

# Tmax ゲートウェイガイド (X.25)

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 (X.25)

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
<b>第1章 紹介 .....</b>	<b>1</b>
1.1. 概要 .....	1
1.2. サービスのタイプ .....	2
1.2.1. 同期型X25GW .....	3
1.2.2. 非同期型X25GW .....	6
1.3. ゲートウェイのその他の機能 .....	8
1.3.1. ゲートウェイ・ヘッダー .....	8
1.3.2. サービス名の検索順 .....	8
1.3.3. ユーザーが任意でチャンネルを指定 .....	9
1.3.4. リセット処理方法 .....	9
<b>第2章 環境設定 .....</b>	<b>11</b>
2.1. 概要 .....	11
2.2. Tmax環境構成 .....	11
2.2.1. サービス・ブロック型X25GW .....	14
2.2.2. サービス非ブロック型X25GW .....	15
2.2.3. リモート同期型と非同期型X25GW .....	16
2.3. X25GWチャンネル環境設定 .....	17
2.3.1. チャンネル環境ファイル .....	17
2.3.2. マシン別のチャンネル環境ファイル .....	19
2.4. ユーザー・ヘッダー環境設定 .....	19
<b>第3章 ユーザー・プログラムと関数 .....</b>	<b>21</b>
3.1. 概要 .....	21
3.2. custom.h .....	21
3.3. custom.c .....	22
3.3.1. init_remote_info .....	23
3.3.2. remote_connected .....	23
3.3.3. remote_closed .....	24
3.3.4. get_msg_info .....	24
3.3.5. get_channel_num .....	25
3.3.6. put_msg_info .....	25
3.3.7. get_service_name3 .....	26
3.3.8. prepare_shutdown .....	26
3.3.9. inmsg_recovery .....	27
3.3.10. outmsg_recovery .....	27
3.4. register.c .....	28
<b>第4章 例 .....</b>	<b>31</b>
4.1. アウトバウンドX25GW .....	31
4.1.1. 環境ファイル .....	31

4.1.2.	X25GW .....	32
4.1.3.	サーバー .....	36
4.2.	同期型インバウンドX25GW .....	36
4.2.1.	環境ファイル .....	37
4.2.2.	X25GW .....	38
4.2.3.	クライアント .....	42
4.3.	サービス非ブロック型X25GW .....	43
4.3.1.	環境ファイル .....	44
4.3.2.	X25GW .....	45
4.3.3.	サーバー .....	48
4.3.4.	クライアント .....	50
<b>付録 A.</b>	<b>X25GWのエラーコード .....</b>	<b>53</b>
<b>索引</b>	<b>.....</b>	<b>55</b>

## 図目次

[図 1.1]	X25GWの動作構造 .....	1
[図 1.2]	同期/非同期X25GWの動作構造 .....	2
[図 1.3]	ブロック型同期X25GWの動作構造 .....	3
[図 1.4]	非ブロック型の同期X25GWの動作構造 .....	5
[図 1.5]	リモート同期型X25GWの動作構造 .....	6
[図 1.6]	Tmax要求の非同期型X25GWの動作構造 .....	7
[図 1.7]	リモート要求の非同期型X25GWの動作構造 .....	7
[図 4.1]	アウトバウンドX25GWの動作構造 .....	31
[図 4.2]	同期型インバウンドX25GWの動作構造 .....	37
[図 4.3]	非ブロック型X25GWの動作構造 .....	43





# このガイドについて

## 対象読者

本書は、Tmax<sup>®</sup>(以下、Tmax)が提供するX.25ゲートウェイを使用する開発者を対象としています。同書では、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X.25ゲートウェイについて説明します。

## 前提知識

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

本書を理解するためには、以下の事項についての知識が必要です。

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

## 制限事項

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

---

### 参考

Tmaxシステムの開発についての基本的な内容は、『Tmax 運用ガイド』および『Tmax アプリケーション開発ガイド』を、Tmaxが提供するコマンドとC APIについては、『Tmax リファレンスガイド』を参照してください。

