        tpend();
        exit(1);
    }

    n = tpgetrply(&cd, (char **)&rcvbuf, (long *)&len, TPNOFLAGS);
    if (n < 0) {
        fprintf(stderr, "tpgetrply fail! tperrno = %d\n", tperrno);
        tpend();
        exit(1);
    }

    n = tx_commit();
    if (n < 0) {
        fprintf(stderr, "tx commit fail! tx error = %d\n", n);
        tx_rollback();
        tpend();
        exit(1);
    }

    printf("rtn msg = %s\n", rcvbuf);
    tpfree((char *)sndbuf);
    tpfree((char *)rcvbuf);
    tpend();
}

```

サーバー・プログラム

以下は、サーバー・プログラムの例です。

<update.pc>

```
#include <stdio.h>
#include <ctype.h>
#include <usrinc/atmi.h>
#include "../sdl/demo.s"

#define OKMSG "YOU COMPLETE THE TRANSACTION"

EXEC SQL include sqlca.h;

EXEC SQL BEGIN DECLARE SECTION;
    int      account_id;
    int      branch_id;
    char      ssn[15];
    char      phone[15];
    char      address[61];
EXEC SQL END DECLARE SECTION;

UPDATE(TPSVCINFO *msg)
{
    struct input *rcvbuf;
    int      ret, cd;
    long      acnt_id, rcvlen;
    char      *send;

    rcvbuf = (struct input *) (msg->data);
    send = (char *) tmalloc("STRING", NULL, 0);
    if (send == NULL) {
        fprintf(stderr, "tpalloc fail errno = %s\n", strerror(tperrno));
        tpreturn(TPFAIL, 0, (char *) NULL, 0, TPNOFLAGS);
    }
    account_id = rcvbuf->account_id;
    branch_id = rcvbuf->branch_id;
    strcpy(phone, rcvbuf->phone);
    strcpy(address, rcvbuf->address);
    strcpy(ssn, "1234567");

    EXEC SQL UPDATE ACCOUNT
    SET BRANCH_ID = :branch_id,
    PHONE = :phone,
    ADDRESS = :address,
    SSN = :ssn
    WHERE ACCOUNT_ID = :account_id;

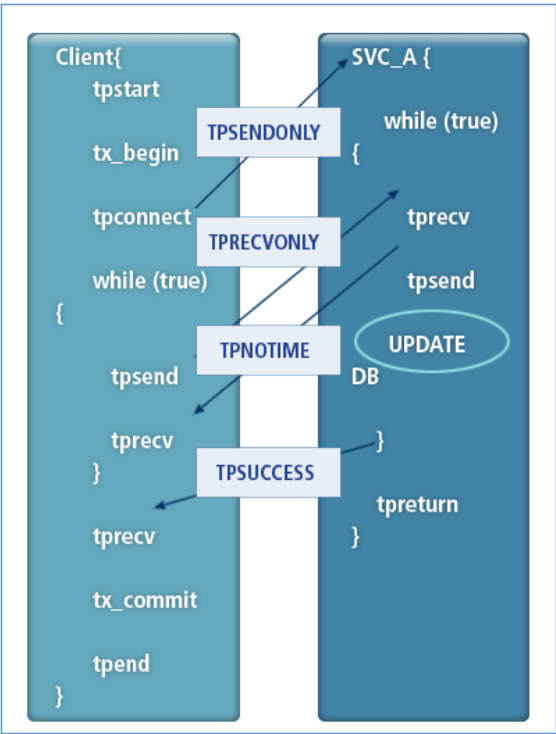
    if (sqlca.sqlcode != 0 && sqlca.sqlcode != 1403 ) {
        fprintf(stderr, "update failed sqlcode = %d\n", sqlca.sqlcode);
        tpreturn(TPFAIL, -1, (char *) NULL, 0, TPNOFLAGS);
    }
}
```

```
strcpy(send, OKMSG);
tpreturn(TPSUCCESS, 1, (char *)send, strlen(send), TPNOFLAGS);
}
```

5.4.4. 会話型モード(同機種)

以下は、会話型モードで同機種のデータベースに接続する場合のプログラムのフローを表した図です。

[図 5.5] 会話型モードのフロー図(同機種のデータベース接続)



プログラム構成

- 共通プログラム

プログラム・ファイル	説明
demo.s	SDLFILEです
sample.m	Tmax環境ファイルです
tmax.env	環境ファイルです
mktable.sql	データベース・テーブルを作成するSQLスクリプトです

- クライアント・プログラム

プログラム・ファイル	説明
client.c	クライアント・プログラムです

- サーバー・プログラム

プログラム・ファイル	説明
update.pc	サーバー・プログラムです
Makefile	Tmaxが提供するメイクファイルを変更する必要があります

プログラムの特徴

- クライアント・プログラム

機能	説明
Tmax接続	NULLパラメータ接続
バッファ・タイプ	STRUCT
サブタイプ	input構造体ファイルをsdicでコンパイルし、SDLファイルの作成が必要(アプリケーションを実行させるために必要)
トランザクション可否	クライアントでトランザクション指定

- サーバー・プログラム

機能	説明
サービス数	UPDATEサービスを要求
データベース接続	Oracleデータベースを指定し、Tmax環境ファイルのSVRGROUPセクションにデータベース情報を指定

プログラム環境

区分	説明
システム	SunOS 5.7 32bit
データベース	Oracle 8.0.5

構造体バッファ

以下は、構造体バッファの例です。

<demo.s>

```
struct input {
    int      account_id;
    int      branch_id;
    char     phone[15];
    char     address[61];
};
```

Tmax環境ファイル

以下は、Tmax環境ファイルの例です。

```
*DOMAIN
res    SHMKEY=88000, MINCLH=1, MAXCLH=5, TPORTNO=8880, BLOCKTIME=60

*NODE
tmax    TMAXDIR="/user/ tmax ",
        APPDIR="/user/ tmax /appbin",
        PATHDIR = "/user/ tmax /path",
        TLOGDIR = "/user/ tmax /log/tlog",
        ULOGDIR="/user/ tmax /log/ulog",
        SLOGDIR="/user/ tmax /log/slog"

*SVRGROUP
svg1    NODENAME = tmax, DBNAME = ORACLE,
        OPENINFO = "ORACLE_XA+Acc=P/bmt/bmt+SesTm=60",
        TMSNAME = svg1_tms

*SERVER
update    SVGNAME=svg1, CONV=YES

*SERVICE
UPDATE    SVRNAME= update
```

以下は、環境ファイルの例です。

<tmax.env>

```
[ tmax ]
TMAX_HOST_ADDR=192.168.1.39
TMAX_HOST_PORT=8880
SDLFILE=/user/tmax/sample/sdl/tmax.sdl
TMAX_CONNECT_TIMEOUT=5
```

データベース・スクリプト

以下は、データベースにテーブルを作成するSQLスクリプトの例です。

<mktable.sql>

```
$ORACLE_HOME/bin/sqlplus bmt/bmt <<!  
drop table ACCOUNT;  
create table ACCOUNT (  
    ACCOUNT_ID integer,  
    BRANCH_ID integer not null,  
    SSN char(13) not null,  
    BALANCE number,  
    ACCT_TYPE char(1),  
    LAST_NAME char(21),  
    FIRST_NAME char(21),  
    MID_INIT char(1),  
    PHONE char(15),  
    ADDRESS char(61),  
    CONSTRAINT ACCOUNT_PK PRIMARY KEY(ACCOUNT_ID)  
);  
quit  
!
```

クライアント・プログラム

以下は、クライアント・プログラムの例です。

<client.c>

```
#include <stdio.h>  
#include <usrinc/atmi.h>  
#include "../sdl/demo.s"  
  
#define TEMP_PHONE "6283-2115"  
#define TEMP_ADDRESS "Korea"  
  
void main(int argc, char *argv[])  
{  
    struct input *sndbuf;  
    char *rcvbuf;  
    int acntid, timeout;  
    long revent, rcvlen;  
    int cd, n;  
  
    if (argc != 2) {  
        fprintf(stderr, "Usage:%s acntid\n", argv[0]);  
    }
```

```

        exit(1);
    }

    acntid = atoi(argv[1]);
    timeout = 5;
    n = tmaxreadenv("tmax.env", "tmax");
    if (n < 0) {
        fprintf(stderr, "tmaxreadenv fail tperrno = %d\n", tperrno);
        exit(1);
    }

    n = tpstart((TPSTART_T *)NULL);
    if (n < 0) {
        fprintf(stderr, "tpstart fail tperrno = %s\n", tperrno);
        exit(1);
    }
    printf("tpstart ok!\n");
    sndbuf = (struct input *)tpalloc("STRUCT", "input", sizeof(struct input));

    if (sndbuf == NULL) {
        fprintf(stderr, "tpalloc fail: sndbuf tperrno = %d\n", tperrno);
        tpend();
        exit(1);
    }

    rcvbuf = (char *)tpalloc("CARRAY", NULL, 0);
    if (rcvbuf == NULL) {
        fprintf(stderr, "tpalloc fail: rcvbuf tperrno = %d\n", tperrno);
        tpend();
        exit(1);
    }

    sndbuf->account_id = acntid;
    sndbuf->branch_id = acntid;
    strcpy(sndbuf->phone, TEMP_PHONE);
    strcpy(sndbuf->address, TEMP_ADDRESS);

    tx_set_transaction_timeout(timeout);
    n = tx_begin();
    if (n < 0)
        fprintf(stderr, "tx begin fail tx error = %d\n", n);
    printf("tx begin ok!\n");

    cd = tpconnect("UPDATE", (char *)sndbuf, 0, TPSENDONLY);
    if (cd < 0) {
        fprintf(stderr, "tpconnect fail tperrno = %d\n", tperrno);
        tpend();
    }

```

```

        exit(1);
    }

    while (1) {
        n = tpsend(cd, (char *)sndbuf, sizeof(struct input), TPRECVONLY,
                  &revent);

        if (n < 0) {
            fprintf(stderr, "tpsend fail revent = 0x%08x\n", revent);
            tx_rollback();
            tpend();
            exit(1);
        }
        printf("tpsend ok\n");

        n = tprecv(cd, (char **)&rcvbuf, (long *)&rcvlen, TPNOTIME, &revent);

        if (n < 0 && revent != TPEV_SENDOONLY) {
            fprintf(stderr, "tprecv fail revent = 0x%08x\n", revent);
            tx_rollback();
            tpend();
            exit(1);
        }
        printf("tprecv ok\n");
        sndbuf->account_id++;

        if (revent != TPEV_SENDOONLY)
            break;
    }
    n = tprecv(cd, (char **)&rcvbuf, (long *)&rcvlen, TPNOTIME, &revent);
    if (n < 0 && revent != TPEV_SVCSUCC) {
        fprintf(stderr, "tprecv fail revent = 0x%08x\n", revent);
        tx_rollback();
        tpend();
        exit(1);
    }
    printf("rcvbuf = [%s]\n", rcvbuf);

    n = tx_commit();
    if (n < 0) {
        fprintf(stderr, "tx commit fail tx error = %d\n", n);
        tx_rollback();
        tpend();
        exit(1);
    }
    printf("tx commit ok!\n");
    printf("rtn msg = %s\n", rcvbuf);
    tpfree((char *)sndbuf);

```



```

        tpfree((char *)rcvbuf);
        tpend();
    }

```

サーバー・プログラム

以下は、サーバー・プログラムの例です。

<update.pc>

```

#include <stdio.h>
#include <ctype.h>
#include <usrinc/atmi.h>
#include "../sdl/demo.s"

void _db_work();

#define OKMSG "YOU COMPLETE THE TRANSACTION"

