

Tmax ゲートウェイガイド (TCP/IPスレッド)

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 ゲートウェイガイド (TCP/IPスレッド)

発行日: 2016年8月5日

ソフトウェアバージョン: Tmax v6.0

ガイドバージョン: v2.1.1

目次

このガイドについて	ix
第1章 紹介	1
1.1. 概要	1
1.2. サービスのタイプ	3
1.2.1. サーバー・モード	3
1.2.2. クライアント・モード	5
第2章 環境設定	9
2.1. 概要	9
2.2. Tmax環境構成	9
2.3. TCPGWTHR環境ファイル	13
2.3.1. アドレス情報の環境ファイル	13
第3章 ユーザーAPI関数	15
3.1. コールバック関数	15
3.1.1. user_thrmain	15
3.2. API関数	16
3.2.1. tcpgw_tpcall	16
3.2.2. tcpgw_tpacall	17
3.2.3. tcpgw_tpreply	17
3.2.4. tcpgw_get_svcddata	18
3.2.5. tcpgw_select	18
3.2.6. tcpgw_network_connect	19
3.2.7. tcpgw_read	20
3.2.8. tcpgw_write	20
3.2.9. tcpgw_portno_ipaddr	21
3.2.10. tcpgw_client_id	21
3.2.11. tcpgw_getaddr_from_winfo	22
第4章 例	23
4.1. サーバー・モードのアウトバウンド・サービス呼び出し	23
4.1.1. 環境ファイル	23
4.1.2. TCPGWTHR	24
4.1.3. リモート・ノード	27
4.1.4. Tmaxノード	31
4.2. サーバー・モードのインバウンド・サービス呼び出し	32
4.2.1. 環境ファイル	32
4.2.2. リモート・ノード	33
4.2.3. Tmaxノード	37
4.3. クライアント・モードのインバウンド・サービス呼び出し	38
4.3.1. 環境ファイル	39
4.3.2. リモート・ノード	39

付録 A. ヘッダー・ファイル	45
A.1. TCPHDR.Hファイル	45
索引	49

図目次

[図 1.1]	TCPGWTHRの動作構造	2
[図 1.2]	サーバー・モードのTCPGWTHRの動作構造 - shared方式のクライアント接続	3
[図 1.3]	サーバー・モードのTCPGWTHRの動作構造 - dedicated方式のクライアント接続	4
[図 1.4]	クライアント・モードのTCPGWTHRの動作構造 - シングル・サーバー接続	5
[図 1.5]	クライアント・モードのTCPGWTHRの動作構造 - マルチサーバー接続	6

このガイドについて

対象読者

本書は、Tmax[®](以下、Tmax) TCP/IPスレッド・ゲートウェイを使用して開発するユーザーを対象としています。同書では、非TmaxクライアントとTmaxの間でインターフェースの役割をする、TCP/IPスレッド・ゲートウェイについて記述しています。

前提知識

本書は、Tmaxシステムに関する全般的な内容と、Tmaxシステムが提供している各種機能および特性を習得するためのガイドです。

本書を理解するには、事前に下記の内容を把握しておく必要があります。

- ミドルウェアおよびUNIXシステムに関する知識
- Tmaxの基本概念に関する知識
- Java、Cプログラミングに関する知識

制限事項

本書を読む前に、Tmaxの基本概念について熟知している必要があります。実務における詳細な使用方法および管理、運用に関する内容については、各製品のガイドを参照してください。

参考

Tmaxシステム開発に関する基本的な内容は、『Tmax 運用ガイド』、または『Tmax アプリケーション開発ガイド』を参照してください。Tmaxが提供しているコマンドとC APIに関する説明については、『Tmax リファレンスガイド』を参照してください。

本書の構成

Tmax ゲートウェイガイド(TCP/IPスレッド)は、計4章と付録で構成されています。

各章の主な内容は下記のとおりです。

- 第1章: 紹介

システムの概要およびプロセス、サービスのタイプについて記述します。

- 第2章: 環境設定

TCP/IPスレッド・ゲートウェイの環境構成および環境ファイルの設定について記述します。

- 第3章: ユーザーAPI関数

TCP/IPスレッド・ゲートウェイで使用するAPI関数について記述します。

- 第4章: 例

TCP/IPスレッド・ゲートウェイの各役割について記述します。

- 付録A: ヘッダー・ファイル

TCPHDR.Hファイルについて記述します。

表記上の規則

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

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

関連ガイド

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

お問合せ先

Korea

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

USA

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

Japan

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

China

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

Brazil

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

Russia

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

Singapore

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

United Kingdom

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

Canada

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

Australia

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

India

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

Turkey

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

第1章 紹介

本章では、システムの概要およびプロセス、サービスのタイプについて説明します。

1.1. 概要

TCP/IPスレッド・ゲートウェイ(TCP/IP Thread Gateway、以下TCPGWTHR)は、非Tmaxクライアント(非Tmaxサーバー)とTmaxの間でインターフェースの役割をします。すなわち、非Tmaxクライアントとゲートウェイ間では一般的なTCP/IP通信をし、ゲートウェイとTmaxシステム間にはTmax通信をします。

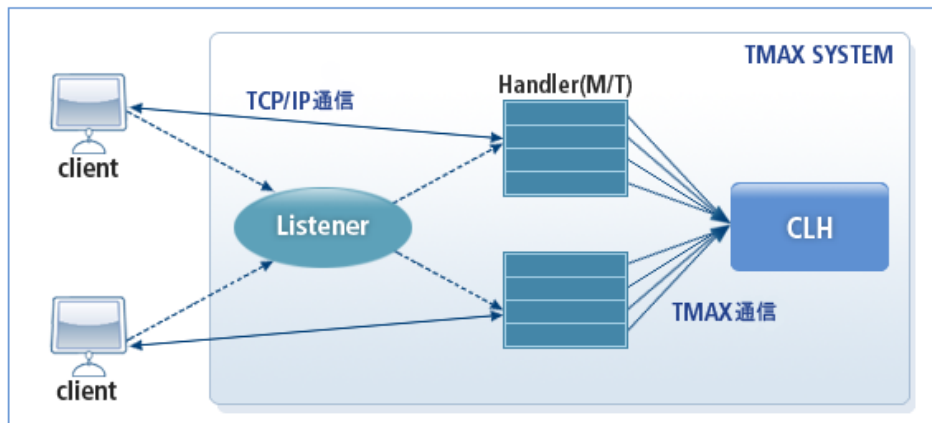
ゲートウェイは使用方法によってサーバーで動作することも、クライアントで動作することもできます。これは、サービス要求ではなく接続の要求がどこから開始するのかを意味します。ゲートウェイがサーバーで動作する場合はリモートで接続を要求する場合で、逆の場合はゲートウェイ(スレッド)で接続を要求する場合です。どちらからでも、接続さえできれば相互にサービスを要求することが可能です。

また、クライアントとハンドラーのスレッドを割り当てし、接続する方式なので、クライアントとスレッドの間にユーザーが任意でプロトコルを追加することができます。Tmaxサービスでクライアントを要求する際、ラウンドロビン方式でクライアントと接続されているアイドルなスレッドを内部的に見つけて渡す方法と、特定のクライアントと接続されているスレッドに渡す方法を提供します。ゲートウェイはTmax内に含まれ、統合的に管理されます。

以前使用していたTCP/IPゲートウェイは、1つのプロセスでリモートと複数のチャンネルを接続させ、単純にRequest/Responseのみ送受信する形式です。一方、TCPGWTHRは、ユーザーが任意でプロトコルが追加でき、さらに多様な環境で 사용할 ことが可能です。RCA(Raw Client Agent)は、Tmaxにクライアントで接続するため、Tmaxにサービスを要求することは可能ですが、RCAにサービスを要求することはできません。しかし、TCPGWTHRは、双方向のサービス要求が可能です。

下記は、TCPGWTHRの動作構造です。

【図 1.1】 TCPGWTHRの動作構造



TCPGWTHRは、サーバー・モードの場合はリスナー(Listener)プロセスとハンドラー(Handler)プロセスで構成されており、クライアント・モードの場合は、クライアント共有メモリー管理プロセス(以下、Clishm)とハンドラー・プロセスで構成されています。

- リスナー・プロセス

リスナー・プロセスは、TCPGWTHRがサーバーで動作する場合に必要なプロセスです。リスナーは、与えられたポートをリスンしており、外部から接続要求が受信されたら、これを受けてハンドラー・プロセスに渡す役割をします。さらに、Tmaxシステムと接続されTmaxシステムが起動したら実行され、ダウンすると終了されます。

- ハンドラー・プロセス

非Tmaxクライアント(非Tmaxサーバー)と接続して実質的なジョブを処理するプロセスです。リスナー・プロセスから渡されたソケットを内部のスレッドに割り当て、割り当てられたスレッドがジョブを処理するか、または特定のスレッドが非Tmaxクライアントと接続してジョブを処理します。

ハンドラー・プロセスの内部には多くのスレッドが存在しており、それぞれのスレッドはリモートと接続されます。TCPGWTHRがサーバーで動作する場合は、リスナー・プロセスからソケットが渡されます。また、TCPGWTHRがクライアントで動作する場合は、ハンドラー・プロセスの内部に存在するそれぞれのスレッドがリモートと接続します。接続されたスレッドは、非Tmaxクライアントから要求を受けたり、Tmaxサービスに要求したりすることができます。