---

# 本書の構成

本書は、計4章と付録で構成されています。

各章の主な内容は以下のとおりです。

- 第1章: 紹介

Tmaxが提供するX.25ゲートウェイの概要およびサービスのタイプについて説明します。

- 第2章: 環境設定

Tmaxが提供するX.25ゲートウェイの環境構成および設定方法について説明します。

- 第3章: ユーザー・プログラムと関数

X.25ゲートウェイを使用してユーザー・プログラムを作成するための方法とAPIについて説明します。

- 第4章: 例

サービス・タイプの例について説明します。

- 付録A: X25GWのエラーコード

X25ゲートウェイのエラーコードについて説明します。

## 表記上の規則

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

## 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

## 関連文書

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

## お問合せ先

### Korea

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

### USA

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

### Japan

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

## China

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

## Brazil

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

## Russia

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

## Singapore

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

## United Kingdom

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

## Canada

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



## Australia

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

## India

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

## Turkey

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



# 第1章 紹介

本章では、Tmaxが提供するX.25ゲートウェイの概要およびサービスのタイプについて説明します。

## 1.1. 概要

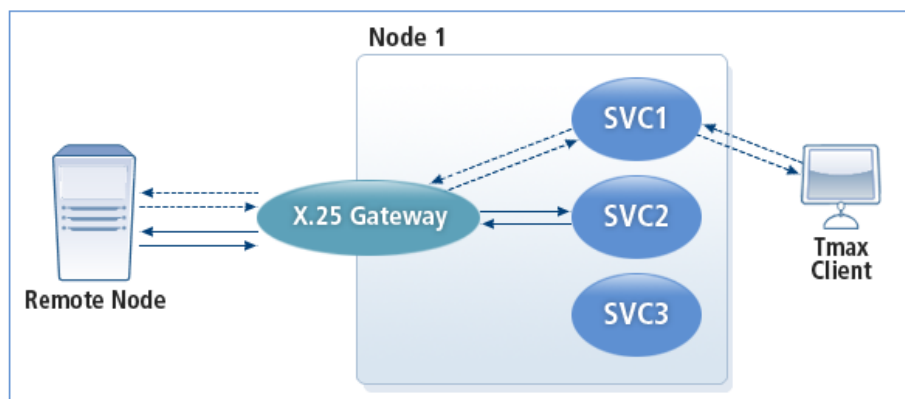
X.25ゲートウェイ(以下、X25GW)は、Tmaxサーバーと非Tmaxサーバー(以下、リモートノード)間のインターフェースを担当する、Tmaxが提供するゲートウェイです。X25GWはTmaxサーバーの一種として、X.25で接続されているUNIX/Windowsサーバーとのゲートウェイの役割をします。X25GWは、リモートノードからのメッセージを該当のサービスにtpacall()し、サービス結果は最初に要求したリモートノードに転送します。一方、TmaxサーバーからX25GWに、tpcall()またはその他の方式でサービスを要求する場合、X25GWはリモートノードに要求メッセージを転送し、応答が受信されたら自身を呼び出したサービスにtpreturn()します。

アタッチ(Attach)、デタッチ(Detach)、メッセージの送受信など、他のシステムとX.25で接続するために必要な作業はすべてX.25ゲートウェイで処理するため、開発者は業務ロジックのみ作成します。

X25GWの動作方式は、Tmaxのサービスまたはクライアントからリモートノードにサービスを要求する方式と、リモートノードからTmaxのサービスを呼び出す方式で分けられます。

下記は、X25GWの動作構造です。

[図 1.1] X25GWの動作構造



- Tmaxからのサービス要求

[図 1.1]の点線で表示されている部分です。Tmaxクライアントまたはサービスからサービス要求を受け、リモートノードにサービスを要求することができます。このようなサービスを**インバウンド・サービス**といいます。