EXEC SQL include sqlca.h;
EXEC SQL BEGIN DECLARE SECTION;
    int      account_id;
    int      branch_id;
    char      ssn[15];
    char      phone[15];
    char      address[61];
EXEC SQL END DECLARE SECTION;

struct input *rcvbuf;

UPDATE(TPSVCINFO *msg)
{
    int      ret, count;
    long      acnt_id;
    long      revent, rcvlen, flag;
    char      *send;

    rcvbuf = (struct input *)tpalloc("STRUCT", "input", 0);
    send = (char *)tpalloc("CARRAY", NULL, 0);

    count = 1;
    flag = 0;

    while (1) {
        ret = tprecv(msg->cd, (char **)&rcvbuf, &rcvlen, TPNOTIME, &revent);
        if (ret < 0 && revent != TPEV_SENDOONLY) {

```

```

        fprintf(stderr, "tprecv fail revent = 0x%08x\n", revent);
        tpreturn(TPFAIL, -1, (char *)rcvbuf, 0, TPNOFLAGS);
    }
    printf("tprecv ok!\n");

    if (count == 10) {
        flag &= ~TPRECVONLY;
        flag |= TPNOTIME;
    }
    else
        flag |= TPRECVONLY;

    ret = tpsend(msg->cd, (char *)send, strlen(send), flag, &revent);
    if (ret < 0) {
        fprintf(stderr, "tpsend fail revent = 0x%08x\n", revent);
        tpreturn(TPFAIL, -1, (char *)NULL, 0, TPNOFLAGS);
    }
    printf("tpsend ok!\n");
    _db_work();

    /* break after 10 iterations */
    if (count == 10)
        break;

    count++;
}

strcpy(send, OKMSG);
printf("tpreturn ok!\n");
tpreturn(TPSUCCESS, 1, (char *)send, strlen(send), TPNOFLAGS);
}

void _db_work() {
    account_id = rcvbuf->account_id;
    branch_id = rcvbuf->branch_id;
    strcpy(phone, rcvbuf->phone);
    strcpy(address, rcvbuf->address);
    strcpy(ssn, "1234567");

    EXEC SQL UPDATE ACCOUNT
    SET BRANCH_ID = :branch_id,
        PHONE = :phone,
        ADDRESS = :address,
        SSN = :ssn
    WHERE ACCOUNT_ID = :account_id;

    if (sqlca.sqlcode != 0 && sqlca.sqlcode != 1403 )

```

```

    {
        fprintf(stderr, "update failed sqlcode = %d\n", sqlca.sqlcode);
        tpreturn(TPFAIL, -1, (char *)NULL, 0, 0);
    }
}

```

5.5. TIPを利用したプログラム

Tmaxでは、TIP(Tmax Information Provider)を使用して、システム環境情報、統計情報の確認、システム運用管理など、多様な機能を実行できます。TIPは、TIPSVCを処理するためにTmaxが提供する機能プロセスで、TIPを使用して以下の機能を実行できます。

- システム環境情報の確認: システムの静的な環境情報を確認します
- システム統計情報の確認: システムの運用中に各プロセスの状態などを確認します
- システム運用管理: 追加でプロセスを起動または終了します

5.5.1. TIPの構造

TIPSVCを処理するための別途の機能プロセスであるTIPサーバーは、SYS_SVRサーバータイプで、TIPサーバーグループに属します。クライアントあるいはサーバーの要求を受けたTIPサーバーは、その要求をCLH/TMMIに転送し、処理結果を再度要求者に戻します。TIPサーバーは、フィールド・キーをパターンにしてサービスを処理します。要求するクライアントやサーバーは、要求しようとするデータをフィールド・バッファーに記載して要求し、その結果をフィールド・バッファーで受け取ります。

- CHLOG section (ログレベルの変更)

CHLOGは、TMM、CLH、TMS、SVRのログレベルを変更できるセクションです。tmadminで**chlog**を実行するのと同じ動作を実行します。

TIPサービスを呼び出す場合、フィールド・バッファーに以下を設定し、TIPSVCを呼び出します。

項目	説明
TIP_OPERATION (string)	GETと設定します
TIP_SEGMENT (string)	ADMINISTRATIONと設定します
TIP_SECTION (string)	CHLOGと設定します
TIP_MODULE (int)	動的にログを変更したいモジュールを設定する項目で、以下のいずれかを選択します <ul style="list-style-type: none"> – TIP_TMM – TIP_CLH

項目	説明
	<ul style="list-style-type: none"> – TIP_TMS – TIP_SVR
TIP_FLAGS (int)	<p>flagsを設定する項目で、以下のいずれかを選択します</p> <ul style="list-style-type: none"> – TIP_VFLAG – TIP_GFLAG – TIP_NFLAG
TIP_SVRNAME (string)	<p>該当サーバー名を設定する項目で、TIP_FLAGSをTIP_VFLAGと設定した場合のみ設定します</p>
TIP_SVGNAME (string)	<p>該当サーバー・グループ名を設定する項目で、TIP_FLAGSをTIP_GFLAGと設定した場合のみ設定します</p>
TIP_NODENAME (string)	<p>該当ノード名でTIP_FLAGSをTIP_NFLAGと設定した場合のみ設定します</p>
TIP_LOGLVL (string)	<p>変更するログレベルを設定する項目です。以下のいずれかを選択し、必ず小文字で設定します。結果として取得する値は、TIP_ERRORに設定されます</p> <ul style="list-style-type: none"> – compact – basic – detail – debug1 – debug2 – debug3 – debug4
TIP_ERROR (int)	<p>エラー値が設定される項目で、設定されているエラー値は以下のとおりです</p> <ul style="list-style-type: none"> – TIPSVCFAIL : 該当するサービスを正常に処理できなかった場合 – TIPEOS : メモリーの割り当てに失敗した場合 – TIPEBADFLD : TIP_MODULE値を設定していない場合

- CHTRC section

TMSとSPRのTMAX_TRACEを修正し、trace logオプションを変更できるセクションです。tmadminのchtrcと同一動作を実行します。

TIPサービスを呼び出す場合、フィールド・バッファに以下の内容を設定し、TIPSVCを呼び出します。

項目	説明
TIP_OPERATION (string)	GETと設定します
TIP_SEGMENT (string)	ADMINISTRATIONと設定します
TIP_SECTION (string)	CHTRCに設定します
TIP_FLAGS (int)	フラグをを設定する項目で、以下のいずれかを設定します – TIP_PFLAG – TIP_VFLAG – TIP_GFLAG – TIP_NFLAG
TIP_SPRI (int)	spriを設定する項目で、TIP_FLAGSをTIP_PFLAGと設定した場合のみ設定します
TIP_SVGNAME (string)	該当サーバグループ名を設定する項目で、TIP_FLAGSをTIP_GFLAGと設定した場合のみ設定します
TIP_NODENAME (string)	該当ノード名を設定する項目で、TIP_FLAGSをTIP_NFLAGと設定した場合のみ設定します
TIP_SPEC (string)	変更しようとするfilter spec、receiver spec、trigger specを設定する項目です。結果として受け取る値はTIP_ERRORに設定されます
TIP_ERROR (int)	エラー値が設定されます。設定されるエラー値は以下のとおりです – TIPSVCFAIL : 該当サービスを正常に処理できなかった場合 – TIPEOS : メモリーの割り当てに失敗した場合 – TIPEBADFLD : TIP_MODULE値を設定していない場合

以下は、TIPSVCに要求するために含める必要があるデータです。

● 組織(TIP_OPERATION)

設定値	説明
GET	システムの静的な環境情報、統計情報の確認、BOOT/DOWNシステムを運用管理する場合はGETと設定します
SET	システム設定状態を変更する場合、SETと設定します。現在はGETのみ使用されます

- セグメント(TIP_SEGMENT)

どんな機能を実行するかを決定するために使用するフィールドです。以下は、TIP_SEGMENTフィールドに設定できる値です。

設定値	説明
CONFIGURATION	システムの静的な設定情報を確認します
STATISTICS	システムの運用中に統計情報を確認します
ADMINISTRATION	システムの運用管理(起動と終了)を確認します

- セクション(TIP_SECTION)

TIP_SECTIONを設定した場合、処理する項目の詳細を決定します。

設定値	説明
CONFIGURATION	DOMAIN, NODE, SVRGROUP, SERVER, SERVICE, ROUTING, RQ, GATEWAY
STATISTICS	NODE, TPROC, SPR, SERVICE, RQ, TMGW, NTMGW, TMS, TMMS, CLHS, SERVER(SVR)
ADMINISTRATION	BOOT, DOWN, CHLOG, CHTRC

- コマンド(TIP_CMD)

フィールドはTIP_SECTIONがADMINISTRATIONの場合のみ使用されます。

設定値	説明
TIP_BOOT	Tmaxシステムを起動します
TIP_DOWN	Tmaxシステムを終了します

5.5.2. TIPの使用

TIPの使用方法和エラーの確認方法について説明します。

TIPの使用方法

- 環境設定

TIPサーバーはTmaxが提供する機能プロセスであるため、ユーザーがサービスを作成する必要がありません。但し、環境ファイルにTIPサーバーを登録する必要があります。以下は、環境設定の例です。

```

*DOMAIN
res                ..., TIPSVC = TIPSVC

*NODE
tmaxsl            ...

*SVRGROUP
tsvg              ..., SVGTYPE = TIP

*SERVER
TIP               SVGNAME = tsvg, SVRTYPE = SYS_SVR

```

- DOMAINセクションにはTIPSVCを登録します。登録しない場合、デフォルト値としてTIPSVCが登録されます。
- SVRGROUPセクションにはTIPサーバーグループ(SVGTYPE=TIP)を登録します。
- SERVERセクションにはTIPサーバー(SVRTYPE=SYS_SVR)を登録します。

● システム接続

Tmaxシステムの接続時、TPSTART_T構造体のusername項目にtpadminを設定する必要があります。TIP_SEGMENTがADMINISTRATIONの場合のみusernameを設定し、他の場合は設定しなくても構いません。設定されたusernameが正しくない場合、TIP_ERRORフィールドにTIPEAUTHエラーが設定されます。

```
strcpy(tpinfo->username, ".tpadmin");
```

● バッファの割り当て

TIPサーバーはフィールド・キーをパターンにしてサービス进行处理するため、要求するクライアントやサーバー・プログラムはフィールド・キー・バッファを割り当てる必要があります。

● TIP要求項目の設定

項目	説明
TIP_OPERATION	システム環境情報の変更及び確認をする場合に設定します
TIP_SEGMENT	システム情報の確認及び運用管理のために設定します
TIP_SECTION	SEGMENTで設定項目に対する詳細処理を設定します
TIP_CMD	TIP_SEGMENTがADMINISTRATIONの場合のみ設定します
TIP_NODENAME	マルチノードの場合のみ設定します。単一ノードの場合、指定しなければデフォルト値としてローカルノードが設定されます。設定が正しくない場合、TPEINVALエラーが発生します

- TIPSVCの要求

TIPの要求項目を設定した場合、設定したフィールド・バッファをsndbufとし、tpcallあるいはtpacallを利用してTIPSVCに要求を行います。クライアントとサーバーですべて要求でき、トランザクションはサポートしません。

- 結果の受信

受信フィールド・キー・バッファにサービス結果が保存されます。

エラー 確認

- 正常処理された場合

受信フィールド・キー・バッファのTIP_ERROR項目に0が設定されます。

- エラーが発生した場合

エラー	説明
TIP_STATUS	TIP_ERRORに設定されたエラー内容の詳細を確認できます
TIP_BADFIELD	エラーを引き起こしたフィールドを確認できます
TIP_ERROR	受信フィールド・キー・バッファのTIP_ERROR項目に0より大きい値が設定されます。これは、/usrinc/tip.hで確認できます

以下は、TIP_ERRORに設定されているエラー値についての説明です。

設定値	説明
TIPNOERROR	エラーが発生しません
TIPEBADFLD	有効でないフィールド・キーが使用されました。一般的にfdlcユーティリティを使用して、コンパイルされていないフィールド・キーが使用された場合、TIP_ERRORはTIPEBADFLDと設定されます
TIPEIMPL	実装されていない機能の要求を行いました
TIPEAUTH	現在の権限では許容されていないサービスです
TIPEOS	運用システムまたはシステムエラーでメモリーの割り当てに失敗した場合、Tmaxシステムに接続が失敗した場合、またはネットワーク状態が不安定な場合などに現れるエラーです
TIPENOENT	存在しない項目にアクセスしました
TIPESVCFAIL	TIPサービスルーチンのエラーが発生し、TPFAILでtpreturn()を呼び出しました

5.5.3. TIPの使用例

本節では、TIPを使用した例を説明します。

環境ファイル

以下は、単一ノードを使用した例です。

<cfg.m>