ハンドラー・プロセスは、1プロセスあたり最大50個までのスレッドが作成できます。

- Clishmプロセス

クライアント・モードでリスナー・プロセスの代わりに使用されており、共有メモリーを初期化するために必要です。Clishmプロセスも、Tmaxシステムと接続されTmaxシステムが起動したら実行され、ダウンすると終了されます。

1.2. サービスのタイプ

1.2.1. サーバー・モード

サーバー・モードでは、下記の2つの接続方式でクライアントに接続することができます。

- shared方式
- dedicated方式

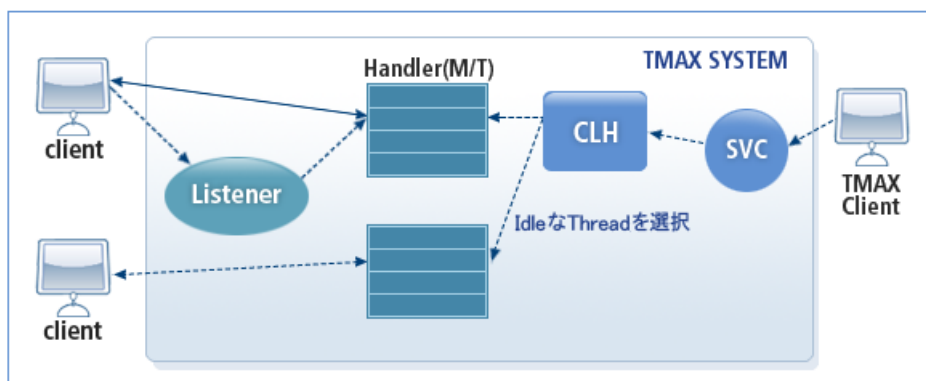
クライアント接続(shared方式)

サーバーで動作する場合は、リスナー・プロセスが必要となります。リスナー・プロセスは、非Tmaxクライアントから受信される接続要求を受け、ハンドラー・プロセスに渡す役割をします。リスナー・プロセスは受信される要求に対して、ラウンドロビン方式でハンドラー・プロセスに均一に割り当てます。

Tmaxサービスからゲートウェイ・サービスを呼び出す場合、ラウンドロビン方式でクライアントと接続されているハンドラー・プロセスのアイドルなスレッドの中で1つのスレッドに渡します。

下記は、サーバー・モードでshared方式を利用してクライアントを接続する場合のTCPGWTHRの動作構造です。

【図 1.2】サーバー・モードのTCPGWTHRの動作構造 - shared方式のクライアント接続



下記は、サーバー・モードでshared方式を利用してクライアントを接続する場合のTmax環境ファイルの例です。

```
*SERVER
tcpgwlsn  SVGNAME = svg1, MIN = 1, MAX = 1,
          SVRTYPE = CUSTOM_GATEWAY, RESTART = N,
          CLOPT = "-- -P 1029 -N 3 -k 98765"

tcpgwhdr1 SVNAME = svg1, MIN = 3, MAX = 3, SCHEDULE = RR,
          SVRTYPE = CUSTOM_GATEWAY, CPC = 10, TARGET = tcpgwhdr,
```

```

CLOPT = "-- -P 1029 -N 10 -s -L tcpgwlsn"

*SERVICE
svcgw      SVRNAME = tcpgwhdr1

```

クライアント接続(dedicated方式)

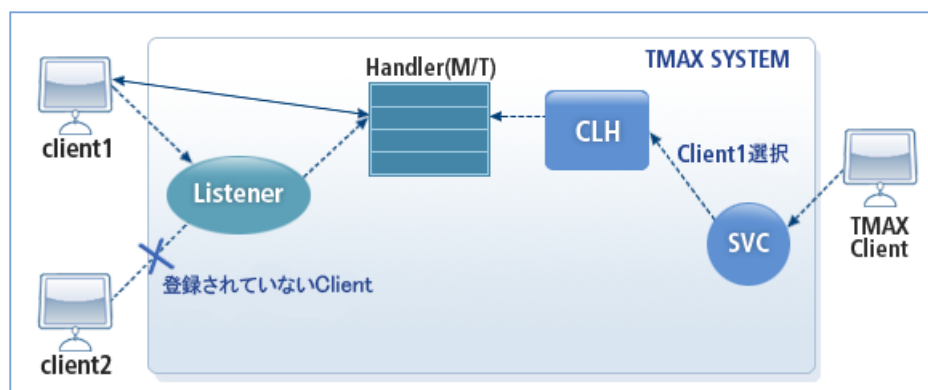
サーバーで動作する場合はリスナー・プロセスが必要となります。リスナー・プロセスは、非Tmaxクライアントから受信される接続要求を受け、当該クライアントがゲートウェイ環境ファイルに登録されているか否かを確認した後、ハンドラー・プロセスに渡してスレッドを割り当てます。登録されていない場合は警告メッセージを出力した後、接続を断ります。この過程で接続を要求したクライアント情報を共有メモリーで検索し、登録されていない場合はゲートウェイ環境ファイルを内部的に再検索するため、動的に追加することができます。

Tmaxサービスでクライアントを区別するために1IPあたり、1つの接続のみ許可します。例えば、クライアントとスレッドが接続されている状態で、同じクライアントから追加で接続を要求する場合は、既存の接続は切断され、追加要求もキャンセルされます。

Tmaxサービスからゲートウェイ・サービスを呼び出す場合、ゲートウェイ環境ファイルに登録されているクライアントIDを選択し、呼び出し時にゲートウェイの内部的に接続されているスレッドを見つけて渡します。ハンドラー・プロセスのスレッドは、ゲートウェイ環境ファイルに登録されているクライアント数と動的に接続を許可するために、追加で50個が割り当てられます。なお、1ハンドラー・プロセスあたりのスレッド数が50を超えないようにハンドラー・プロセス数を調整する必要があります。

下記は、サーバー・モードでdedicated方式を利用してクライアントを接続する場合のTCPGWTHRの動作構造です。

[図 1.3] サーバー・モードのTCPGWTHRの動作構造 - dedicated方式のクライアント接続



下記は、サーバー・モードでdedicated方式を利用してクライアントを接続する場合のTmax環境ファイルの例です。

```

*SERVER
tcpgwlsn  SVGNAME = svg1, MIN = 1, MAX = 1,

```

```

SVRTYPE = CUSTOM_GATEWAY, RESTART = N,
CLOPT = "-- -P 1029 -N 3 -k 98765 -F tcpgwlsn.dat"
tcpgwhdr1 SVNAME = svgl, MIN = 3, MAX = 3, SCHEDULE = RR,
SVRTYPE = CUSTOM_GATEWAY, CPC = 10, TARGET = tcpgwhdr,
CLOPT = "-- -P 1029 -s -L tcpgwlsn"

*SERVICE
svcgw      SVRNAME = tcpgwhdr1

```

下記は、サーバー・モードでdedicated方式を利用してクライアントを接続する場合のゲートウェイ環境ファイルの例です。

# Client IP	Server Port	Client ID
61.33.32.123	1029	CLI1
61.33.32.124	1029	CLI2
61.33.32.125	1029	CLI3
61.33.32.126	1029	CLI4

1.2.2. クライアント・モード

クライアント・モードでは、接続するサーバーの数によって下記のように区分されます。

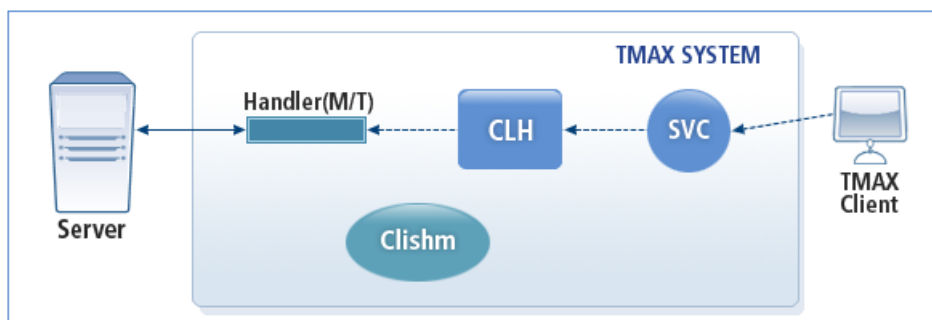
- シングル・サーバー接続
- マルチサーバー接続

シングル・サーバー接続

クライアントで動作する場合は、リスナー・プロセスの代わりにClishmプロセスを使用します。非Tmaxサーバーに接続する場合は、Tmax環境ファイルに接続するサーバーIPとポートを登録します。