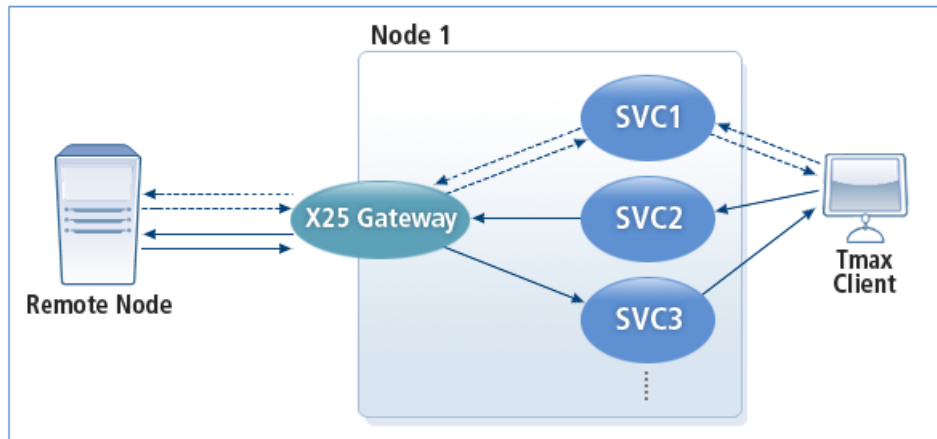
- リモートノードからのサービス要求

[図 1.1]の実線で表示されている部分です。X25GWはリモートノードからサービス要求を受け処理することができます。このようなサービスを**アウトバウンド・サービス**といいます。

アウトバウンドの場合は、リモートノードでTmaxのサービス名を利用してサービスを呼び出します。

下記は、同期/非同期方式でサービスを要求する場合のX25GWの動作構造です。

[図 1.2] 同期/非同期X25GWの動作構造



- 同期型呼び出し方式

[図 1.2]の点線で表示されている部分です。Tmaxクライアントまたはサービスで直接X25GWを呼び出し、応答が受信されるまで待機する通信方式です。

- 非同期型呼び出し方式

[図 1.2]の実線で表示されている部分です。TmaxクライアントがTmaxサービスを呼び出すと、そのサービスからX25GWに制御権が移り、当該サービスは他のサービス要求が受けられます。さらに、X25GWがリモートノードから応答を受けると、応答を処理するサービスに要求を渡す方式で動作します。

TmaxにインストールされるX25GWはTmaxサーバーの一種です。X25GWを使用するには、Tmax環境ファイルにサーバーとして登録する必要があります。一般的には、TCSまたはUCS用サーバー・ライブラリーを利用してサーバーを作成しますが、X25GWは外部との通信を担当するライブラリー(libx25gw.a、libx25gw.so)とユーザーが作成したプログラム(custom.c)をリンクしてサーバーを作成します。Tmax環境を設定し、プログラム(custom.c、custom.h)を作成する必要があるため、詳しい内容については、「[第2章 環境設定](#)」と「[第3章 ユーザー・プログラムと関数](#)」を参照してください。

## 1.2. サービスのタイプ

X25GWは、Tmaxが提供するライブラリー(libx25gw.a、libx25gw.so)と開発者が作成するcustom.c、custom.hをリンクして作成されます。このように作られたX25GWはリモートノードと通信し、Tmaxクライアントの要求をリモートノードに転送するか、リモートノードの要求をTmaxサービスで処理できるようにします。

X25GWは動作方式によって、同期型X25GWと非同期型X25GWで分けられます。

### 1.2.1. 同期型X25GW

同期型方式は、Tmaxのクライアントまたはサーバーでサービスを要求し、その応答がサービスを要求したクライアントまたはサーバーに受信される方式です。一方、リモートノードでサービスを要求する場合、X25GWはTmaxのサービスを要求し、その結果をサービスを要求したリモートノードに返すこともあります。前者の場合、リモートノードにサービスを要求したTmaxサービスのブロック有無によって、X25GWの動作方式が異なります。後者の場合は、リモートノードが処理結果を返すとき、要求したチャンネルに返す方式です。

Tmaxのクライアントまたはサーバーからリモートノードに同期型方式でサービスを要求する場合は、X25GWとリモートノード間にUID(Unique ID)を共有する必要があります。