```
*DOMAIN
res          SHMKEY=78850,      MAXUSER=200, MINCLH=1, MAXCLH=5,
              TPORTNO=8850,     BLOCKTIME=60, TXTIME=50, RACPORT=3355

*NODE
tmaxh4       TMAXDIR="/data1/starbj81/tmax",
              APPDIR="/data1/starbj81/tmax/appbin",
              PATHDIR="/data1/starbj81/tmax/path",
              TLOGDIR="/data1/starbj81/tmax/log/tlog",
              ULOGDIR="/data1/starbj81/tmax/log/ulog",
              SLOGDIR="/data1/starbj81/tmax/log/slog"

*SVRGROUP
tsvg         NODENAME = "tmaxh4", SVGTYPE=TIP
svg1         NODENAME = "tmaxh4"

*SERVER
TIP          SVGNAME=tsvg, SVRTYPE=SYS_SVR, MIN=1, MAX=1
svr          SVGNAME=svg1, MIN=1

*SERVICE
TOUPPER      SVRNAME=svr
```

以下は、マルチノードを使用した例です。

<cfg.m>

```
*DOMAIN
tmax         SHMKEY=98850,
              TPORTNO=8850,
              BLOCKTIME=60,
              RACPORT=3355,
              MAXUSER=10

*NODE
```

```

Tmaxh4      TMAXDIR="/data1/starbj81/tmax",
            APPDIR="/data1/starbj81/tmax/appbin",
            PATHDIR = "/data1/starbj81/tmax/path",
            TLOGDIR = "/data1/starbj81/tmax/log/tlog",
            ULOGDIR="/data1/starbj81/tmax/log/ulog",
            SLOGDIR="/data1/starbj81/tmax/log/slog"

tmaxh2      TMAXDIR="/data1/starbj81/tmax",
            APPDIR="/data1/starbj81/tmax/appbin",
            PATHDIR = "/data1/starbj81/tmax/path",
            TLOGDIR = "/data1/starbj81/tmax/log/tlog",
            ULOGDIR="/data1/starbj81/tmax/log/ulog",
            SLOGDIR="/data1/starbj81/tmax/log/slog"

*SVRGROUP
tsvg        NODENAME = "tmaxh4", SVGTYPE=TIP

svg1        NODENAME = "tmaxh4", COUSIN = "svg2"
svg2        NODENAME = "tmaxh2"

*SERVER
TIP         SVGNAME=tsvg, SVRTYPE=SYS_SVR, MIN=1, MAX=1
svr         SVGNAME=svg1, MIN=1, MAX=5

*SERVICE
TOUPPER     SVRNAME=svr, ROUTING = "rout1"

*ROUTING
rout1       FIELD="STRING", RANGES = "'bbbbbbb'-'ccccccc' : svg1, * :
            svg2

```

フィールド・キー・テーブル

以下は、フィールド・キー・テーブルの例です。

< tip.f >

```

#
#      common field
#
# name          number  type    flags  comments
*base 16000001
TIP_OPERATION   0       string
TIP_SEGMENT     1       string
TIP_SECTION     2       string

```

```

TIP_NODE      3      string
TIP_OCCURS    4      int
TIP_FLAGS     5      int
TIP_CURSOR    6      string
TIP_SESTM     7      int
TIP_ERROR     8      int
TIP_STATE     9      int
TIP_MORE     10     int
TIP_BADFIELD  11     string
TIP_CMD       12     string
TIP_CLID      13     int
TIP_MSG       14     string

#
#          DOMAIN section fields
#
# name                number  type    flags  comments
*base 16000100
TIP_NAME            0      string
TIP_SHMKEY          1      int
TIP_MINCLH          2      int
TIP_MAXCLH          3      int
TIP_MAXUSER         4      int
TIP_TPORTNO         5      int
TIP_RACPORT         6      int
TIP_MAXSACALL       7      int
TIP_MAXCACALL       8      int
TIP_MAXCONV_NODE    9      int
TIP_MAXCONV_SERVER  10     int
TIP_CMTRET          11     int
TIP_BLOCKTIME       12     int
TIP_TXTIME          13     int
TIP_IDLETIME        14     int
TIP_CLICHKINT       15     int
TIP_NLIVEINQ        16     int
TIP_SECURITY        17     string
TIP_OWNER           18     string
TIP_CPC             19     int
#TIP_LOGINSVC       20     string
#TIP_LOGOUTSVC      21     string
TIP_NCLHCHKTIME     22     int
TIP_DOMAINID        23     int
TIP_IPCPERM         24     int
TIP_MAXNODE         25     int
TIP_MAXSVG          26     int
TIP_MAXSVR          27     int
TIP_MAXSVC          28     int

```

TIP_MAXSPR	29	int	
TIP_MAXTMS	30	int	
TIP_MAXCPC	31	int	
TIP_MAXROUT	32	int	
TIP_MAXROUTSVG	33	int	
TIP_MAXRQ	34	int	
TIP_MAXGW	35	int	
TIP_MAXCOUSIN	36	int	
TIP_MAXCOUSINSVG	37	int	
TIP_MAXBACKUP	38	int	
TIP_MAXBACKUPSVG	39	int	
TIP_MAXTOTALSVG	40	int	
TIP_MAXPROD	41	int	
TIP_MAXFUNC	42	int	
TIP_TXPENDINGTIME	43	int	
TIP_NO	44	int	
TIP_TIPSV	45	string	
TIP_NODECOUNT	46	int	
TIP_SVGCOUNT	47	int	
TIP_SVRCOUNT	48	int	
TIP_SVCCOUNT	49	int	
TIP_COUSIN_COUNT		50	int
TIP_BACKUP_COUNT		51	int
TIP_ROUT_COUNT	52	int	
TIP_STYPE	53	string	
TIP_VERSION	54	string	
TIP_EXPDATE	55	string	
TIP_DOMAINCOUNT	56	int	
TIP_RSVG_GCOUNT	57	int	
TIP_RSVG_COUNT	58	int	
TIP_CSVG_GCOUNT	59	int	
TIP_CSVG_COUNT	60	int	
TIP_BSVG_GCOUNT	61	int	
TIP_BSVG_COUNT	62	int	
TIP_PROD_COUNT	63	int	
TIP_FUNC_COUNT	64	int	
TIP_SHMSIZE	65	int	
TIP_CRYPT	66	string	
TIP_DOMAIN_TMMLOGLVL	67	string	
TIP_DOMAIN_CLHLOGLVL	68	string	
TIP_DOMAIN_TMSLOGLVL	69	string	
TIP_DOMAIN_LOGLVL	70	string	
TIP_DOMAIN_MAXTHREAD	71	int	
#			
#	NODE section fields		
#			

# name	number	type	flags	comments
*base	16000200			
#TIP_NAME	0	string		
TIP_DOMAINNAME	1	string		
#TIP_SHMKEY	2	int		
#TIP_MINCLH	3	int		
#TIP_MAXCLH	4	int		
TIP_CLHQTIMEOUT	5	int		
#TIP_IDLETIME	6	int		
#TIP_CLICKINT	7	int		
#TIP_TPORTNO	8	int		
#TIP_TPORTNO2	9	int		
#TIP_TPORTNO3	10	int		
#TIP_TPORTNO4	11	int		
#TIP_TPORTNO5	12	int		
#TIP_RACPORT	13	int		
#TIP_TMAXPORT	14	string		
TIP_CMPPRPORT	15	string		
TIP_CMPPRSIZE	16	int		
#TIP_MAXUSER	17	int		
TIP_TMAXDIR	18	string		
TIP_TMAXHOME	19	string		
TIP_APPDIR	20	string		
TIP_PATHDIR	21	string		
TIP_TLOGDIR	22	string		
TIP_SLOGDIR	23	string		
TIP_ULOGDIR	24	string		
TIP_ENVFILE	25	string		
#TIP_LOGINSVC	26	string		
#TIP_LOGOUTSVC	27	string		
TIP_IP	28	string		
#TIP_PEER	29	string		
TIP_TMMOPT	30	string		
TIP_CLHOPT	31	string		
#TIP_IPCPERM	32	int		
#TIP_MAXSVG	33	int		
#TIP_MAXSVR	34	int		
#TIP_MAXSPR	35	int		
#TIP_MAXTMS	36	int		
#TIP_MAXCPC	37	int		
TIP_MAXGWSVR	38	int		
TIP_MAXRQSVR	39	int		
TIP_MAXGWCPC	40	int		
TIP_MAXRQCPC	41	int		
TIP_CPORTNO	42	int		
TIP_REALSVR	43	string		
TIP_RSCPC	44	int		

```

TIP_AUTOBACKUP 45      int
TIP_HOSTNAME   46      string
TIP_NODETYPE   47      int
TIP_CPU        48      int
#TIP_MAXRSTART 49      int
#TIP_GPERIOD   50      int
#TIP_RESTART   51      int
TIP_CURCLH     49      int
TIP_LIVETIME   50      string
TIP_NODE_TMMLOGLVL 51      string
TIP_NODE_CLHLOGLVL 52      string
TIP_NODE_TMSLOGLVL 53      string
TIP_NODE_LOGLVL 54      string
TIP_NODE_MAXTHREAD 55      int
TIP_EXTPORT    56      int
TIP_EXTCLHPORT 57      int
TIP_MSGSIZEWARN 58      int
TIP_MSGSIZEMAX 59      int

#
#          SVRGROUP section fields
#
# name          number  type    flags  comments
*base 16000300
#TIP_NAME       0       string
#TIP_NODENAME   1       string
TIP_SVGTYPE     2       string
#TIP_PRODNAME   3       string
TIP_COUSIN      4       string
TIP_BACKUP      5       string
TIP_LOAD        6       int
#TIP_APPDIR     7       string
#TIP_ULOGDIR    8       string
TIP_DBNAME      9       string
TIP_OPENINFO    10      string
TIP_CLOSEINFO   11      string
TIP_MINTMS      12      int
#TIP_MAXTMS     13      int
TIP_TMSNAME     14      string
#TIP_SECURITY   15      string
#TIP_OWNER      16      string
#TIP_ENVFILE    17      string
#TIP_CPC        18      int
TIP_XAOPTION    19      string
TIP_SVG_TMSTYPE 20      string
TIP_SVG_TMSOPT  21      string

```

```

TIP_SVG_TMSTHEADS      22      int
TIP_SVG_TMSLOGLVL      23      string
TIP_SVG_LOGLVL         24      string
TIP_NODENAME           25      string