下記は、クライアント・モードでシングル・サーバーを接続する場合のTCPGWTHRの動作構造です。

[図 1.4] クライアント・モードのTCPGWTHRの動作構造 - シングル・サーバー接続



下記は、クライアント・モードでシングル・サーバー接続方式を利用してクライアントを接続する場合のTmax環境ファイルの例です。

```
*SERVER
clihdrshm  SVGNAME = svg1, MIN = 1, MAX = 1,
           SVRTYPE = CUSTOM_GATEWAY,
           CLOPT = "-- -N 1 -k 91000"
tcpgwhdr1  SVNAME = svg1, MIN = 1, MAX = 1,
           SVRTYPE = CUSTOM_GATEWAY,
           CPC = 9, SCHEDULE = RR, TARGET = tcpgwhdr,
           CLOPT = "-- -k 91000 -P 3777 -r 100.100.100.1 -N 1 -L clihdrshm"

*SERVICE
svcgw      SVRNAME = tcpgwhdr1
```

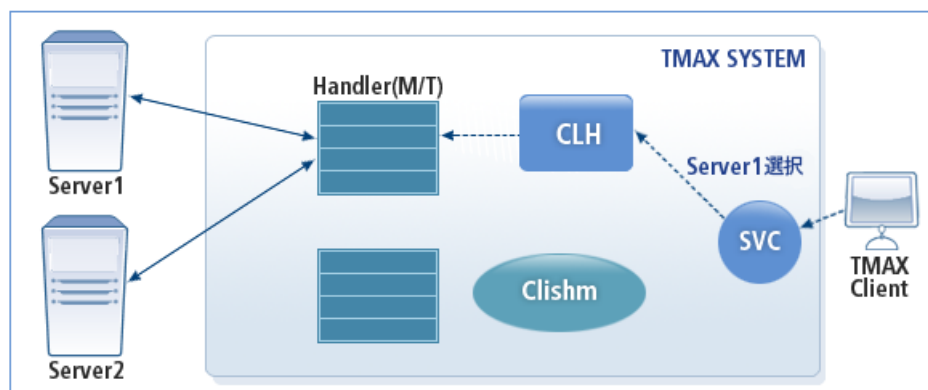
マルチサーバー接続

クライアントで動作する場合は、リスナー・プロセスの代わりにClishmプロセスを使用します。複数の非Tmaxサーバーに接続する場合は、ゲートウェイ環境ファイルを利用して接続するサーバーの情報を登録します。

Tmaxサービスからゲートウェイ・サービスを呼び出す場合、割り当てられているスレッドの有無を確認し、割り当てられているスレッドが存在しない場合は、アイドルなスレッドを割り当てし非Tmaxサーバーに接続します。

下記は、クライアント・モードでマルチサーバーを接続する場合のTCPGWTHRの動作構造です。

[図 1.5] クライアント・モードのTCPGWTHRの動作構造 - マルチサーバー接続



下記は、クライアント・モードでマルチサーバー接続方式を利用してクライアントを接続する場合のTmax環境ファイルの例です。

```
*SERVER
clihdrshm  SVGNAME = svg1, MIN=1, MAX=1,
           SVRTYPE=CUSTOM_GATEWAY,
           CLOPT="-- -N 10 -k 91000 -F tcpgwhdr.dat"
tcpgwhdr1  SVGNAME svg1, MIN=10, MAX=10, SVRTYPE=CUSTOM_GATEWAY,
```



```
CPC = 9, SCHEDULE = RR, TARGET = tcpgwhdr,  
CLOPT = "-- -k 91029 -F tcpgwhdr.dat -L clihdrshm"  
  
*SERVICE  
svcgw      SVRNAME = tcpgwhdr1
```

下記は、クライアント・モードでマルチサーバー接続方式を利用してクライアントを接続する場合のゲートウェイ環境ファイルの例です。

#	Server IP	Server Port	Server ID
61.33.32.123	9000	SERVER1	
61.33.32.124	9100	SERVER2	

第2章 環境設定

本章では、TCPGWTHRの環境構成および環境ファイルの設定について説明します。

2.1. 概要

TCPGWTHRサーバーを構成するには、オペレーティング・システムによって下記のようなファイルが必要となります。

- UNIX

ディレクトリー	ファイル名
lib	libtcpgwhdr.a, libtcpgwhdr.so, libtmaxgw.a, libtmaxgw.so, libtmaxgwmt.a, libtmaxgwmt.so
lib64	libtcpgwhdr.a, libtcpgwhdr.so, libtmaxgw.a, libtmaxgw.so, libtmaxgwmt.a, libtmaxgwmt.so
appbin	tcpgwlsn, clihdrshm, tcpgwlsn64, clhhdrshm64
usrinc	tcphdr.h, hlinkapi.h

- Windows

ディレクトリー	ファイル名
lib	tcpgwhdr.lib, tcpgwhdr.dll, tmaxgw.lib, tmaxgw.dll, tmaxgwmt.lib, tmaxgwmt.dll
bin	tcpgwlsn.exe, clihdrshm.exe
usrinc	tcphdr.h, hlinkapi.h

2.2. Tmax環境構成

TCPGWTHRを使用するには、Tmax環境ファイルにリスナー・プロセス、またはClishmプロセスおよびハンドラー・プロセスをサーバーとして登録する必要があります。Tmaxサーバーのうち、UCS方式のサーバーと登録方法が似ていますが、SVRTYPEが「UCS」ではなく「CUSTOM_GATEWAY」という点以外に相違点はありません。TCPGWTHRを使用するためTmax環境ファイルを修正する際には、SERVERセクションとSERVICEセクションのみ設定すれば使用できます。

下記は、Tmax環境ファイルの例です。

```
*DOMAIN
tmax      SHMKEY = 88000,
          MINCLH = 1,
          MAXCLH = 1,
          TPORTNO = 8800

*NODE
tmax1     TMAXDIR = "/home/tmax",
          APPDIR = "/home/tmax/appbin"

*SVRGROUP
svg1      NODENAME = tmax1

*SERVER
tcpgwlsn  SVGNAME = svg1, MIN = 1, MAX = 1,
          SVRTYPE = CUSTOM_GATEWAY, RESTART = N,
          CLOPT = "-- -P 1029 -N 3 -k 98765"

tcpgwhdr1 SVNAME = svg1, MIN = 3, MAX = 3, SCHEDULE = RR,
          SVRTYPE = CUSTOM_GATEWAY, CPC = 10, TARGET = tcpgwhdr,
          CLOPT = "-- -P 1029 -N 10 -s -L tcpgwlsn"

*SERVICE
svcgw     SVRNAME = tcpgwhdr1
```

CLOPT項目(TCPGWTHRオプション)

TCPGWTHRは、Tmax環境ファイルに登録できる項目が制限されているため、CLOPT項目にいくつかのオプションを設定する必要があります。オプションによってTCPGWTHRの動作方式が異なるため、下記の説明を十分に理解する必要があります。

- リスナー・オプション

下記は、リスナー・オプションに関する説明です。

オプション	説明
[-F]	オプションを使用するとdedicateモードで動作します。ゲートウェイ環境ファイルを入力するときは、フルパス(Full Path)で入力します。ファイルの登録方法については、 「2.3. TCPGWTHR環境ファイル」 を参照してください
[-P]	リスンするポート番号です
[-k]	ゲートウェイ内で情報を共有するために使用される共有メモリーのためのキー値です。入力したキーとキー+1を使用します

オプション	説明
[-N]	ハンドラー・プロセス数です。ハンドラー・プロセスのMIN値を適用します。ハンドラー・プロセスのMIN、MAXは同一にします
[-w]	システムがWindowsの場合、ゲートウェイの内部的に使用するポート番号です
[-X SERVER_IPV6=[IPV4 IPV6 SDP]]	Listenソケットを作成時に使用するプロトコルを指定するオプションです。(デフォルト値: 'IPV4') <ul style="list-style-type: none"> - 'IPV6'または'Y'を指定すると、IPv6プロトコルを使用します - 'IPV4'または'N'を指定するか、オプションを指定しない場合は、IPv4プロトコルを使用します - InfiniBandのSDP(Socket Direct Protocol)を使用する場合は、'SDP'を指定します

● ハンドラー・オプション

下記は、ハンドラー・オプションに関する説明です。

オプション	説明
[-F]	クライアント・モードでマルチ接続をする場合、ゲートウェイ環境ファイルを指定します。ゲートウェイ環境ファイルを入力するときは、フスパスで入力します。ファイルの登録方法については、 「2.3. TCPGWTHR環境ファイル」 を参照してください。IPv6プロトコルをサポートします
[-r]	クライアント・モード(シングル接続)の場合に接続する非TmaxサーバーのIPまたはホスト名です。IPv6プロトコルをサポートします
[-P]	サーバーモードの場合はリスナーのポート番号です。クライアント・モード(シングル接続)の場合は、接続する非Tmaxサーバーのポート番号です
[-k]	ゲートウェイ内で情報を共有するために使用される共有メモリーのためのキー値です。入力したキーとキー+1を使用します
[-s]	ハンドラー・プロセスがサーバー・モードで動作するか、クライアント・モードで動作するかを指定します
[-N]	ハンドラー・プロセスは1ハンドラーあたりのスレッド数を指定します。サーバー・モードでゲートウェイ環境ファイルを指定した場合は、ファイルを参照して1ハンドラーあたりのスレッド数を計算して割り当てできるため、指定する必要はありません。 クライアント・モードでは、シングル接続の場合は1を指定し、マルチ接続の場合は接続しようとする数の分だけスレッド数を入力しますが、入力