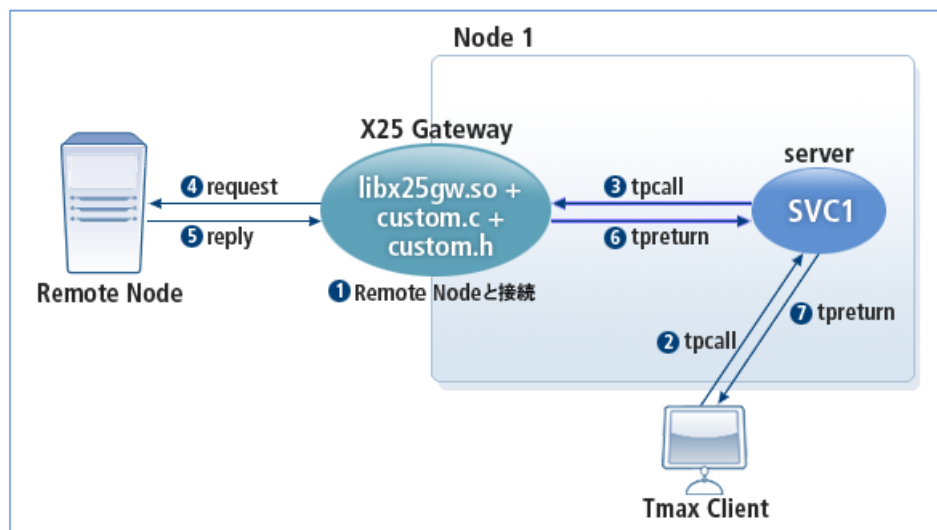
### サービス・ブロック型方式

サービス・ブロック型方式は、TmaxのサーバーまたはクライアントからX25GWにサービスを要求し、その結果が受信されるまで待機する、最も一般的な方式です。下記図のように、Tmaxクライアントの要求を受けたサービスがX25GWにサービスをtpcallすると、リモートノードが処理結果を送信するまでTmaxサービス(SVC1)がブロックされます。

X25GWを動作させると、Tmaxクライアントが呼び出したTmaxサービスは、X25GWを呼び出した後、結果を受けるまでブロックされます。tpacallした場合も、tpgetrplyで応答を受ける場所がブロックされるため、tpcall()と同様にブロックされます。外部通信は様々な障害(マシン、ネットワーク)に備え、多くのサーバーが必要となります。したがって、多くの要求を受けるためには、ブロックされる時間まで考慮して多数のサーバーを実行する必要があります。

下記は、同期型X25GWでのサービス・ブロック型方式の動作構造です。

[図 1.3] ブロック型同期X25GWの動作構造



1. X25GWとリモートノードが接続されている状態です。
2. TmaxクライアントはTmaxサービスをtpcallします。
3. Tmaxサービスではクライアントの要求を受け、X25GWにサービスをtpcallします。
4. X25GWは、接続されているリモートノードにサービスを要求します。
5. リモートノードから結果が受信されたらエラー有無を判断します。
6. リモートノードからの結果を、X25GWサービスを呼び出したTmaxサービスにtpreturnします。
7. 結果を受けたTmaxサービスは、Tmaxクライアントにtpreturnします。