#
#      SERVER section fields
#
# name          number  type    flags  comments
*base 16000350
#TIP_NAME       0       string
TIP_SVGNAME     1       string
#TIP_NODENAME   2       string
TIP_CLOPT       3       string
TIP_SEQ         4       int
TIP_MIN         5       int
TIP_MAX         6       int
#TIP_ULOGDIR    7       string
TIP_CONV        8       int
TIP_MAXQCOUNT  9       int
TIP_ASQCOUNT  10      int
TIP_MAXRSTART   11      int
TIP_GPERIOD     12      int
TIP_RESTART     13      int
TIP_SVRTYPE     14      string
#TIP_CPC        15       int
TIP_SCHEDULE    16      int
#TIP_MINTHR     17      int
#TIP_MAXTHR     18      int
TIP_TARGET      19      string
TIP_DEPEND      20      string
TIP_CASCADE     21      int
TIP_PROCNAME    22      string
TIP_LIFESPAN    23      string
TIP_DDRI        24      string
TIP_CURSVR      25      int
TIP_SVGNO       26      int
TIP_SVR_LOGLVL  27      string

#
#      SERVICE section fields
#
# name          number  type    flags  comments
*base 16000400
#TIP_NAME       0       string
TIP_SVRNAME     1       string

```

```

TIP_PRI          2      int
TIP_SVCTIME      3      int
TIP_ROUTING      4      string
TIP_EXPORT       5      int
TIP_AUTOTRAN     6      int

#
#          ROUTING section fields
#
# name          number  type    flags  comments
*base 16000425
#TIP_NAME       0      string
TIP_FLDTYPE     1      string
TIP_RANGES      2      string
TIP_SUBTYPE     3      string
TIP_ELEMENT     4      string
TIP_BUFTYPE     5      string
TIP_OFFSET      6      int
TIP_FLDLEN      7      int
#TIP_FLDOFFSET  8      int

#
#          RQ section fields
#
# name          number  type    flags  comments
*base 16000450
#TIP_NAME       0      string
#TIP_SVGNAME    1      string
TIP_PRESVC      2      string
TIP_QSIZE       3      int
TIP_FILEPATH    4      string
TIP_BOOT        5      string
TIP_FSYNC       6      int
TIP_BUFFERING   7      int
#TIP_ENQSVC     8      int
#TIP_FAILINTERVAL 9      int
#TIP_FAILRETRY  10     int
#TIP_FAILSVC    11     string
#TIP_AFTERSVC   12     string

#
#          GATEWAY section fields
#
# name          number  type    flags  comments
*base 16000500

```



```

#TIP_NAME          0      string
TIP_GWTYPE         1      string
TIP_PORTNO        2      int
#TIP_CPC           3      int
TIP_RGWADDR       4      string
TIP_RGWPORTNO     5      int
#TIP_BACKUP        6      string
#TIP_NODENAME      7      string
TIP_KEY           8      string
TIP_BACKUP_RGWADDR 9      string
TIP_BACKUP_RGWPORTNO 10    int
TIP_TIMEOUT       11     int
TIP_DIRECTION     12     string
TIP_MAXINRGW      13     int
TIP_GWOWNER       15     string
TIP_RGWOWNER      16     string
TIP_RGWPASSWD     17     string
TIP_PTIMEOUT      18     int
TIP_PTIMEINT      19     int

#
#      FUNCTION section fields
#
# name          number  type    flags  comments
*base 16000550
#TIP_NAME      0      string
#TIP_SVRNAME   1      string
TIP_FQSTART    2      int
TIP_FQEND      3      int
TIP_ENTRY      4      string

#
#      STATISTICS segment fields
#
# name          number  type    flags  comments
*base 16000600
#TIP_NAME      0      string
TIP_STATUS     1      string
TIP_STIME      2      string
TIP_TTIME      3      int
TIP_SVC_STIME  4      int
TIP_COUNT      5      int
#TIP_NO        6      int
TIP_NUM_FREE   7      int
TIP_NUM_REPLY  8      int
TIP_NUM_FAIL   9      int

```

```

TIP_NUM_REQ      10      int
TIP_ENQ_REQS     11      int
TIP_DEQ_REQS     12      int
TIP_ENQ_REPLYS   13      int
TIP_DEQ_REPLYS   14      int
TIP_CLHNO        15      int
TIP_SVR_NAME     16      string
TIP_SVC_NAME     17      string
TIP_AVERAGE     18      float
TIP_QCOUNT      19      int
TIP_CQCOUNT     20      int
TIP_QAVERAGE    21      float
TIP_MINTIME      22      float
TIP_MAXTIME      23      float
TIP_FAIL_COUNT   24      int
TIP_ERROR_COUNT  25      int
TIP_PID          26      int
TIP_TOTAL_COUNT  27      int
TIP_TOTAL_SVCFAIL_COUNT 28      int
TIP_TOTAL_ERROR_COUNT 29      int
TIP_TOTAL_AVG    30      float
TIP_TOTAL_RUNNING_COUNT 31      int
TIP_TMS_NAME     32      string
TIP_SVG_NAME     33      string
TIP_SPRI         34      int
TIP_TI_THRI      35      int
TIP_TI_AVG       36      float
TIP_TI_XID       37      string
TIP_TI_XA_STATUS 38      string
TIP_GW_NAME      39      string
TIP_GW_NO        40      int
TIP_GW_HOSTN     41      string
TIP_GW_CTYPE     42      string
TIP_GW_CTYPE2    43      string
TIP_GW_IPADDR    44      string
TIP_GW_PORT      45      int
TIP_GW_STATUS    46      string

#
#      ADMIN segment fields
#
# name          number  type    flags  comments
*base 16000650
TIP_IPADDR      0      string
TIP_USERNAME    1      string
TIP_MODULE      2      int
TIP_LOGLVL      3      string

```

```

TIP_SPEC          4          string

#
#          boot time
#
# name            number  type    flags  comments
TIP_BOOTTIME_SEC          5      int
TIP_BOOTTIME_MSEC        6      int

#
#          EXTRA flag fields
#
# name            number  type    flags  comments
*base 16000700
TIP_EXTRA_OPTION          0      int
TIP_SVRI          1      int
TIP_QPCOUNT        2      int
TIP_EMCOUNT        3      int
TIP_SVR_STATUS      4      string

```

5.5.4. システム環境情報照会プログラム

以下は、システムの環境情報を確認するクライアント・プログラムの例です。

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <usrinc/atmi.h>
#include <usrinc/fbuf.h>
#include <usrinc/tip.h>

#define SEC_DOMAIN          1
#define SEC_NODE           2
#define SEC_SVGROUP        3
#define SEC_SERVER         4
#define SEC_SERVICE        5
#define SEC_ROUTING        6
#define SEC_RQ              7
#define SEC_GATEWAY        8

main(int argc, char *argv[])
{
    FBUF      *sndbuf, *rcvbuf;
    TPSTART_T *tpinfo;
    int      i, n, sect;

```

```

long   rcvlen;
char   nodename[NAMELEN];
int     pid, count = 0;

if (argc != 3) {
    printf("Usage: %s section nodename\n", argv[0]);
    printf("section:\n");
    printf("\t1: domain\n");
    printf("\t2: node\n");
    printf("\t3: svrgroup\n");
    printf("\t4: server\n");
    printf("\t5: service\n");
    printf("\t6: routing\n");
    printf("\t7: rq\n");
    printf("\t8: gateway\n");
    exit(1);
}

if (!isdigit(argv[3][0])) {
    printf("fork count must be a digit\n");
    exit(1);
}
count = atoi(argv[3]);

sect = atoi(argv[1]);
if (sect < SEC_DOMAIN || sect > SEC_GATEWAY) {
    printf("out of section [%d - %d]\n", SEC_DOMAIN, SEC_GATEWAY);
    exit(1);
}

strncpy(nodename, argv[2], sizeof(nodename) - 1);

n = tmaxreadenv("tmax.env", "TMAX");
if (n < 0) {
    fprintf(stderr, "can't read env\n");
    exit(1);
}

tpinfo = (TPSTART_T *)tpalloc("TPSTART", NULL, 0);
if (tpinfo == NULL) {
    printf("tpalloc fail tperrno = %d\n", tperrno);
    exit(1);
}
strcpy(tpinfo->username, ".tpadmin");
if (tpstart((TPSTART_T *)tpinfo) == -1){
    printf("tpstart fail [%s]\n", tpstrerror(tperrno));
    exit(1);
}

```

```

}

if ((sndbuf = (FBUF *)tpalloc("FIELD", NULL, 0)) == NULL) {
    printf("tpalloc failed! errno = %d\n", tperrno);
    tpend();
    exit(1);
}

if ((rcvbuf = (FBUF *)tpalloc("FIELD", NULL, 0)) == NULL) {
    printf("tpalloc failed! errno = %d\n", tperrno);
    tpend();
    exit(1);
}

n = fbput(sndbuf, TIP_OPERATION, "GET", 0);
n = fbput(sndbuf, TIP_SEGMENT, "CONFIGURATION", 0);
switch (sect) {
    case SEC_DOMAIN:
        n = fbput(sndbuf, TIP_SECTION, "DOMAIN", 0);
        break;
    case SEC_NODE:
        n = fbput(sndbuf, TIP_SECTION, "NODE", 0);
        break;
    case SEC_SVGROUP:
        n = fbput(sndbuf, TIP_SECTION, "SVGROUP", 0);
        break;
    case SEC_SERVER:
        n = fbput(sndbuf, TIP_SECTION, "SERVER", 0);
        break;
    case SEC_SERVICE:
        n = fbput(sndbuf, TIP_SECTION, "SERVICE", 0);
        break;
    case SEC_ROUTING:
        n = fbput(sndbuf, TIP_SECTION, "ROUTING", 0);
        break;
    case SEC_RQ:
        n = fbput(sndbuf, TIP_SECTION, "RQ", 0);
        break;
    case SEC_GATEWAY:
        n = fbput(sndbuf, TIP_SECTION, "GATEWAY", 0);
        break;
}

n = fbput(sndbuf, TIP_NODENAME, nodename, 0);

n = tpcall("TIP SVC", (char *)sndbuf, 0, (char **)&rcvbuf, &rcvlen,
          TPNOFLAGS);

```

```

        if (n < 0) {
            printf("tpcall fail [%s]\n", tpstrerror(tperrno));
            fbprint(rcvbuf);
            tpfree((char *)sndbuf);
            tpfree((char *)rcvbuf);
            tpend();
            exit(1);
        }

#ifdef 1
        fbprint(rcvbuf);
#endif

        tpfree((char *)sndbuf);
        tpfree((char *)rcvbuf);
        tpend();
    }
}

```

[結果]

以下は、該当プログラムの結果(Domain Conf)です。

```

$ client 1 tmaxh4
fkey = 217326601, fname = TIP_ERROR, type = int, value = 0
...
fkey = 485762214, fname = TIP_CRYPT, type = string, value = NO
    fkey = 485762215, fname = TIP_DOMAIN_TMMLOGLVL, type = string, value = DEBUG1

    fkey = 485762216, fname = TIP_DOMAIN_CLHLOGLVL, type = string, value = DEBUG2

    fkey = 485762217, fname = TIP_DOMAIN_TMSLOGLVL, type = string, value = DEBUG3

    fkey = 485762218, fname = TIP_DOMAIN_LOGLVL, type = string, value = DEBUG4
    fkey = 217326763, fname = TIP_DOMAIN_MAXTHREAD, type = int, value = 128