オプション	説明
	しなくても構いません。入力しない場合は、ハンドラーの[-F]オプションに指定されたゲートウェイ環境ファイルを参照して割り当てられます
[-w]	システムがWindowsの場合、ゲートウェイの内部的に使用するポート番号です
[-i]	TCPGWTHRで、チャンネルIRT機能が動作するようにします。COUSINで構成されており、TCPGWTHRに接続されているリモートが1つも存在しない場合は、接続が存在している別のTCPGWTHRに要求が渡されます
[-f]	Dedicatedモードで動作する場合、基本的に1IPあたり1つの接続のみ許可します。すなわち、特定のIPと接続されている状態で、同じIPで新しい接続が要求されると、既存の接続および新しい接続の両方とも解除されることとなります。一方、このオプションを使用すると、既存の接続は解除されますが、新しい接続には成功するため、新しい接続を利用して通信することとなります
[-X CLIENT_IPV6=["IPV4" "IPV6" "SDP"]]	リモートに接続時に使用するプロトコルを指定するオプションです。(デフォルト値: 'IPV4') <ul style="list-style-type: none"> - 'IPV6'または'Y'を指定すると、IPv6プロトコルを使用します - 'IPV4'または'N'を指定するか、オプションを指定しない場合は、IPv4プロトコルを使用します - InfiniBandのSDP(Socket Direct Protocol)を使用する場合は、'SDP'を指定します

● Clishmオプション

下記は、Clishmオプションに関する説明です。

オプション	説明
[-F]	オプションを使用するとdedicateモードで動作します。ゲートウェイ環境ファイルを入力するときは、フルパスで入力します。ファイルの登録方法については、 「2.3. TCPGWTHR環境ファイル」 を参照してください
[-k]	ゲートウェイ内で情報を共有するために使用される共有メモリーのためのキー値です。入力したキーとキー+1を使用します
[-N]	ハンドラー・プロセス数です。ハンドラー・プロセスのMIN値を適用します。ハンドラー・プロセスのMIN、MAXは同一にします
[-w]	システムがWindowsの場合、ゲートウェイの内部的に使用するポート番号です

2.3. TCPGWTHR環境ファイル

Tmax環境ファイルのCLOPTセクションに[-F]オプションで指定したファイルは、下記で説明するフォーマットで登録する必要があります。ファイルはリモート・ノードのアドレスとサーバーIDを登録するファイルです。

2.3.1. アドレス情報の環境ファイル

アドレス情報の環境ファイルは、モードによって下記のように設定します。

● クライアント・モード

クライアント・モードでマルチサーバーに接続する場合は、下記のように設定します。

```
#####
#          TCP/IP Thread Gateway Config          #
#####
# Server IP  Server Port  Server ID
# 1.1.1.1      9000          SERVER1
#####
# line start with "#" is comment line
#####
61.33.32.123  9000          SERVER1
61.33.32.124  9100          SERVER2
```

下記は、IPv6プロトコル環境の設定例です。IPv4とIPv6のアドレスを下記のように設定できます。

```
# Server IP      Server Port  Server ID
2011::100:100    9000          SERVER1
2011::100:200    9100          SERVER2
61.33.32.123     9200          SERVER3
61.33.32.124     9300          SERVER4
```

下記は、アドレス情報の環境ファイルの項目についての説明です。

項目	説明
Server IP	リモート・ノードのIPアドレスを登録します
Server Port	リモート・ノードのポート番号を登録します
Server ID	Tmaxサービスからゲートウェイ・サービスを呼び出す場合、dedicated方式でチャンネルを選択するために必要なIDです。最大の長さは31字です。

● サーバー・モード

サーバー・モードでdedicated方式で接続する場合は、下記のように設定します。

```
#####
#          TCP/IP Thread Gateway Config          #
#####
# Client IP  Server Port  Client ID
# 1.1.1.1      9000          CLIENT1
#####
# line start with "#" is comment line
#####
61.33.32.123  9000          CLIENT1
61.33.32.124  9100          CLIENT2
```

リスナーの「**-X SERVER_IPV6**」設定によってIPv4またはIPv6、SDPのうち1つの方式でのみ設定が可能です。リスナーがIPv6プロトコルを使用する場合は、クライアントのアドレスをすべてIPv6形式で作成する必要があります。

```
# Client IP      Server Port  Client ID
2011::100:100    9000          CLIENT1
2011::100:200    9100          CLIENT2
2011::100:300    9200          CLIENT3
2011::100:400    9300          CLIENT4
```


第3章 ユーザーAPI関数

本章では、ユーザーusrmain()プログラムで利用できるAPI関数について説明します。

3.1. コールバック関数

3.1.1. user_thrmain

ユーザーが作成するスレッド・メイン・コールバック関数です。TCPGWTHRは、ユーザーが作成したプログラムをスレッド形式で実行します。

- プロトタイプ

```
int user_thrmain(WORKTHRINFO *wthrinfo, int server)
```

- パラメータ

パラメータ	説明
wthrinfo	スレッドが渡したWORKTHRINFOのポインターです
server	スレッドがサーバー・モードか、あるいはクライアント・モードかを示す変数です – 0 : クライアント・モード – 1 : サーバー・モード

- 戻り値

戻り値に使用されるマクロは、tcphdr.hヘッダ・ファイルに定義されています。

戻り値	説明
WTHR_NORMAL_RETURN	当該スレッドが正常に完了された場合です。エラー・ログは発生せず、サーバーまたはクライアントとの接続が正常に終了されます
WTHR_ABNORMAL_RETURN	当該スレッドが異常終了した場合です。エラー・ログがslogに記録され、接続が終了されます

3.2. API関数

下記は、API関数の一覧です。関数のプロトタイプは、tcphdr.hヘッダー・ファイルに定義されています。

関数名	説明
tcpgw_tpcall	Tmaxサービスの同期型呼び出し関数です
tcpgw_tpacall	Tmaxサービスの非同期型呼び出し関数です
tcpgw_tpreply	Tmaxに結果を返すための関数です
tcpgw_get_svcddata	Tmaxからのデータを受信するための関数です
tcpgw_select	スレッドの要求を受けるまで、決められた時間の間待機する関数です
tcpgw_network_connect	TCPGWTHRがクライアント・モードで動作する場合、リモートと接続するための関数です
tcpgw_read	リモートからデータを受信する関数です
tcpgw_write	リモートにデータを転送する関数です
tcpgw_portno_ipaddr	リモートのアドレスとポート番号を取得する関数です
tcpgw_client_id	スレッドに割り当てられたゲートウェイ環境ファイルに入力したIDを取得する関数です
tcpgw_getaddr_from_wininfo	スレッドが渡したWORKTHRINFO構造体から、リモートのアドレスとポート番号を取得する関数です

3.2.1. tcpgw_tpcall

Tmaxサービスの同期型呼び出し関数です。

- プロトタイプ

```
int tcpgw_tpcall(WORKTHRINFO *wthrinfo, char *svc, char *ptr, long alen,
                 char *optr, long *olen)
```

- パラメータ

パラメータ	説明
wthrinfo	スレッドが渡したWORKTHRINFOのポインターです
svc	Tmaxサービス名です
ptr	Tmaxサービスに渡すデータが保存されているバッファ・ポインターです
alen	Tmaxサービスに渡すデータ長です
optr	Tmaxサービスから処理結果を受けるバッファ・ポインターです
olen	処理結果のデータ長が保存されるロング・ポインターです

- 戻り値

Tmaxサービス进行处理する際にエラーが発生すると、戻り値としてエラーコードが返されます。戻り値が「-1」の場合は、異常エラーが発生した場合です。以外のminus値は、Tmaxでサービス进行处理する際にエラーが発生した場合のエラーコードです(符号を除いた実際のエラーコード値)。

3.2.2. tcpgw_tpacall

Tmaxサービスの非同期呼び出し関数です。要求のみ可能で結果を受けることはできません(TPNOREPLY)。

- プロトタイプ

```
int tcpgw_tpacall(WORKTHRINFO *wthrinfo, char *svc, char *ptr, long alen)
```

- パラメータ

パラメータ	説明
wthrinfo	スレッドが渡したWORKTHRINFOのポインターです
svc	Tmaxサービス名です
ptr	Tmaxサービスに渡すデータが保存されているバッファ・ポインターです
alen	Tmaxサービスに渡すデータ長です

- 戻り値

Tmaxサービス进行处理する際にエラーが発生すると、戻り値としてエラーコードが返されます。戻り値が「-1」の場合は、異常エラーが発生した場合です。以外のminus値は、Tmaxでサービス进行处理する際にエラーが発生した場合のエラーコードです(符号を除いた実際のエラーコード値)。

3.2.3. tcpgw_tpreply

Tmaxに結果を返すための関数です。Tmaxからサービス要求が先に受信された場合、リモート・ノードでサービス进行处理し、その結果をTmaxに転送します。

- プロトタイプ

```
int tcpgw_tpreply(WORKTHRINFO *wthrinfo, char *ptr, long alen, int err)
```