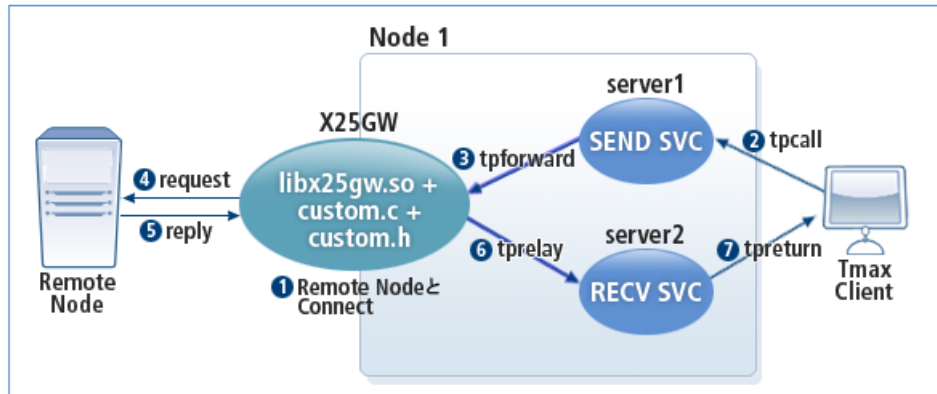
## サービス非ブロック型方式

サービス非ブロック方式は、Tmaxクライアントから直接X25GWを要求する方式では使用できません。X25GWにサービスを要求し、結果を受信するサーバーを間に設けて処理する方式です。X25GWの前に送信サービスと受信サービスをおき、Tmaxクライアントは送信サービスを呼び出し、送信サービスはX25GWにサービスを転送してサービスを終了します。X25GWはリモートノードにサービスを要求してその結果が受信されたら、当該結果を受信サービスに渡します。受信サービスは、X25GWから結果を受けてクライアントに渡したらサービスのサイクルが終了される方式です。結果的にクライアントはサービスを要求して結果を受ける同期方式ですが、サーバーは要求を渡して終了する非同期方式のように動作します。

非ブロック型方式でのX25GWは、ブロック型方式より少ないサーバーで多くの処理ができます。ブロック型方式はサービスがX25GWを呼び出してブロックされるため、同時に多くの処理をするには多いサーバー・プロセスを実行する必要があります。しかし、非ブロック型方式では自身の作業のみ処理してサービスを終了するため、少ないサーバー・プロセスで多くの作業が処理できます。したがって、外部通信を処理する場合は、非ブロック型方式がより効率的です。

下記は、同期型X25GWでのサービス非ブロック型方式の動作構造です。

【図 1.4】 非ブロック型の同期X25GWの動作構造



1. X25GWとリモートノードが接続されています。
2. TmaxクライアントはTmaxサービスをtpcallします。
3. Tmaxサービスではクライアントの要求を受け、X25GWサービスをtpforwardします。
4. X25GWは接続されているリモートノードにメッセージを転送します。
5. リモートノードから結果が受信されたら、エラー有無を判断します。
6. リモートノードからの結果をtprelayするサービスにtprelayします。
7. X25GWサービスから結果を受信したサービス(Tmaxクライアントからtprelayするサービスとして指定されたサービス)は、Tmaxクライアントにtpreturnします。

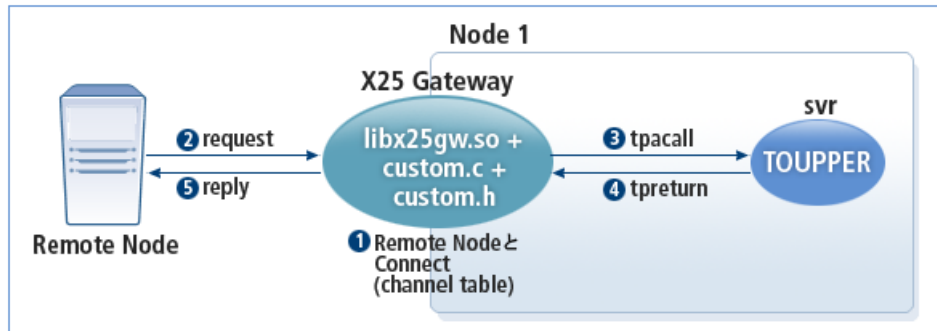
## リモート同期型呼び出し方式

リモート同期型呼び出し方式は、リモートノードからX25GWに先にサービスを要求する方式です。X25GWはリモートノードが要求したサービスを呼び出し、その結果を受けて当該サービスを要求したチャンネルに転送します。リモートノードはX25GWに対して、同時にTmax Configに定義されているMAXSACALL数を超過して呼び出すことはできません。

リモート同期型呼び出しは、リモートノードがTmaxのサービスを呼び出す最も一般的な方式です。X25GWはリモートノードのチャンネル情報を保存しておき、サービスから結果が受信されたら保有しているチャンネルのうち該当のチャンネルを見つけ、結果を転送します。そのとき、該当のチャンネルに結果を転送する前に、別の要求を受けることも可能です。すなわち、X25GWはリモートノードで要求したチャンネルはブロックせず、次の要求が受けられるように処理するため、X25GWを運用する方式によって異なる処理ができます。

下記は、同期型X25GWでのリモート同期型呼び出し方式の動作構造です。

[図 1.5] リモート同期型X25GWの動作構造



1. X25GWとリモートノードが接続されています。
2. リモートノードはX25GWと接続されているチャンネルにメッセージを転送します。
3. X25GWはtpacall()でTmaxサービスを呼び出します。
4. X25GWはサービス処理結果を受け、メッセージを要求したチャンネルを検索します。
5. 該当のチャンネルが正常に接続されていたら、結果を転送します。