```

5.5.5. システム統計情報照会プログラム

以下は、システムの統計情報を確認するプログラムの例です。

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <usrinc/atmi.h>
#include <usrinc/fbuf.h>
#include <usrinc/tip.h>

```

```

#define SEC_NODE      1
#define SEC_TPROC     2
#define SEC_SPR       3
#define SEC_SERVICE   4
#define SEC_RQ        5
#define SEC_TMS       6
#define SEC_TMMS      7
#define SEC_CLHS      8
#define SEC_SERVER    9

#define NODE_NAME_SIZE 32

main(int argc, char *argv[])
{
    FBUF      *sndbuf, *rcvbuf;
    TPSTART_T *tpinfo;
    int      i, n, sect;
    long     rcvlen;
    char     nodename[NODE_NAME_SIZE];
    int      stat;

    if (argc != 3) {
        printf("Usage: %s section node\n", argv[0]);
        printf("section:\n");
        printf("\t1: node\n");
        printf("\t2: tproc\n");
        printf("\t3: spr\n");
        printf("\t4: service\n");
        printf("\t5: rq\n");
        printf("\t6: tms\n");
        printf("\t7: tmms\n");
        printf("\t8: clhs\n");
        printf("\t9: server\n");
        exit(1);
    }

    sect = atoi(argv[1]);
    if (sect < SEC_NODE || sect > SEC_CLHS) {
        printf("out of section [%d - %d]\n", SEC_NODE, SEC_TMMS);
        exit(1);
    }

    memset(nodename, 0x00, NODE_NAME_SIZE);
    strncpy(nodename, argv[2], NODE_NAME_SIZE - 1);

    n = tmaxreadenv("tmax.env", "TMAX");

```

```

if (n < 0) {
    fprintf(stderr, "can't read env\n");
    exit(1);
}

tpinfo = (TPSTART_T *)tpalloc("TPSTART", NULL, 0);
if (tpinfo == NULL) {
    printf("tpalloc fail tperrno = %d\n", tperrno);
    exit(1);
}
strcpy(tpinfo->dcompwd, "xamt123");

if (tpstart((TPSTART_T *)tpinfo) == -1){
    printf("tpstart fail tperrno = %d\n", tperrno);
    exit(1);
}

if ((sndbuf = (FBUF *)tpalloc("FIELD", NULL, 0)) == NULL) {
    printf("tpalloc failed! errno = %d\n", tperrno);
    tpend();
    exit(1);
}

if ((rcvbuf = (FBUF *)tpalloc("FIELD", NULL, 0)) == NULL) {
    printf("tpalloc failed! errno = %d\n", tperrno);
    tpend();
    exit(1);
}

n = fbput(sndbuf, TIP_OPERATION, "GET", 0);
n = fbput(sndbuf, TIP_SEGMENT, "STATISTICS", 0);
switch (sect) {
    case SEC_NODE:
        n = fbput(sndbuf, TIP_SECTION, "NODE", 0);
        break;
    case SEC_TPROC:
        n = fbput(sndbuf, TIP_SECTION, "TPROC", 0);
        break;
    case SEC_SPR:
        n = fbput(sndbuf, TIP_SECTION, "SPR", 0);
        break;
    case SEC_SERVICE:
        n = fbput(sndbuf, TIP_SECTION, "SERVICE", 0);
        break;
    case SEC_RQ:
        n = fbput(sndbuf, TIP_SECTION, "RQ", 0);
        break;
}

```



```

        case SEC_TMS:
            stat = 1;
            n = fbput(sndbuf, TIP_SECTION, "TMS", 0);
            n = fbput(sndbuf, TIP_EXTRA_OPTION, (char *)&stat, 0);
            break;
        case SEC_TMMS:
            n = fbput(sndbuf, TIP_SECTION, "TMMS", 0);
            break;
        case SEC_CLHS:
            n = fbput(sndbuf, TIP_SECTION, "CLHS", 0);
            break;
        case SEC_SERVER:
            n = fbput(sndbuf, TIP_SECTION, "SERVER", 0);
            break;
    }
    n = fbput(sndbuf, TIP_NODENAME, nodename, 0);

    n = tpcall("TIP SVC", (char *)sndbuf, 0, (char **)&rcvbuf,
               &rcvlen, TPNOFLAGS);
    if (n < 0) {
        printf("tpcall fail [%s]\n", tpstrerror(tperrno));
        fbprint(rcvbuf);
        tpfree((char *)sndbuf);
        tpfree((char *)rcvbuf);
        tpend();
        exit(1);
    }

    fbprint(rcvbuf);

    tpfree((char *)sndbuf);
    tpfree((char *)rcvbuf);
    tpend();
}

```

[結果]

以下は、該当プログラムの結果(TMS STATISTICS)を表した例です。

```

$ client 3000 1 2
fkey = 217326601, fname = TIP_ERROR, type = int, value = 0
fkey = 485762680, fname = TIP_TMS_NAME, type = string, value = tms_ora2
fkey = 485762681, fname = TIP_SVG_NAME, type = string, value = xal
fkey = 217327226, fname = TIP_SPRI, type = int, value = 0
fkey = 485762649, fname = TIP_STATUS, type = string, value = RUN
fkey = 217327197, fname = TIP_COUNT, type = int, value = 0
fkey = 351544938, fname = TIP_AVERAGE, type = float, value = 0.000000

```

```
fkey = 217327212, fname = TIP_CQCOUNT, type = int, value = 0
fkey = 217327227, fname = TIP_TI_THRI, type = int, value = 1
fkey = 351544956, fname = TIP_TI_AVG, type = float, value = 0.000000
fkey = 485762685, fname = TIP_TI_XID, type = string, value = 00000013664
fkey = 485762686, fname = TIP_TI_XA_STATUS, type = string, value = COMMIT
```

5.5.6. サーバー・プロセスの起動および終了プログラム

例1

以下は、サーバー・プロセスを追加で起動および終了するプログラムの例です。

< cli.c >

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <usrinc/atmi.h>
#include <usrinc/fbuf.h>
#include <usrinc/tmaxapi.h>
#include <usrinc/tip.h>

#define NODE_NAME_SIZE 32

main(int argc, char *argv[])
{
    FBUF      *sndbuf, *rcvbuf;
    TPSTART_T *tpinfo;
    int      i, n, type, clid, count, flags;
    long     rcvlen;
    char     svrname[TMAX_NAME_SIZE];
    char     svgname[TMAX_NAME_SIZE];
    char     nodename[NODE_NAME_SIZE];
    int      pid, forkcnt;

    if (argc != 6) {
        printf("Usage: %s type svrname count nodename forkcnt\n", argv[0]);
        printf("type 1: BOOT, 2: DOWN, 3: DISCON\n");
        exit(1);
    }

    type = atoi(argv[1]);
    if ((type != 1) && (type != 2) && (type != 3)) {
        printf("couldn't support such a type %d\n", type);
```

```

        exit(1);
    }

    if (strlen(argv[2]) >= TMAX_NAME_SIZE) {
        printf("too large name [%s]\n", argv[1]);
        exit(1);
    }
    strcpy(svrname, argv[2]);
    count = atoi(argv[3]);
    flags = 0;

    strncpy(nodename, argv[4], NODE_NAME_SIZE - 1);
    forkcnt = atoi(argv[5]);

    n = tmaxreadenv("tmax.env", "TMAX");
    if (n < 0) {
        fprintf(stderr, "can't read env\n");
        exit(1);
    }

    tpinfo = (TPSTART_T *)tpalloc("TPSTART", NULL, 0);
    if (tpinfo == NULL) {
        printf("tpalloc fail tperrno = %d\n", tperrno);
        exit(1);
    }
    strcpy(tpinfo->usrname, ".tpadmin");

    for (i = 1; i < forkcnt; i++) {
        if ((pid = fork()) < 0)
            exit(1);
        else if (pid == 0)
            break;
    }

    if (tpstart((TPSTART_T *)tpinfo) == -1){
        printf("tpstart fail tperrno = %d\n", tperrno);
        exit(1);
    }

    if ((sndbuf = (FBUF *)tpalloc("FIELD", NULL, 0)) == NULL) {
        printf("tpalloc failed! errno = %d\n", tperrno);
        tpend();
        exit(1);
    }

    if ((rcvbuf = (FBUF *)tpalloc("FIELD", NULL, 0)) == NULL) {
        printf("tpalloc failed! errno = %d\n", tperrno);
    }

```

```

        tpend();
        exit(1);
    }

    n = fbput(sndbuf, TIP_OPERATION, "GET", 0);
    n = fbput(sndbuf, TIP_SEGMENT, "ADMINISTRATION", 0);
    if (type == 1)
        n = fbput(sndbuf, TIP_CMD, "BOOT", 0);
    else if (type == 2)
        n = fbput(sndbuf, TIP_CMD, "DOWN", 0);
    else
        n = fbput(sndbuf, TIP_CMD, "DISCON", 0);

    if (type == 3) {
        clid = count;          /* at type 3 */
        flags |= TIP_SFLAG;
        n = fbput(sndbuf, TIP_CLID, (char *)&clid, 0);
        n = fbput(sndbuf, TIP_FLAGS, (char *)&flags, 0);
    } else {
        flags |= TIP_SFLAG;
        n = fbput(sndbuf, TIP_SVRNAME, svrname, 0);
        n = fbput(sndbuf, TIP_COUNT, (char *)&count, 0);
        n = fbput(sndbuf, TIP_FLAGS, (char *)&flags, 0);
    }

    n = fbput(sndbuf, TIP_NODENAME, nodename, 0);

    n = tpcall("TIPSVC", (char *)sndbuf, 0, (char **)&rcvbuf,
               &rcvlen, TPNOFLAGS);
    if (n < 0) {
        printf("tpcall failed! errno = %d[%s]\n", tperrno, tpstrerror(tperrno));

        fbprint(rcvbuf);
        tpfree((char *)sndbuf);
        tpfree((char *)rcvbuf);
        tpend();
        exit(1);
    }

    fbprint(rcvbuf);

    tpfree((char *)sndbuf);
    tpfree((char *)rcvbuf);
    tpend();

```

例2

以下は、svr23_stat_insというサーバーのログレベルをdebug4に変更する例です。

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <usrinc/atmi.h>
#include <usrinc/fbuf.h>
#include <usrinc/tmaxapi.h>
#include <usrinc/tip.h>
#include "../fdl/tip_fdl.h"

#define NFLAG 32
#define GFLAG 8
#define VFLAG 1024

int case_chlog(int, char *[], FBUF *);

#define NODE_NAME_SIZE 32

main(int argc, char *argv[])
{
    FBUF      *sndbuf, *rcvbuf;
    TPSTART_T *tpinfo;
    int      i, ret, n, type, clid, count, flags = 0;
    long     rcvlen;
    char     svrname[TMAX_NAME_SIZE];
    char     svgname[TMAX_NAME_SIZE];
    char     nodename[NODE_NAME_SIZE];
    int      pid, forkcnt;

    if (argc < 6) {
        printf("Usage: %s svgname svrname nodename [chlogmodule] [flags] [loglvl]\n", argv[0]);
        printf("chlogmodule 1: TIP_TMM, 2: TIP_CLH, 4: TIP_TMS, 8: TIP_SVR\n");

        printf("flags 1: NFLAGS, 2: GFLAGS, 3: VFLAGS\n");
        printf("loglvl : 1: compact, 2: basic, 3: detail, 4: debug1, 5: debug2, 6: debug3, 7: debug4\n");
        exit(1);
    }

    n = tmaxreadenv("tmax.env", "TMAX");
    if (n < 0) {
        fprintf(stderr, "can't read env\n");
    }
}
```