- パラメータ

パラメータ	説明
wthrinfo	スレッドが渡したWORKTHRINFOのポインターです

パラメータ	説明
ptr	Tmaxサービスに渡すデータが保存されているバッファ・ポインターです
alen	Tmaxサービスに渡すデータ長です
err	サービスを処理する際にエラーが発生する場合のエラーコードです

- 戻り値

Tmaxサービスを処理する際にエラーが発生すると、戻り値としてエラーコードが返されます。戻り値が「-1」の場合は、異常エラーが発生した場合です。以外のminus値は、Tmaxでサービスを処理する際にエラーが発生した場合のエラーコードです(符号を除いた実際のエラーコード値)。

3.2.4. tcpgw_get_svcddata

Tmaxからのデータを受信するための関数です。

- プロトタイプ

```
int tcpgw_get_svcddata(WORKTHRINFO *wthrinfo, char *ptr, int *err, int *flags)
```

- パラメータ

パラメータ	説明
wthrinfo	スレッドが渡したWORKTHRINFOのポインターです
ptr	データが保存されるバッファです
err	エラーコードが保存されるintポインターです
flags	Tmaxでサービスを要求する際、応答を受けるか否かを指定します。ゼロの場合はTmaxサービスで応答を待つ状態で、1の場合は応答を待たない状態です

- 戻り値

戻り値	説明
データ長	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

3.2.5. tcpgw_select

スレッドの要求を受けるまで、決められた時間の間待機する関数です。

- プロトタイプ

```
int tcpgw_select(WORKTHRINFO *wthrinfo, int sec, int usec)
```

- パラメータ

パラメータ	説明
wthrinfo	スレッドが渡したWORKTHRINFOのポインターです
sec	タイムアウトです(単位: 秒)
usec	タイムアウトです(単位: 100万分の1秒)

- 戻り値

戻り値	説明
WTHR_SELECT_TIMEOUT(1)	タイムアウトの場合です
WTHR_TMAX_REQUEST(2)	Tmax要求の場合です
WTHR_USER_REQUEST(3)	リモート要求の場合です
WTHR_SELECT_ERROR(-1)	その他のエラーの場合です

3.2.6. tcpgw_network_connect

TCPGWTHRがクライアント・モードで動作する場合、リモートと接続するための関数です。

- プロトタイプ

```
int tcpgw_network_connect(char *host, int port, int sec)
```

- パラメータ

パラメータ	説明
host	リモートと接続するアドレスまたはリモート・ホスト名です。IPv6アドレスをサポートします
port	リモートと接続するポート番号です
sec	タイムアウトです(単位: 秒)

- 戻り値

戻り値	説明
ソケット番号	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

3.2.7. tcpgw_read

リモートからデータを受信する関数です。

- プロトタイプ

```
int tcpgw_read(int fd, char *ptr, int nbytes, int sec, int usec)
```

- パラメータ

パラメータ	説明
fd	リモート・ソケット番号です
ptr	データが保存されているバッファ・ポインターです
nbytes	受信するデータ長です
sec	タイムアウトです(単位: 秒)
usec	タイムアウトです(単位: 100万分の1秒)

- 戻り値

戻り値	説明
データ長	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

3.2.8. tcpgw_write

リモートにデータを転送する関数です。

- プロトタイプ

```
int tcpgw_write(int fd, char *ptr, int nbytes)
```

- パラメータ

パラメータ	説明
fd	リモート・ソケット番号です
ptr	データが保存されているバッファ・ポインターです
nbytes	リモートで受信するデータ長です

- 戻り値

戻り値	説明
データ長	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

3.2.9. tcpgw_portno_ipaddr

リモートのアドレスとポート番号を取得する関数です。

- プロトタイプ

```
int tcpgw_portno_ipaddr(char *ipaddr)
```

- パラメータ

パラメータ	説明
ipaddr	リモートのアドレスが保存されるバッファです

- 戻り値

環境設定に指定されたリモートのポート番号が返されます。

3.2.10. tcpgw_client_id

スレッドに割り当てられたゲートウェイ環境ファイルに入力したIDを取得する関数です。

- プロトタイプ

```
int tcpgw_client_id (WORKTHRINFO *wthrinfo, char *id)
```

- パラメータ

パラメータ	説明
wthrinfo	スレッドが渡したWORKTHRINFOのポインターです
id	IDが保存されるバッファです

- 戻り値

使用されません。

3.2.11. tcpgw_getaddr_from_winfo

スレッドが渡したWORKTHRINFO構造体から、リモートのアドレスとポート番号を取得する関数です。

- プロトタイプ

```
int tcpgw_getaddr_from_winfo(WORKTHRINFO *wthrinfo, char *ipaddr)
```

- パラメータ

パラメータ	説明
wthrinfo	スレッドが渡したWORKTHRINFOのポインターです
ipaddr	リモートのアドレスが保存されるバッファです。IPv6アドレスを設定した場合は、十分なバッファ・サイズを割り当てる必要があります

- 戻り値

環境設定に指定されたリモートのポート番号が返されます。

第4章 例

本章では、第1章で説明したTCPGWTHRの各役割に関する例について説明します。

4.1. サーバー・モードのアウトバウンド・サービス呼び出し

サーバー・モードのTCPGWTHRを利用して、リモートからTmaxのサービスを要求する例です。

プログラムの構成は下記のとおりです。

区分	ファイル名
環境ファイル	tmax.m
TCPGWTHR	usermain.c
リモート・ノード	tcpcli1.c
Tmaxノード	svr.c

4.1.1. 環境ファイル

< tmax.m >

```
*DOMAIN
res SHMKEY = 88000,
MINCLH = 1,
MAXCLH = 1,
TPORTNO = 8888

*NODE
node1TMAXDIR="/home/tmax",
APPDIR="/home/tmax/appbin"

*SVRGROUP
svg1NODENAME = node1

*SERVER
tcpgwlsn SVGNAME = svg1, MIN=1, MAX=1, SVRTYPE = CUSTOM_GATEWAY, RESTART = N,
        CLOPT="-- -P 9777 -N 3 -k 91000"
tcpgwhdr1 SVNAME = svg1, MIN=3, MAX=3, SCHEDULE=RR, SVRTYPE=CUSTOM_GATEWAY, CPC=10,
```

```

        TARGET = tcpgwhdr,
        CLOPT="-- -P 9777 -N 10 -s -L tcpgwlsn"
svr      SVGNAME = svg1

*SERVICE
TESTSVC  SVRNAME = svr

```

4.1.2. TCPGWTHR

<usermain.c >

```

#ifdef _WIN32
#include <winsock2.h>
#include <windows.h>
#include <io.h>
#else
#include <pthread.h>
#endif
#include <tcphdr.h>

char  sndbuf[4096];
char  rcvbuf[4096];
long  sndlen, rcvlen;
int    errflag;
extern int _portno;

int user_thrmain(WORKTHRINFO *winfo, int server)
{
    if (server)
        server_process(winfo);
    else
        client_process(winfo);

    return 1;
}

int server_process(WORKTHRINFO *winfo)
{
    int  n, len, flags;
    char tmp[10];

    memset(tmp, 0x00, sizeof(tmp));
    while (1) {
        n = tcpgw_select(winfo, 0, 0);
        if (n < 0)

```

```

        return -1;

    printf("tcpgw_select: n = %d\n", n);

    /* request ffrom tmax service & client */
    switch(n) {
    case WTHR_TMAX_REQUEST:
        len = tcpgw_get_svcddata(wininfo, &rcvbuf[4], &errflag, &flags);
        if (len < 0) {
            if (len == WTHR_CLIENT_CLOSE)
                return -1;

            printf("service data read failed\n");
            return -1;
        }
        printf("TMAX_REQUEST: length = %d\n", len);

        /* no reply */
        if (flags)
            n = tcpgw_tpreply(wininfo, rcvbuf, n, 0);

        sprintf(tmp, "%04d", len);
        memcpy(rcvbuf, tmp, 4);
        len += 4;

        n = tcpgw_write(wininfo->fd, rcvbuf, len);
        if (n < 0) {
            printf("remote client closed\n");
            return -1;
        }
        printf("TMAX_REQUEST: remote write ok [%d]\n", n);

        n = tcpgw_read(wininfo->fd, tmp, 4, 0, 0);
        if (n <= 0) {
            printf("remote client closed\n");
            return -1;
        }
        len = atoi(tmp);
        if (len <= 0)
            break;

        printf("TMAX_REQUEST: read length = %d\n", len);
        n = tcpgw_read(wininfo->fd, rcvbuf, len, 0, 0);
        if (n <= 0) {
            printf("remote client closed\n");
            return -1;
        }
    }