## 1.2.2. 非同期型X25GW

非同期型呼び出し方式は、Tmaxのクライアント、サーバーまたはリモートノードがX25GWサービスを要求するのみで、その結果は受けないか、あるいはサービスを要求した場所ではなく別のプログラムで処理する方式です。TmaxのサービスからX25GWにサービスを要求し、その結果は別のサービスで受け取ることができます。しかし、クライアントの場合は、非同期型で処理する際にはtpacallを利用して応答を受けない場合のみ可能です。一方、リモートノードからサービスを先に要求する際、サービスのみ要求して結果は受けないか、サービスを要求して結果は別のチャンネルに渡す方式が非同期型方式です。

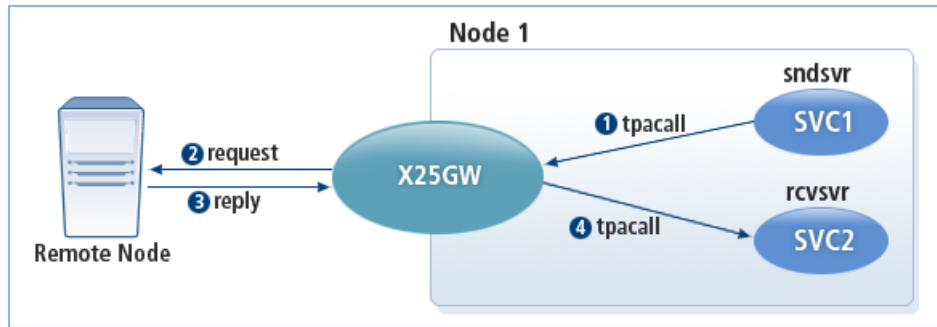
### Tmaxからのサービス要求方式

TmaxのサービスからX25GWにtpacallのTPNOREPLYで要求し、当該サービスは終了します。X25GWはリモートノードに要求を送信し、その結果が受信されたらTmaxの別のサービスをtpacallのTPNOREPLYで呼び出して結果を処理する方式です。

下記は、非同期型X25GWでTmaxからサービスを要求する場合の動作構造です。



**[図 1.6] Tmax要求の非同期型X25GWの動作構造**



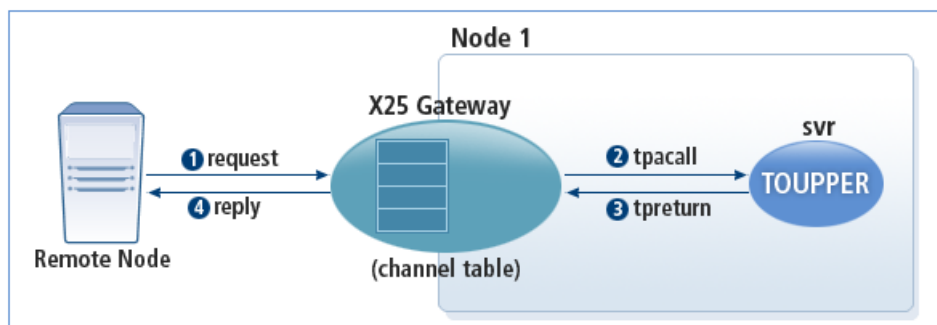
1. TmaxサービスからX25GWにサービスを要求します。(tpacallのTPNOREPLY)
2. X25GWがリモートノードにデータを転送します。
3. リモートノードからX25GWにデータが受信されます。
4. X25GWがTmaxの別のサービスを呼び出します。(tpacallのTPNOREPLY)

## リモートからのサービス要求方式

リモートノードで先にX25GWにサービスを要求すると、X25GWはtpacallでTmaxサービスを要求します。Tmaxのサービス処理が完了したら、X25GWはリモートノードと接続されているチャンネルのうち、使用可能なチャンネルに処理結果を転送します。