```

        exit(1);
    }

    tpinfo = (TPSTART_T *)tpalloc("TPSTART", NULL, 0);
    if (tpinfo == NULL) {
        printf("tpalloc fail tperrno = %d\n", tperrno);
        exit(1);
    }
    strcpy(tpinfo->usrname, ".tpadmin");
    strcpy(svgname, argv[1]);
    strcpy(svrname, argv[2]);
    strncpy(nodename, argv[3], NODE_NAME_SIZE - 1);

    if (tpstart((TPSTART_T *)tpinfo) == -1){
        printf("tpstart fail tperrno = %d\n", tperrno);
        exit(1);
    }

    if ((sndbuf = (FBUF *)tpalloc("FIELD", NULL, 0)) == NULL) {
        printf("tpalloc failed! errno = %d\n", tperrno);
        tpend();
        exit(1);
    }

    if ((rcvbuf = (FBUF *)tpalloc("FIELD", NULL, 0)) == NULL) {
        printf("tpalloc failed! errno = %d\n", tperrno);
        tpend();
        exit(1);
    }

    ret = case_chlog(argc, argv, sndbuf);
    n = fbput(sndbuf, TIP_OPERATION, "GET", 0);
    n = fbput(sndbuf, TIP_SEGMENT, "ADMINISTRATION", 0);
    n = fbput(sndbuf, TIP_CMD, "CHLOG", 0);
    n = fbput(sndbuf, TIP_NODENAME, nodename, 0);
    n = fbput(sndbuf, TIP_SVGNAME, svgname, 0);
    n = fbput(sndbuf, TIP_SVRNAME, svrname, 0);

    n=tpcall("TIPSVC", (char *)sndbuf, 0, (char **)&rcvbuf, &rcvlen, TPNOFLAGS);

    if (n < 0) {
        printf("tpcall failed! errno = %d[%s]\n", tperrno,
            tpstrerror(tperrno));
        fbprint(rcvbuf);
        tpfree((char *)sndbuf);
        tpfree((char *)rcvbuf);
        tpend();
    }

```

```

        exit(1);
    }

    fbprint(rcvbuf);
    tpfree((char *)sndbuf);
    tpfree((char *)rcvbuf);
    tpend();
}

int case_chlog(int argc2, char *argv2[], FBUF *sndbuf)
{
    int  chlogmdl, loglvl, flags, n=0;
    char  cloglvl[TMAX_NAME_SIZE];
    const int true = 1, false = 0;

    chlogmdl = atoi(argv2[4]);
    if( (chlogmdl != 1) && (chlogmdl != 2) && (chlogmdl != 4) &&
        (chlogmdl != 8)
    {
        printf("couldn't support such a chlogmdl\n");
        exit(1);
    }

    flags = atoi(argv2[5]);
    if( (flags != NFLAG) && (flags != GFLAG) && (flags != VFLAG) )
    {
        printf("couldn't support such a flags\n");
        exit(1);
    }

    loglvl = atoi(argv2[6]);
    if( (loglvl < 1) || (loglvl > 7) )
    {
        printf("couldn't support such a loglvl\n");
        exit(1);
    }

    switch (loglvl)
    {
        case 1 :
            strcpy(cloglvl, "compact");
            break;
        case 2 :
            strcpy(cloglvl, "basic");
            break;
        case 3 :
            strcpy(cloglvl, "detail");

```

```

        break;
    case 4 :
        strcpy(cloglvl, "debug1");
        break;
    case 5 :
        strcpy(cloglvl, "debug2");
        break;
    case 6 :
        strcpy(cloglvl, "debug3");
        break;
    case 7 :
        strcpy(cloglvl, "debug4");
        break;
}
n = fbput(sndbuf, TIP_MODULE, (char *)&chlogmdl, 0);
n = fbput(sndbuf, TIP_FLAGS, (char *)&flags , 0);
n = fbput(sndbuf, TIP_LOGLVL, cloglvl , 0);
return 1;
}

```

[結果] (TIP_SVR, => DEBUG4)

```

$ client xal svr23_stat_ins $HOSTNAME 8 1024 7
fkey = 217326601, fname = TIP_ERROR, type = int, value = 0
>>> tadmin (cfg -v)

loglvl = DEBUG4

```

5.6. 再帰呼び出し(Local recursive call)

サーバーでtpcall()を実行する場合、同じサーバーに存在するサービスの場合にも内部で再帰呼び出しができるように機能が追加されました。これは、サーバーでマルチ・コンテキストの手法を使用して、tpcall()に限って再帰呼び出し(Local Recursive call)が可能となるようにしたものです。無限ループを防ぐために、最大階層は8に制限します。

注

再帰呼び出しを使用するためには、サーバー・プログラムをコンパイルする際に必ずCFLAGSに-D_MCONTEXTを追加し、libsvr.soの代わりにlibsvrmc.soサーバー・ライブラリーを利用してコンパイルする必要があります。

サーバー・プログラム

以下は、サーバー・プログラムの例です。


```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <usrinc/atmi.h>

SVC15004_1(TPSVCINFO *msg)
{
    int      i;
    char      *rcvbuf;
    long      rcvlen;

    if ((rcvbuf = (char *)tpalloc("STRING", NULL, 0)) == NULL)
        printf("rcvbuf tpalloc fail[%s]\n", tpstrerror(tperrno));
    if (tpcall("SVC15004_2", msg->data, 0, &rcvbuf, &rcvlen, 0) == -1)
    {
        printf("tpcall fail [%s]\n", tpstrerror(tperrno));
        tpfree((char *)rcvbuf);
        tpreteturn(TPFAIL, 0, 0, 0, 0);
    }

    strcat(rcvbuf, "_Success");

    tpreteturn(TPSUCCESS, 0, (char *)rcvbuf, 0, 0);
}

SVC15004_2(TPSVCINFO *msg)
{
    int      i;
    char      *rcvbuf;
    long      rcvlen;

    if ((rcvbuf = (char *)tpalloc("STRING", NULL, 0)) == NULL)
        printf("rcvbuf tpalloc fail \n");
}

if (tpcall("SVC15004_3", msg->data, 0, &rcvbuf, &rcvlen, 0) == -1)
{
    printf("tpcall fail [%s]\n", tpstrerror(tperrno));
    tpfree((char *)rcvbuf);
    tpreteturn(TPFAIL, 0, 0, 0, 0);
}
strcat(rcvbuf, "_Success");
tpreteturn(TPSUCCESS, 0, (char *)rcvbuf, 0, 0);
}

SVC15004_3(TPSVCINFO *msg)

```

```

{
    int      i;
    char     *rcvbuf;
    long     rcvlen;

    if ((rcvbuf = (char *)tpalloc("STRING", NULL, 0)) == NULL)
        printf("rcvbuf tpalloc fail \n");

    if (tpcall("SVC15004_4", msg->data, 0, &rcvbuf, &rcvlen, 0) == -1)
    {
        printf("tpcall fail [%s]\n", tpstrerror(tperrno));
        tpfree((char *)rcvbuf);
        tpreturn(TPFAIL, 0, 0, 0, 0);
    }

    strcat(rcvbuf, "_Success");
    tpreturn(TPSUCCESS, 0, (char *)rcvbuf, 0, 0);
}

```

メイクファイル

以下は、メイクファイルの例です。

<Makefile.c.mc>

```

# Server makefile

TARGET   = $(COMP_TARGET)
APOBJS   = $(TARGET).o
NSDLOBJ  = $(TMAXDIR)/lib64/sdl.o

LIBS     = -lsvrmc -lnodb
OBJJS    = $(APOBJS) $(SVCTOBJ)

SVCTOBJ  = $(TARGET)_svctab.o

CFLAGS   = -O -Ae -w +DSblended +DD64 -D_HP -I$(TMAXDIR) -D_MCONTEXT

APPDIR   = $(TMAXDIR)/appbin
SVCTDIR  = $(TMAXDIR)/svct
LIBDIR   = $(TMAXDIR)/lib64

#
.SUFFIXES : .c

.c.o:

```

```

$(CC) $(CFLAGS) -c $<

#
# server compile
#

$(TARGET): $(OBJS)
    $(CC) $(CFLAGS) -L$(LIBDIR) -o $(TARGET) $(OBJS) $(LIBS) $(NSDLOBJ)
    mv $(TARGET) $(APPDIR)/.
    rm -f $(OBJS)

$(APOBJS): $(TARGET).c
    $(CC) $(CFLAGS) -c $(TARGET).c

$(SVCTOBJ):
    cp -f $(SVCTDIR)/$(TARGET)_svctab.c .
    touch ./$(TARGET)_svctab.c
    $(CC) $(CFLAGS) -c ./$(TARGET)_svctab.c