```

```

        if (flags) {
            printf("USER_REQUEST: service call length = %d\n", n);
            flags = 1;
            n = tcpgw_tpcall(winfo, "TESTSVC", rcvbuf, n, rcvbuf, &rcvlen);
        }
        else {
            printf("TMAX_REQUEST: tpcall reply length = %d\n", n);
            n = tcpgw_tpreply(winfo, rcvbuf, n, 0);
        }
        if (n < 0) {
            printf("tmax down\n");
            return -1;
        }
        break;

case WTHR_USER_REQUEST:
    n = tcpgw_read(winfo->fd, tmp, 4, 0, 0);
    if (n <= 0) {
        printf("remote client closed\n");
        return -1;
    }
    len = atoi(tmp);
    if (len <= 0)
        break;

    printf("USER_REQUEST: read length = %d\n", len);
    n = tcpgw_read(winfo->fd, sndbuf, len, 0, 0);
    if (n <= 0) {
        printf("remote client closed\n");
        return -1;
    }

    printf("USER_REQUEST: service call length = %d\n", n);
    n = tcpgw_tpcall(winfo, "TESTSVC", sndbuf, n, rcvbuf, &rcvlen);
    if (n < 0) {
        printf("service failed: [%d]\n", n);
    }

    sprintf(tmp, "%04d", rcvlen);
    memcpy(sndbuf, tmp, 4);
    memcpy(&sndbuf[4], rcvbuf, rcvlen);
    len = rcvlen + 4;

    printf("USER_REQUEST: service reply length = %d\n", len);
    n = tcpgw_write(winfo->fd, sndbuf, len);
    if (n < 0) {

```

```

        printf("remote client closed\n");
        return -1;
    }
    break;

    case WTHR_SELECT_TIMEOUT:
        /* timeout process */
        break;
    }
}
return 1;
}

int client_process(WORKTHRINFO *winfo)
{
    int    fd, portno;
    char   ipaddr[20];

    portno = tcpgw_getaddr_from_winfo(winfo, ipaddr);
    /* socket connect */
    while (1) {
        fd = tcpgw_network_connect(ipaddr, portno, 0);
        if (fd > 0)
            break;

        printf("remote connect failed\n");
#ifdef _WIN32
        Sleep(10000);
#else
        sleep(10);
#endif
        continue;
    }

    winfo->fd = fd; /* must save */
    server_process(winfo);

    return 1;
}

```

4.1.3. リモート・ノード

<tcpcli1.c >

```

#include <stdio.h>
#include <stdlib.h>

```

```

#include <string.h>

#ifdef _WIN32
#include <winsock2.h>
#include <windows.h>
#include <io.h>
#else
#include <unistd.h>
#include <netdb.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#endif

#ifdef _WIN32
#define GW_ADDR "host"
#else
#define GW_ADDR "10.10.10.10"
#endif

#define GW_PORT 9777
#define NUM_LOOP 1
#define MAX_MSG 496

#if (defined(_SOCK1) || defined(_SOCK11))
#define _LOBYTE 1
#define _HIBYTE 1
#else
#define _LOBYTE 2
#define _HIBYTE 0
#endif

#ifdef _WIN32
int winsock_init(void)
{
    WORD wVersionRequested;
    WSADATA wsaData;
    int err;

    wVersionRequested = MAKEWORD(_LOBYTE, _HIBYTE);
    err = WSStartup(wVersionRequested, &wsaData);
    if (err != 0) {
        /* Tell the user that we couldn't find a usable */
        /* WinSock DLL.                                     */
        printf("0060 Winsock startup error\n");
        return -1;
    }
}

```

```

}

/* Confirm that the WinSock DLL supports 2.0.*/
/* Note that if the DLL supports versions greater */
/* than 2.0 in addition to 2.0, it will still return */
/* 2.0 in wVersion since that is the version we */
/* requested. */
if (LOBYTE(wsaData.wVersion) != _LOBYTE ||
    HIBYTE(wsaData.wVersion) != _HIBYTE) {
    /* Tell the user that we couldn't find a usable */
    /* WinSock DLL. */
    WSACleanup();
    printf("0061 Winsock version check error\n");
    return -1;
}
/* The WinSock DLL is acceptable. Proceed. */

return 1;
}
#endif

int _network_connect(char *host, int port)
{
    struct sockaddr_in serv_addr;
    unsigned int inaddr;
    struct hostent *hp;
    int i, fd;

    memset((char *) &serv_addr, 0, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(port);

    if ((inaddr = (unsigned int) inet_addr(host)) != -1) {
        memcpy((char *) &serv_addr.sin_addr, (char *) &inaddr,
            sizeof(struct in_addr));
    } else {
        if ((hp = gethostbyname(host)) == NULL) {
            printf("COM3412: host name error: %s ", host);
            return(-1);
        }
        memcpy((char *) &serv_addr.sin_addr, hp->h_addr,
            hp->h_length);
    }

    if ((fd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        printf("COM3413: can't open stream socket");
        return -1;
    }
}

```

```

    }

    if (connect(fd, (struct sockaddr *) &serv_addr,
                sizeof(serv_addr)) >= 0)
        return fd;

    close(fd);

    return -1;
}

int main(int argc, char *argv[])
{
    char gw_addr[256], tmp[10];
    int gw_port, fd, i, n, len, num_loop;
    char data[MAX_MSG];

    strcpy(gw_addr, GW_ADDR);
    gw_port = GW_PORT;
    num_loop = NUM_LOOP;
    if (argc == 2) {
        num_loop = atoi(argv[1]);
    } else if (argc == 3) {
        gw_port = atoi(argv[1]);
        num_loop = atoi(argv[2]);
    } else if (argc >= 4) {
        strcpy(gw_addr, argv[1]);
        gw_port = atoi(argv[2]);
        num_loop = atoi(argv[3]);
    }

#ifdef _WIN32
    winsock_init();
#endif

    fd = _network_connect(gw_addr, gw_port);
    if (fd < 0) {
        printf("Connect to (%s:%d) fail \n", gw_addr, gw_port);
        return -1;
    }

    sleep(5);
    memset(data, 0x00, MAX_MSG);
    for (i=0; i<num_loop; i++) {
        sprintf(&data[4], "Msg(%d) produced by PID(%d)", i, getpid());
        len = strlen(&data[4]);
        sprintf(tmp, "%04d", len);
    }
}

```



```

memcpy(data, tmp, 4);

len += 4;
n = send(fd, data, len, 0);
if (n != len) {
    printf("Sent only %d / %d bytes\n", n, len);
    return -1;
}

len = 4;
n = recv(fd, data, len, 0);
if (n != len) {
printf("Recv error %d\n", n);
    return -1;
}

memcpy(tmp, data, 4);
tmp[4] = 0x00;
len = atoi(tmp);
if (len <= 0) {
    printf("Pid (%d) received %d bytes\n", getpid(), n);
    continue;
}

n = recv(fd, data, len, 0);
if (n != len) {
    printf("Recv error %d\n", n);
    return -1;
}
printf("Pid (%d) received %d bytes\n",
    getpid(), n);
sleep(5);
}

return 1;
}

```

4.1.4. Tmaxノード

< svr.c >

```

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

TESTSVC(TPSVCINFO *msg)

```

```

{
    inti;

    printf("TESTSVC service is started!\n");
    sleep(1);
    printf("OUTPUT: data=%.*s\n", msg->len, msg->data);
    printf("TESTSVC service is stoped!\n");
    tpreturn(TPSUCCESS,0,(char *)msg->data, msg->len, 0);
}

```

4.2. サーバー・モードのインバウンド・サービス呼び出し

サーバー・モードのTCPGWTHRを利用して、Tmaxからリモートのサービスを要求する例です。

プログラムの構成は下記のとおりです。

区分	ファイル名
環境ファイル	tmax.m, tcpgwthr.cfg
TCPGWTHR	usermain.c(ファイルの例は 「4.1. サーバー・モードのアウトバウンド・サービス呼び出し」 で説明したため、本節では省略します)
リモート・ノード	tcpcli2.c
Tmaxノード	toupper.c

4.2.1. 環境ファイル

< tmax.m >

```

*DOMAIN
Res SHMKEY = 88000,
MINCLH = 1,
MAXCLH = 1,
TPORTNO = 8888

*NODE
node1TMAXDIR="/home/tmax",
APPDIR="/home/tmax/appbin"

*SVRGROUP
svg1NODENAME = node1

*SERVER
tcpgwlsn SVGNAME = svg1, MIN=1, MAX=1, SVRTYPE = CUSTOM_GATEWAY, RESTART = N,

```

```

        CLOPT="-- -P 9777 -N 3 -k 91000 -F /home/tmax/appbin/tcpgwthr.cfg"

tcpgwhdr1 SVNAME = svgl, MIN=3, MAX=3, SCHEDULE=RR,SVRTYPE=CUSTOM_GATEWAY, CPC=
10,
        TARGET    = tcpgwhdr,
        CLOPT="-- -P 9777 -N 10 -s -L tcpgwlsn"

*SERVICE
svcgw      SVRNAME = tcpgwhdr1

```

< tcpgwthr.cfg >

```

# Client IP   Server Port Client ID
10.10.10.10   9777       SVR0001

```

4.2.2. リモート・ノード

< tcpcli2.c >

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#ifdef _WIN32
#include <winsock2.h>
#include <windows.h>
#include <io.h>
#else
#include <unistd.h>
#include <netdb.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#endif

#ifdef _WIN32
#define GW_ADDR    "host"
#else
#define GW_ADDR    "10.10.10.10"
#endif

#define GW_PORT    9777
#define NUM_LOOP  1

```

```

#define MAX_MSG    496

#if (defined(_SOCK1) || defined(_SOCK11))
#define _LOBYTE    1
#define _HIBYTE    1
#else
#define _LOBYTE    2
#define _HIBYTE    0
#endif

#ifdef _WIN32
int winsock_init(void)
{
    WORD wVersionRequested;
    WSADATA wsaData;
    int err;

    wVersionRequested = MAKEWORD(_LOBYTE, _HIBYTE);
    err = WSStartup(wVersionRequested, &wsaData);
    if (err != 0) {
        /* Tell the user that we couldn't find a usable */
        /* WinSock DLL.                                   */
        printf("0060 Winsock startup error\n");
        return -1;
    }

    /* Confirm that the WinSock DLL supports 2.0.*/
    /* Note that if the DLL supports versions greater */
    /* than 2.0 in addition to 2.0, it will still return */
    /* 2.0 in wVersion since that is the version we */
    /* requested.                                       */
    if (LOBYTE(wsaData.wVersion) != _LOBYTE ||
        HIBYTE(wsaData.wVersion) != _HIBYTE) {
        /* Tell the user that we couldn't find a usable */
        /* WinSock DLL.                                   */
        WSACleanup();
        printf("0061 Winsock version check error\n");
        return -1;
    }
    /* The WinSock DLL is acceptable. Proceed. */

    return 1;
}
#endif

int _network_connect(char *host, int port)
{

```

```

struct sockaddr_in serv_addr;
unsigned int inaddr;
struct hostent *hp;
int i, fd;

memset((char *) &serv_addr, 0, sizeof(serv_addr));
serv_addr.sin_family = AF_INET;
serv_addr.sin_port = htons(port);

if ((inaddr = (unsigned int) inet_addr(host)) != -1) {
    memcpy((char *) &serv_addr.sin_addr, (char *) &inaddr,
        sizeof(struct in_addr));
} else {
    if ((hp = gethostbyname(host)) == NULL) {
        printf("COM3412: host name error: %s ", host);
        return(-1);
    }
    memcpy((char *) &serv_addr.sin_addr, hp->h_addr,
        hp->h_length);
}

if ((fd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
    printf("COM3413: can't open stream socket");
    return -1;
}

if (connect(fd, (struct sockaddr *) &serv_addr,
    sizeof(serv_addr)) >= 0)
    return fd;

close(fd);

return -1;
}

int main(int argc, char *argv[])
{
    char gw_addr[256], tmp[10];
    int gw_port, fd, i, n, len, num_loop;
    char data[MAX_MSG];

    strcpy(gw_addr, GW_ADDR);
    gw_port = GW_PORT;
    if (argc == 2) {
        gw_port = atoi(argv[1]);
    }

```

```

#ifdef _WIN32
    winsock_init();
#endif

fd = _network_connect(gw_addr, gw_port);
if (fd < 0) {
    printf("Connect to (%s:%d) fail \n", gw_addr, gw_port);
    return -1;
}

memset(data, 0x00, MAX_MSG);
while (1) {
    len = 4;
    n = recv(fd, data, len, 0);
    if (n != len) {
        printf("Recv error %d\n", n);
        return -1;
    }

    memcpy(tmp, data, 4);
    tmp[4] = 0x00;
    len = atoi(tmp);
    if (len <= 0) {
        printf("Pid (%d) received %d bytes\n", getpid(), n);
        continue;
    }

    n = recv(fd, &data[4], len, 0);
    if (n != len) {
        printf("Recv error %d\n", n);
        return -1;
    }
    printf("Pid (%d) received %d bytes\n", getpid(), n);

    len = n + 4;
    for (i = 4; i < len; i++)
        data[i] = toupper(data[i]);

    n = send(fd, data, len, 0);
    if (n != len) {
        printf("Sent only %d / %d bytes\n", n, len);
        return -1;
    }
    printf("Pid (%d) sended %d bytes\n", getpid(), len-4);
}

```

```
    return 1;
}
```

4.2.3. Tmaxノード

< toupper.c >

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

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

    strcpy(gwinfo.svc, "SVR0001");
    if (argc != 2) {
        printf("Usage: toupper string\n");
        exit(1);
    }

    if ( (ret = tmaxreadenv( "tmax.env", "TMAX" )) == -1 ) {
        printf( "tmax read env failed\n" );
        exit(1);
    }

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

    if ((sndbuf = (char *)tpalloc("CARRAY", NULL, 0)) == NULL) {
        printf("sndbuf alloc failed !\n");
        tpend();
        exit(1);
    }

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

```

    exit(1);
}

memcpy(sndbuf, &gwinfo, TPGWINFO_SIZE);
strcpy(sndbuf+TPGWINFO_SIZE, argv[1]);
len = strlen(argv[1]);
len += TPGWINFO_SIZE;

if(tpcall("svcgw", sndbuf, len, &rcvbuf, &rcvlen, 0)==-1){
    printf("Can't send request to service svcgw - %d\n", tperrno);
    tpfree((char *)sndbuf);
    tpfree((char *)rcvbuf);
    tpend();
    exit(1);
}

printf("send data: %s\n", sndbuf+TPGWINFO_SIZE);
printf("recv data: %s\n", rcvbuf);

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

```

4.3. クライアント・モードのインバウンド・サービス呼び出し

クライアント・モードのTCPGWTHRを利用して、Tmaxからリモートのサービスを要求する例です。

プログラムの構成は下記のとおりです。

区分	ファイル名
環境ファイル	tmax.m, tcpgwthr.cfg
TCPGWTHR	usermain.c(ファイルの例は「 4.1. サーバー・モードのアウトバウンド・サービス呼び出し 」で説明したため、本節では省略します)
リモート・ノード	tcpsvr2.c
Tmaxノード	toupper.c(ファイルの例は「 4.2. サーバー・モードのインバウンド・サービス呼び出し 」で説明したため、本節では省略します)

4.3.1. 環境ファイル

< tmax.m >

```
*DOMAIN
res          SHMKEY = 88000,
             MINCLH = 1,
             MAXCLH = 1,
             TPORTNO = 8888

*NODE
node1        TMAXDIR = "/home/tmax",
             APPDIR = "/home/tmax/appbin"

*SVRGROUP
svg1         NODENAME = node1

*SERVER
clihdrshm    SVGNAME = svg1, MIN = 1, MAX = 1,
             SVRTYPE = CUSTOM_GATEWAY,
             CLOPT = "-- -N 10 -k 91000 -F /home/tmax/appbin/tcpgwthr.cfg"
tcpgwhdr1    SVGNAME svg1, MIN = 10, MAX = 10, SVRTYPE = CUSTOM_GATEWAY,
             CPC = 9, SCHEDULE = RR, TARGET = tcpgwhdr,
             CLOPT = "-- -k 91000 -L clihdrshm -F /home/tmax/appbin/tcpgwthr.cfg"

*SERVICE
svcgw        SVRNAME = tcpgwhdr1
```

4.3.2. リモート・ノード

< tcpsvr2.c >

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>

#ifdef _WIN32
#include <winsock2.h>
#include <windows.h>
#include <io.h>
#else
#include <unistd.h>
#include <netdb.h>
```

```

#include <sys/types.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#endif

#define GW_PORT    9777
#define NUM_LOOP   1
#define MAX_MSG    496

#if (defined(_SOCK1) || defined(_SOCK11))
#define _LOBYTE    1
#define _HIBYTE    1
#else
#define _LOBYTE    2
#define _HIBYTE    0
#endif
#define max(a,b) ((a) > (b) ? (a) : (b))

int    listen_fd = -1;
int    work_fd;
fd_set readfds;
int    maxfd = 0;

#ifdef _WIN32
int winsock_init(void)
{
    WORD wVersionRequested;
    WSADATA wsaData;
    int err;

    wVersionRequested = MAKEWORD(_LOBYTE, _HIBYTE);
    err = WSAStartup(wVersionRequested, &wsaData);
    if (err != 0) {
        /* Tell the user that we couldn't find a usable */
        /* WinSock DLL.                                  */
        printf("0060 Winsock startup error\n");
        return -1;
    }

    /* Confirm that the WinSock DLL supports 2.0.*/
    /* Note that if the DLL supports versions greater */
    /* than 2.0 in addition to 2.0, it will still return */
    /* 2.0 in wVersion since that is the version we */
    /* requested.                                         */
    if (LOBYTE(wsaData.wVersion) != _LOBYTE ||

```

```

        HIBYTE(wsaData.wVersion) != _HIBYTE) {
/* Tell the user that we couldn't find a usable */
/* WinSock DLL.                                     */
WSACleanup();
printf("0061 Winsock version check error\n");
return -1;
}
/* The WinSock DLL is acceptable. Proceed. */

return 1;
}
#endif

int network_listen(int portno)
{
    int fd;
    struct sockaddr_in serv_addr;
    int on;

    if ((fd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
        return(-1);

    on = 1;
    if (setsockopt(fd, SOL_SOCKET, SO_REUSEADDR, (char *) &on,
        sizeof(on)) < 0) {
        printf("0062 setsockopt error: SO_REUSEADDR");
    }

    memset((char *) &serv_addr, 0, sizeof(serv_addr));
    serv_addr.sin_family      = AF_INET;
    serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    serv_addr.sin_port        = htons((unsigned short)portno);

    if (bind(fd, (struct sockaddr *) &serv_addr, sizeof(serv_addr)) < 0) {
        close(fd);
        return(-2);
    }

    if (listen(fd, 50) < 0) {
        close(fd);
        return(-3);
    }

    return(fd);
}

int network_accept(int listenfd)