#
clean:
    -rm -f *.o core $(APPDIR)/$(TARGET)

```


第6章 ガイドの構成

本章は、Tmaxの全ガイドの一覧と各ガイドの内容について説明します。

6.1. 概要

Tmaxを初めて使用するか、または不慣れなユーザーは、様々なTmaxの製品ガイドから必要なガイドを特定できないことがあります。

たとえば、Tmaxを使用する環境でDelphiのような4GL言語を使用してプログラムを開発する際、プログラムのフローや使用するAPIについての知識が必要ですが、参照できるガイドを判断できない場合などです。本章では、このような問題を解消するため、Tmaxガイドの構成および各ガイドの内容について説明しています。また、ガイド同士がどのように関連付けられているかについて説明します。

同書は、ガイドの構成全体を理解できるように作成されているので、Tmaxガイドを初めて接するユーザーは注意深くお読みになることをお勧めします。

6.2. ガイドの構成および内容

Tmaxのガイド一覧は以下のとおりです。

NO	ガイド名
1	Tmax スタートガイド
2	Tmax インストールガイド
3	Tmax 運用ガイド
4	Tmax アプリケーション開発ガイド
5	Tmax メッセージリファレンスガイド
6	Tmax リファレンスガイド
7	Tmax JTmaxServerユーザガイド
8	Tmax FDLリファレンスガイド
9	Tmax Host-linkガイド(SNA LU0, SNA LU6.2)
10	Tmax WebTAsyncユーザガイド
11	Tmax WebTJCAユーザガイド
12	Tmax JTCユーザガイド
13	Tmax HMSユーザガイド

NO	ガイド名
14	Tmax Webユーザガイド
15	Tmax COBOLユーザガイド
16	Tmax ゲートウェイガイド(シリアル)
17	Tmax ゲートウェイガイド(TCP-IP)
18	Tmax ゲートウェイガイド(TCP-IPスレッド)
19	Tmax ゲートウェイガイド(TCP-IPサービス)
20	Tmax ゲートウェイガイド(Webサービス)
21	Tmax ゲートウェイガイド(X.25)
22	Tmax プログラミングガイド(4GL)
23	Tmax プログラミングガイド(ダイナミックライブラリ)
24	Tmax プログラミングガイド(RCA)
25	Tmax プログラミングガイド(RQ)
26	Tmax プログラミングガイド(RPC)
27	Tmax プログラミングガイド(SQ)
28	Tmax プログラミングガイド(UCS)
29	Tmax プログラミングガイド(MultipleRM)
30	Tmax XAライブラリ及びXAゲートウェイガイド
31	Tmax WebAdminユーザガイド
32	TmaxGrid ユーザガイド
33	Tmax TCacheガイド
34	Tmax デプロイメントユーザガイド
35	Tmax リリースノート

Tmaxは、計35のガイドを提供しています。以下の各ガイドについての紹介を参考にして必要なガイドをご利用ください。

- Tmax スタートガイド

Tmaxの基本概念と構成について説明しています。

- Tmax インストールガイド

Tmaxのインストール手順について説明しています。

- Tmax 運用ガイド

Tmaxを使用するための環境設定ファイルとシステムの運用方法について説明しています。

- 環境変数についての説明と設定方法
- 環境ファイルの設定方法と各使用環境のセクションの設定方法
- システムの起動と終了方法
- Tmaxシステムを運用するためのアドミン・ツールの使用方法、情報の照会方法、運用の管理方法
- Tmax アプリケーション開発ガイド

Tmaxを使用してプログラムを開発するための基本的な内容について説明しています。

 - Tmaxを使用するアプリケーションの概念、構成、特徴
 - クライアント・プログラムのフロー、特徴、開発環境、コンパイル方法
 - サーバー・プログラムのフロー、特徴、開発環境、コンパイル方法
 - クライアント/サーバーの通信タイプ
 - クライアント/サーバーの通信タイプに使われるバッファの種類と管理方法
 - トランザクションの特徴と管理方法
 - マルチ・スレッドおよびマルチ・コンテキスト環境におけるプログラムのフロー、実装方法、例
 - Tmaxで提供するセキュリティー・システムについての説明と段階別の特徴
 - クライアントで使用可能なAPIと使用方法
 - サーバーで使用可能なAPIと使用方法
 - 多様な環境におけるアプリケーションの例
 - Tmaxで発生するエラーと対応方法およびデバッグ
 - Tmaxの環境設定ファイルとMakefileの例
- Tmax メッセージリファレンスガイド

Tmax製品の使用時に発生し得るメッセージ(エラー・メッセージを含む)と、その対応方法について説明しています。

 - Tmaxメッセージの構造と種類

- モジュール・メッセージ
- Tmax リファレンスガイド

Tmaxアプリケーションを開発するときに使用するコマンドの概念と使用方法および例について説明しています。また、クライアントとサーバーの接続および通信で使用する関数の使用方法と例について説明しています。

 - コマンドの概念と使用方法、例
 - 関数についての説明と使用方法、例
 - エラーとその対応方法
- Tmax JTmaxServerユーザガイド

Tmaxで提供するJTmaxServerを使用する開発者を対象としており、環境構成方法、環境設定方法、APIと例について説明しています。
- Tmax FDLリファレンスガイド

Tmax FDL関数の定義とプログラムの例を通じて、FDLのすべての機能を十分に活用できる方法について説明しています。

 - FDLの概念、フィールド・バッファについての説明とFDLテーブルの作成
 - FDL関数の使用方法、例
- Tmax Host-linkガイド(SNA LU0, SNA LU6.2)

Tmaxとホストを接続するHost-linkについて説明しています。

 - Host-linkの概念、構造、機能、起動および終了方法
 - INBOUNDセッションとOUTBOUNDセッションの管理方法
 - INBOUNDサービスとOUTBOUNDサービス
 - Host-linkの使用中に発生する問題と対応方法
 - 環境設定方法、状態モニタリングAPI、エラー・コード、ユーザー関数、Host-linkで使用するファイル一覧
- Tmax WebTAsyncユーザガイド

TmaxのAsync Javaゲートウェイと非同期でインバウンド/アウトバウンド通信を行うためのJavaライブラリーであるWebTAsyncについて説明しています

- Tmax WebTJCAユーザガイド

TmaxのWebTJCAを使用してプログラムを開発する方法について説明しています。

- WebTJCAの紹介
- WebTJCA APIについての説明
- WebTJCAのインバウンド通信
- JEUS 6とWebLogicでWebTJCAを使用する例

- Tmax JTCユーザガイド

TmaxのJTC(Jeus-Tuxedo Connector)は、JEUSおよびTuxedoに接続してサービスを利用できるライブラリーであり、WebTライブラリーに含まれてデプロイされます。JTCを使用してプログラムを開発する方法について説明しています。

- JEUSおよびTuxedoに接続するための環境設定方法
- JTCクラスと関数の使用方法、例
- TuxedoとWebTのログ設定方法

- Tmax HMSユーザガイド

HMSを利用してプログラムを使用する方法について説明しています。

- HMSの概念、構造、プログラミングのための概念
- 環境ファイルにHMSを設定する方法と例
- HMS API関数の使用方法と例、HMSビルド、起動と終了、照会などの管理方法
- HMSを使用するための環境設定、プログラムの例、コンパイル方法

- Tmax WebTユーザガイド

Web経由でTmaxサービスを利用するため、JEUSのWebT APIを使用してクライアント・プログラムを開発する方法について説明しています。

- WebTの機能、WebTConnectionPool、WebT-Server Systemの概念と構成要素

- WebT、JMaxの環境設定方法
- 用途に応じたWebT APIの使用方法和例
- Tmax COBOLユーザガイド

COBOLを使用してTmaxサーバー・プログラムを作成する方法について説明しています。

 - Tmax環境ファイルの作成方法、コンパイル、サービス・テーブルの作成、エンジンの起動方法
 - サーバー・プログラムの作成方法、Makefileの例、サーバー・プログラムの起動とテスト方法
 - サーバー・プログラムでのFDLの使用 方法、グローバル・トランザクション・プログラムやTPSVRINIT/TPSVRDONEの使用 方法
 - COBOLで使用するコマンドおよび関数の使用 方法と例
- Tmax ゲートウェイガイド(SERIAL)

Tmaxサーバーと非Tmaxサーバーがシリアル(Serial)通信を行う場合、インターフェースを担当するTmaxが提供するSERIAL(RS232)ゲートウェイについて説明しています。

 - SERIALゲートウェイの概念と動作プロセス
 - SERIALゲートウェイのサービス・タイプ
 - SERIALゲートウェイを使用するための環境設定
 - SERIALゲートウェイを使用したプログラムの例
- Tmax ゲートウェイガイド(TCP-IP)

Tmaxサーバーと非Tmaxサーバーのインターフェースを担当するTmaxが提供するTCP/IPゲートウェイについて説明しています。

 - TCP/IPゲートウェイの概念と動作プロセス
 - TCP/IPゲートウェイのサービス・タイプ
 - TCP/IPゲートウェイを使用するための環境設定
 - TCP/IPゲートウェイを使用したプログラムの例
- Tmax ゲートウェイガイド(TCP-IPサービス)

Tmaxサーバーと非TmaxサーバーのTCP/IP通信時にインターフェースを担当するTmaxが提供するTCP/IPサービス・ゲートウェイについて説明しています。

- TCP/IPサービス・ゲートウェイの概念と動作構造
 - TCP/IPサービス・ゲートウェイのサービス・タイプ
 - TCP/IPサービス・ゲートウェイを使用するための環境設定
 - TCP/IPサービス・ゲートウェイを使用するためのAPI
 - TCP/IPサービス・ゲートウェイを使用したプログラムの例
- Tmax ゲートウェイガイド(TCP-IPスレッド)

非TmaxクライアントとTmaxの間でインターフェースを担当するTCP/IPスレッド・ゲートウェイについて説明しています。

 - TCP/IPスレッド・ゲートウェイの概念とプロセス
 - TCP/IPスレッド・ゲートウェイのタイプ
 - TCP/IPスレッド・ゲートウェイを使用するための環境設定
 - TCP/IPスレッド・ゲートウェイを使用するためのユーザーAPI関数
 - TCP/IPスレッド・ゲートウェイを使用したプログラムの例
- Tmax ゲートウェイガイド(Webサービス)

Tmaxサービスを変更せずにWebサービスで使用するために提供されるゲートウェイ・サーバーのWebサービス・ゲートウェイについて説明しています。

 - Webサービスの概念、Webサービス・ゲートウェイについての紹介
 - Webサービス・ゲートウェイを使用するための環境設定
 - Webサービス・ゲートウェイの使用方法および管理方法
- Tmax ゲートウェイガイド(X.25)

Tmaxサーバーと非Tmaxサーバーのインターフェースを担当するTmaxが提供するX.25ゲートウェイについて説明しています。

 - X.25ゲートウェイの概念と動作プロセス

- X.25ゲートウェイのサービス・タイプ
- X.25ゲートウェイを使用するための環境設定方法
- X.25ゲートウェイを使用したプログラムの例
- Tmax プログラミングガイド(4GL)

4GL言語を使用してTmaxアプリケーションを開発する方法について説明しています。

 - PowerBuilderを利用したプログラム開発方法、関数の使用方法と例、実際に開発されたアプリケーションの例
 - Visual Basicを利用したプログラム開発方法、関数の使用方法と例、実際に開発されたアプリケーションの例
 - Delphiを利用したプログラム開発方法、関数の使用方法と例、実際に開発されたアプリケーションの例
 - Visual Basic .netを利用したプログラム開発方法、関数の使用方法と例、実際に開発されたアプリケーションの例
 - C# .netを利用したプログラム開発方法、関数の使用方法と例、実際に開発されたアプリケーションの例
 - ASPを利用したプログラム開発方法、関数の使用方法と例、実際に開発されたアプリケーションの例
- Tmax プログラミングガイド(ダイナミックライブラリ)

Tmax TDL(Tmax Dynamic Library)を使用してプログラムを開発する方法について説明しています。

 - TDLの概念、特徴、構成
 - TCDLを使用するための環境設定とTmax環境設定
 - TDLコマンドの使用方法、TDL APIの使用方法と例
- Tmax プログラミングガイド(RCA)

Tmaxクライアント・ライブラリーを使用できない既存の通信プログラムやPDAなどをTmaxシステムに接続できるようにサポートするRCAモジュールのインストールおよび環境設定方法について説明しています。

 - RCAの基本概念と構造、特徴
 - RCAとTmaxシステムを連携する方法

- RCAを利用するための環境変数
- 各プラットフォームに提供されるファイルとTmaxの環境ファイルに設定する方法
- 開発者がRCAHをプログラミングするときの注意事項と例およびエラー・メッセージ
- Tmax プログラミングガイド(RPC)

Tmaxプログラム・モデルのうち、RPC(Remote Procedure Call)の機能と特性および構成要素について説明しています。

 - RPCの基本概念と種類および制約事項
 - RPCの構成要素
- Tmax プログラミングガイド(RQ)

TmaxのRQ(Reliable Queue)の概念と使用方法について説明しています。

 - RQの定義、概念、特徴
 - RQを使用するシステム構成とAPI
 - RQを使用するための環境設定と状態情報管理
- Tmax プログラミングガイド(SQ)

TmaxのSQ(Session Queue)の概念と使用方法について説明しています。

 - SQの定義、概念、特徴
 - SQを使用するシステム構成とAPI
 - SQを使用するための環境設定と状態情報管理
 - SQを使用する例
- Tmax プログラミングガイド(UCS)

サーバー・プロセスのうち、Tmax UCSの概念および使用方法について説明しています。

 - プロセスについての基本的な説明とプログラム・フロー
 - UCSプログラムの構成、環境設定およびコンパイル方法、関数の使用方法とプログラムの例