```

```

{
    socklen_t len;
    int      fd, on;
    struct sockaddr_in cli_addr;

    len = sizeof(cli_addr);
    fd = accept(listenfd, (struct sockaddr *) &cli_addr, &len);
    if (fd < 0)
        return(-1); /* often errno=EINTR, if signal caught */

    return(fd);
}

int main(int argc, char *argv[])
{
    char gw_addr[256];
    int gw_port, fd, i, n, len, num_loop;
    char data[MAX_MSG], *sndbuf;

    gw_port = GW_PORT;
    if (argc == 2) {
        gw_port = atoi(argv[1]);
    }

#ifdef _WIN32
    winsock_init();
#endif

    listen_fd = network_listen(gw_port);
    if (listen_fd < 0) {
        printf("Connect to (%d) fail \n", gw_port);
        return -1;
    }

    FD_ZERO(&readfds);
    FD_SET(listen_fd, &readfds);
    maxfd = listen_fd;

    while (1) {
        errno = 0;
        n = select(maxfd + 1, &readfds, NULL, NULL, NULL);
        if (n < 0) {
            if (errno == EINTR) { /* signal is caught */
                continue;
            } else {
                printf("0080 select error");
                return -1;
            }
        }
    }

```

```

    }
}

if (FD_ISSET(listen_fd, &readfds)) {
    if ((fd = network_accept(listen_fd)) < 0) {
        printf("0043 socket accept error");
        continue;
    }
    else {
        work_fd = fd;
        FD_SET(fd, &readfds);
        maxfd = max(maxfd, fd);
    }
}

if (FD_ISSET(work_fd, &readfds)) {
    service_process(work_fd);
    FD_CLR(work_fd, &readfds);
    close(work_fd);
}

return 1;
}

int service_process(int fd)
{
    int    i, n, len;
    char   tmp[10];
    char   data[MAX_MSG];

    len = 4;
    n = recv(fd, data, len, 0);
    if (n != len) {
        printf("Recv error %d\n", n);
        return -1;
    }

    memcpy(tmp, data, 4);
    tmp[4] = 0x00;
    len = atoi(tmp);
    if (len <= 0) {
        printf("Pid (%d) received %d bytes\n", getpid(), n);
        return 1;
    }

    n = recv(fd, &data[4], len, 0);

```

```

if (n != len) {
    printf("Recv error %d\n", n);
    return 1;
}
printf("Pid (%d) received %d bytes\n", getpid(), n);

len = n + 4;
for (i = 4; i < len; i++)
    data[i] = toupper(data[i]);

n = send(fd, data, len, 0);
if (n != len) {
    printf("Sent only %d / %d bytes\n", n, len);
    return 1;
}
printf("Pid (%d) sended %d bytes\n", getpid(), len-4);

return 1;
}

```

付録 A. ヘッダー・ファイル

A.1. TCPHDR.Hファイル

```
#ifndef _TCPHDR_H_
#define _TCPHDR_H_

#ifndef _WIN32
#define __cdecl
#endif

#define CLIENT_ID_SIZE      15

/*      THREAD status */
#define WTHR_START          2
#define WTHR_READY          3
#define WTHR_RUNNING        4

/*      SELECT status */
#define WTHR_SELECT_ERROR   -1
#define WTHR_SELECT_TIMEOUT 1
#define WTHR_TMAX_REQUEST   2
#define WTHR_USER_REQUEST   3

/*      READ  status */
#define WTHR_CLIENT_CLOSE    0
#define WTHR_READ_ERROR      -1
#define WTHR_READ_TIMEOUT    -2

/*      WRITE  status */
#define WTHR_WRITE_ERROR     -1

struct workwthrinfo {
    int    idx;
    int    status;
    int    fd;
    int    portno;
    unsigned int ipaddr;
    char    clientid[CLIENT_ID_SIZE+1];
};
typedef struct workwthrinfo WORKTHRINFO;
```

```

#define SHMHEAD_SIZE  sizeof(struct shmhead)
struct shmhead {
    char  shmdate[8];
    int   shmsize;
    int   handler_num;
    int   maxnum;
    int   resvd;
};
typedef struct shmhead SHMHEAD;

#define HANDLERINFO_SIZE  sizeof(struct handler_info)
struct handler_info {
    int   status;
    int   fd;
    int   svrn;
    int   pid;
    int   portno;
    int   num;
};
typedef struct handler_info HANDLERINFO;

#define CLIENTINFO_SIZE  sizeof(struct clientinfo)
struct clientinfo {
    int   status;
    int   thridx;
    int   svridx;
    int   rcount;
    int   scount;
    int   pid;
    int   portno;
    unsigned int ipaddr;
    int   utime;
    int   resvd;
    char  clientid[CLIENT_ID_SIZE+1];
};
typedef struct clientinfo CLIENTINFO;

#ifdef __cplusplus
extern "C" {
#endif

/* ----- function prototypes ----- */
int __cdecl tcpgw_tpcall(WORKTHRINFO *wthrinfo, char *svc, char *ptr,
                        long alen, char *optr, long *olen);
int __cdecl tcpgw_tpacall(WORKTHRINFO *wthrinfo, char *svc, char *ptr, long alen);
int __cdecl tcpgw_tpreply(WORKTHRINFO *wthrinfo, char *ptr, long alen, int err);
int __cdecl tcpgw_select(WORKTHRINFO *wthrinfo, int sec, int usec);

```



```
int __cdecl tcpgw_get_svcddata(WORKTHRINFO *wthrinfo, char *ptr, int *err,
                               int *flags);
int __cdecl tcpgw_network_connect(char *host, int port, int sec);
int __cdecl tcpgw_read(int fd, char *ptr, int nbytes, int sec, int usec);
int __cdecl tcpgw_write(int fd, char *ptr, int nbytes);
int __cdecl tcpgw_portno_ipaddr(char *ipaddr);
int __cdecl tcpgw_getaddr_from_winfo(WORKTHRINFO *winfo, char *ipaddr);
int __cdecl tcpgw_client_id(WORKTHRINFO *winfo, char *id);

#ifdef __cplusplus
}
#endif

#endif /* _TCPHDR_H_ */
```


索引

A

API関数

- tcpgw_client_id(), 21
- tcpgw_get_svcddata(), 18
- tcpgw_getaddr_from_wininfo(), 22
- tcpgw_network_connect(), 19
- tcpgw_portno_ipaddr(), 21
- tcpgw_read(), 20
- tcpgw_select(), 18
- tcpgw_tpacall(), 17
- tcpgw_tpcall(), 16
- tcpgw_tpreply(), 17
- tcpgw_write(), 20

C

Clishmプロセス, 2

CLOPT項目

- Clishmオプション, 12
- ハンドラー・オプション, 11
- リスナー・オプション, 10

T

TCP/IPスレッド・ゲートウェイ, 1

- tcpgw_client_id(), 21
- tcpgw_get_svcddata(), 18
- tcpgw_getaddr_from_wininfo(), 22
- tcpgw_network_connect(), 19
- tcpgw_portno_ipaddr(), 21
- tcpgw_read(), 20
- tcpgw_select(), 18
- tcpgw_tpacall(), 17
- tcpgw_tpcall(), 16
- tcpgw_tpreply(), 17
- tcpgw_write(), 20
- TCPGWTHR, 1
- TCPGWTHRの動作構造, 1

TCPGWTHR環境ファイル, 13

アドレス情報の環境ファイル, 13

Tmax環境ファイル, 10

Tmax環境構成

CLOPT項目(TCPGWTHRオプション), 10

Tmax環境構成(CLOPT Clishm)

- [-F], 12
- [-k], 12
- [-N], 12
- [-w], 12

Tmax環境構成(CLOPTハンドラー)

- [-F], 11
- [-f], 12
- [-i], 12
- [-k], 11
- [-N], 11
- [-P], 11
- [-r], 11
- [-s], 11
- [-w], 12
- [-X CLIENT_IPV6], 12

Tmax環境構成(CLOPT項目-リスナー)

- [-F], 10
- [-k], 10
- [-N], 11
- [-P], 10
- [-w], 11
- [-X SERVER_IPV6], 11

か

クライアント・モード, 5

シングル・サーバー接続, 5

マルチサーバー接続, 6

クライアント・モードのインバウンド・サービス呼び出しの例, 38

コールバック関数, 15

user_thrmain(), 15

さ

サーバー・モード

クライアント接続(dedicated方式), 4

クライアント接続(shared方式), 3

サーバー・モードのアウトバインド・サービス呼び出しの
例, 23

サーバー・モードのインバウンド・サービス呼び出しの例,
32

は

ハンドラー・プロセス, 2

ら

例

クライアント・モードのインバウンド・サービス呼び出し,
38

サーバー・モードのアウトバウンド・サービス呼び出し,
23

サーバー・モードのインバウンド・サービス呼び出し,
32

リスナー・プロセス, 2