- RDPプログラムの構成、環境設定およびコンパイル方法とプログラムの例
- Tmax プログラミングガイド(MultipleRM)

Tmaxサーバー・プロセスのうち、Tmax MultipleRMについての概要と機能および特性について説明しています。

 - MultipleRMの概要
 - 環境設定とサーバーの使用方法およびプログラムの例
- Tmax XAライブラリ及びXAゲートウェイガイド

XAライブラリーとXAゲートウェイの使用方法について説明しています。
- Tmax WebAdminユーザガイド

Tmaxを管理するユーザー向けのツールであるWebAdminの基本概念とインストールおよび使用方法について説明しています。

 - WebAdminの基本概念と制約事項
 - WebAdminを運用するためのインストール手順と開始および使用方法
 - 各メニューの機能と使用方法
 - WebAdminで発生するメッセージ
- TmaxGrid ユーザガイド

マルチ・ノード環境でインメモリ・データを同期化するTmaxGridを使用してプログラムを開発する方法について説明しています。

 - TmaxGridの基本概念
 - 環境設定とサーバーの使用方法およびプログラムの例
- Tmax TCacheガイド

Tmax TCacheの基本概念と使用方法について説明しています。

 - TCacheの基本概念
 - TCacheの使用用法

- TCache APIの使用方法
- TCacheの同期化方法
- Tmax デプロイメントユーザガイド
 - サーバー・アプリケーション・リソース・ファイル、環境設定ファイルなどのデプロイおよび起動を管理するデプロイメント機能について説明しています。
 - デプロイメントの紹介
 - デプロイメントの環境設定
 - デプロイメントの運用管理
 - デプロイメントを行うための簡単な起動と終了および状態の照会方法

用語集

トランザクション(Transaction)

一体不可分の処理単位です。1つのトランザクションは複数のセッションを含みます。

原子性(Atomicity)

一体不可分の処理単位です。処理がすべて完了するか、あるいは1つも実行されないかのいずれかの状態になることです。

一貫性(Consistency)

正常実行されたトランザクションの結果を共有リソースに更新します。トランザクションが失敗した場合は共有リソースを更新しません。

独立性(Isolation)

トランザクションの影響を受けて共有リソースが変更されても、トランザクションがコミットするまでは、他のトランザクションに影響を与えません。

耐久性(Durability)

トランザクションの結果がコミットされたら、その結果を維持します。

グローバル・トランザクション(Global Transactions)

1つ以上のRMを1つの処理単位とします。システムで作成されるトランザクションは自動コミットされます。

コミット(Commit)

トランザクション処理が成功したとき、その結果を確定ことです。

ロールバック(Rollback)

障害発生時にトランザクションの処理をすべて取り消し最初の時点に戻すか、または、ユーザーが任意でトランザクションの結果を処理以前の状態に戻すことです。

TPモニター(Transaction Processing Monitor)

各種プロトコルで動作するセッションとシステム、およびデータベース間の最小の処理単位であるトランザクションを監視し、一貫性を維持させるトランザクション管理ミドルウェアです。

ダウンサイジング(DOWNSIZING)

大型汎用機のメインフレーム環境からオープン型の分散システムに移行することです。

ミドルウェア(Middleware)

分散コンピューティング環境で単一のユーザー環境を提供し、異種システム間のネットワークの接続やクライアントとサーバー間の通信、およびコンピューター同士を接続するシステム・ソフトウェアです。

Webアプリケーション・サーバー(WAS、Web Application Server)

Webでトランザクションを処理し、異種間の相互通信機能(J2EE)を提供します。

メッセージ指向ミドルウェア(MOM、Messaging Oriented Middleware)

メッセージをキューと呼ばれる中間結果格納用の空間に入れて処理し、キューによるメッセージ管理機能を提供(非同期)します。

データベース・アクセス・システム(Database Access System)

分散環境で複数のデータベース・サーバーを一貫した方法で利用できる環境を提供します。

RPCシステム

ネットワーク上の異なるマシンにあるプログラムを実行(同期)します。

オブジェクト・リクエスト・ブローカ(ORB、Object Request Broker)

クライアント・オブジェクトがORBというソフトウェア・バスを利用してリモート・サーバーのメソッドを呼び出す機能を提供します。

Tmaxマネージャー(TMM、Tmax Manager)

Tmaxシステムを運用および管理する中核プロセスです。Tmaxシステムのすべての共有情報とCLL、CLH、TMS、AP(Application Program)サーバー・プロセスを管理します。

クライアント・リスナー(CLL、Client Listener)

クライアントとTmaxを接続するプロセスです。クライアントの接続状態を管理するためのポート・リスナー(PORT Listener)を設定し、クライアントからの要求を受信します。

クライアント・ハンドラー(CLH、Client Handler)

クライアントとサーバー間を仲介します。サービスを提供するトランザクション処理サーバーにサービスを要求し、サーバーの接続を管理します。

トランザクション管理サーバー(TMS、Transaction Management Server)

データベース管理および分散トランザクション処理を行うプロセスであり、データベース関連のシステムで動作します。XAサービスで発生するコミットおよびロールバックをRMIに渡します。

トランザクション・ログ・マネージャー(TLM、Transaction Log Manager)

トランザクションが発生すると、実際にCLHがコミットを実行する前にTLMがトランザクション・ログを保存します。

RQS(Reliable Queue Server)

Tmaxシステムのディスク・キュー(Disk Queue)を管理するプロセスであり、ファイルに読み込みと書き込みを実行します。

ゲートウェイ・プロセス(GW、Gateway Process)

複数のドメインで構成されている場合、ドメイン間の通信を行います。

Tmadmin(Tmax Administrator)

Tmax関連情報の監視および環境ファイルの変更などを管理します。

TCS(Tmax Control Server)

CLHの要求を受け、ビジネス・ロジックを処理して結果を返します。

UCS(User Control Server)

CLHの要求を受け、ビジネス・ロジックを処理してその結果を返しつつ、当該プロセスがコントロールを維持します。

TIP(Tmax Information Provider)

システム環境情報と統計情報を確認し、システムを運用および管理します。

TIM(Tmax Information MAP)

Tmaxシステムを運用する主要情報であり、Tmaxで管理する共有メモリーのことをいいます。TIMは、エンジン・プロセスのTMMプロセスによって生成されます。

ドメイン・ソケット(Domain Socket)

UNIXドメイン・ソケット通信方式はファイルを利用して内部プロセス間の通信をする方式です。ソケットAPIを変更せずに使用します。

SLM(System Load Management)

定義された割合で負荷を分散処理する方式です。

DDR(Data Dependent Routing)

データ値によって分散処理する方式です。複数のノードで共通したサービスを提供すると、データの範囲に応じてノード間のルーティングができるように指定します。

DLM(Dynamic Load Management)

負荷の割合に応じて動的に処理グループを選択する方式です。特定のノードにロードが集中する場合、Tmaxの動的負荷調整方法に従って負荷を分散し、システム全体の処理量を増やして処理時間を短縮します。

TCS(Tmax Control Server)

クライアントの要求によって手動で実行されるプロセスであり、ほとんどはこのプロセスが使われます。

UCS(User Control Server)

クライアントの要求がなくても能動的にデータを送信できるプロセスであり、Tmax固有の機能です。

POD(Processing On Demand)

クライアントの要求があった場合にのみ、サーバー・プロセスが起動して処理する手法です。

RCA(Raw Client Agent)

マルチ・スレッド方式でプロセスを効率的に処理できるマルチ・ポートをサポートします。

SCA(Simple Client Agent)

非TmaxクライアントまたはTmaxクライアントのいずれも対応できるマルチ・ポートをサポートします。

2PC(2相コミット)プロトコル

複数の同種または異種のデータベースにまたがって処理を行う際、すべてのデータベースで整合性が保証できるようにする手法です。コミット準備とコミット実行の2段階に分けられます。

RQ(Reliable Queue)

サービスの実行中に障害が発生して要求内容が無くなるのを防ぎ、サービスが信頼性を持って処理できるようにします。

HMS(Hybrid Messaging System)

Tmaxの機能です。送信者(Sender)と受信者(Receiver)の疎結合(loosely coupled)のための通信媒介体であり、キュー方式とトピック方式を提供しています。

WebT(Web Transaction)

クライアントとサーバー環境のミドルウェア製品であるTmaxとJavaアプリケーション・プログラム間のトランザクション・サービスをサポートします。

STRINGバッファ

NULL値で終わる文字列であり、バッファの長さを指定する必要がありません。

CARRAYとX_OCTETバッファ

長さが指定されているバイト列で構成されているバッファです。一般的にバイナリ・タイプのデータを送信するときに使われます。データの送受信時には必ず長さを明示する必要があります。

STRUCTとX_C_TYPEバッファ

C言語の構造体をデータ通信に用いるときに使用します。

X_COMMONバッファ

メンバー・タイプがchar型、int型、long型に限定されたC構造体です。

FIELDバッファ

フィールド・キーとデータ値をペアで管理するデータ・バッファです。すべてのプロトタイプのデータを格納することができます。

索引

シンボル

2PC, 17

2PC, 16

A

ACID, 5, 15

AP, 16

Atomicity, 15

C

CARRAYバッファァー, 48

CLH, 6

CLL, 6

Consistency, 15

CRM, 16

D

Database Access System, 3

DDR, 10

DDRによる負荷調整, 19

DLM, 10

DLMによる負荷調整, 20

Domain Socket, 9

DOMAINセクション, 58

Durability, 15

F

FDL, 49

FDL API, 64

FIELDバッファァー, 48

G

GATEWAYセクション, 58

GW, 7

H

HMS, 35

HMS-キュー方式, 36

HMS-トピック方式, 36

I

Isolation, 15

J

JAVA GW, 44

JTmax, 44

L

Local recursive call, 184

M

Messaging Oriented Middleware, 3

N

NODE セクション, 58

O

Object Request Broker, 3

OLTP, 5

OSI, 5

OSI-TP, 16

P

POD, 11, 25

R

RACD, 7

RCA, 12, 29

RDP, 12, 33

RM, 16

ROUTING セクション, 58

RPC System, 3

RQ, 11, 26

RQS, 7

RQセクション, 58

S

SCA, 12, 30
SERVERセクション, 58
SERVICEセクション, 58
SLM, 10
SLMIによる負荷調整, 18
STRINGバッファ, 48
STRUCTバッファ, 48
SVRGROUPセクション, 58

T

TCS, 7, 10, 25
TIM, 8
TIP, 7
TIP (Tmax Information Provider), 155
TLM, 7
TM, 16
Tmaxadmin, 7
Tmax, 5
Tmax ATMI, 61
tmaxmt Library, 34
tmaxreadenv(), 50
TMM, 6
TMS, 7
TP-Monitor, 3
tpacall(), 51
tpalloc(), 51
tpcall(), 51
tpend, 51
tpforward(), 55
tpfree(), 51
tpgetreply(), 51
tpreturn(), 55
tpstart(), 51
tpsvrdone(), 55
tpsvrinit(), 55
TPモニター, 1

U

UCS, 7, 11, 25

W

Web Application Server, 3
WebT, 43
WebT Library, 44
WebT-Serverシステム, 44
WebTConnectionPool, 43
Window制御, 12
WinTmax Library, 34

X

X/Open ATMI, 59
X/Open DTP, 5
X/Open TX API, 60
X_C_TYPEバッファ, 48
X_COMMONバッファ, 48
X_OCTETバッファ, 48

か

対話型通信, 31
クライアント・プログラムのフロー, 49
クライアント・プログラムの構成, 52
グローバル・トランザクション, 17, 84
グローバル・トランザクション-コミットフェーズ(Commit Phase), 18
グローバル・トランザクション-準備フェーズ(Prepare Phase), 18

さ

障害対策, 10
サーバー・プログラムのフロー, 53
サーバー・プログラムの構成, 56
システム管理, 26
セキュリティ機能, 11
ソフトウェア障害, 24

た

伝達型通信, 32
同期通信, 30
トランザクション-コミットフェーズ(Commit Phase), 17
トランザクション-準備フェーズ(Prepare Phase), 17
トランザクション管理, 10

な

ネーミング・サービス, 10

は

非同期通信, 31

負荷調整, 10

分散トランザクション管理, 16

ハードウェア障害, 21

プロセス制御, 10

プロセス管理, 10

ま

ミドルウェア, 2

ら

リソース管理, 27