下記は、リモートノードで非同期型X25GWにサービスを要求する場合の動作構造です。

**[図 1.7] リモート要求の非同期型X25GWの動作構造**



1. リモートノードでX25GWにサービスを要求します。
2. X25GWがTmaxのサービスをtpacallで要求します。
3. Tmaxサービスの処理結果をX25GWに転送します。

4. X25GWがチャンネル表からリモートノードと接続されているチャンネルを検索して結果を転送します。

## 1.3. ゲートウェイのその他の機能

以下は、ゲートウェイのその他の機能についての説明します。

### 1.3.1. ゲートウェイ・ヘッダー

X25GWは、Tmaxクライアントまたはサーバーで呼び出す場合、ゲートウェイ・ヘッダーを使用することができます。ゲートウェイ・ヘッダーを使用するには、X25GWの別のライブラリーを使用する必要があります。通常、libx25gw.a、libx25gw.soを使用しますが、ゲートウェイ・ヘッダーが使用したい場合は、libx25gw.gwh.aまたはlibx25gw.gwh.soを使います。ゲートウェイ・ヘッダーは、全データ・バッファの最初のオフセットに位置する必要があります。ユーザー・ヘッダーを使用する場合、ゲートウェイ・ヘッダーの次にユーザー・ヘッダーが位置する必要があります。

ゲートウェイ・ヘッダーは様々な目的で使用されますが、X25GWではsvc項目のみ使用可能です。非ブロック・モードで使用するか、または非同期方式で使用する際、応答データに対してサービス名を指定する場合に使用します。ゲートウェイ・ヘッダーは、通常ユーザー・ヘッダーを使用せず、応答を処理するサービスをメッセージ別に処理するときに使用されます。

### 1.3.2. サービス名の検索順

非ブロック型X25GWまたは非同期型X25GWの場合、リモートノードからの要求や応答を処理するTmaxのサービスが必要となります。X25GWはサービス名が把握できないため、ユーザーが適当なサービス名を指定する必要があります。X25GWは、下記のように3つの方式でサービス名を検索します。サービス名は、Tmax環境ファイルに登録されている必要があります。

- ゲートウェイ・ヘッダーを使用する場合のみ可能です。TmaxクライアントまたはサービスでX25GWを要求する際、ゲートウェイ・ヘッダーにサービス名を入力してX25GWを呼び出すと、X25GWは優先的にこのサービス名を使用します。
- ユーザー・ヘッダーでサービス名を検索します。TmaxクライアントまたはサービスでX25GWを要求する際、ユーザー・ヘッダーにサービス名を入力した後、ユーザー関数のget\_service\_nameでサービス名を取得することができます。
- CLOPTの[-S]オプションに指定したサービスを利用します。この場合、全メッセージに対して同じサービスが適用されます。

---

## 参考

リモートノードで最初にサービスを要求する場合、上記の方式を従わずユーザーが`get_msg_info`でサービス名を指定する必要があります。

---

### 1.3.3. ユーザーが任意でチャンネルを指定

完全な非同期方式でX25GWを構成した場合、ユーザーはリモートノードに転送するチャンネルを指定することができます。CLOPTの`[-a]`を使用します。完全な非同期方式は相互間の応答はありません。応答データもサービス要求方式で渡される必要があります。例えば、TmaxのクライアントまたはサービスからX25GWに`tpacall`の`TPNOREPLY`で呼び出すと、X25GWはサービス要求に対するUIDやいかなる情報も保存せず、リモートノードにデータを転送した後、当該サービスを終了します。以後、リモートノードから応答メッセージが受信されてもこれをサービス要求として処理します。リモートノードから先に要求した場合も処理方式は同様です。

上記のような場合にユーザーは、CLOPTの`[-u]`オプションを使用してリモートノードに転送するチャンネル番号を指定することができます。チャンネル指定には、下記の2つの方法があります。

- Tmaxから先にリモートノードにサービスを要求する場合、X25GWは`get_channel_num`関数を呼び出します。ユーザーは転送データを分析してチャンネルを指定することができます。
- リモートノードから先にサービスを要求する場合は、リモートノードでサービスを要求する際、`get_msg_info`関数でリモートノードが要求したチャンネルをデータに保存し、Tmaxサービス呼び出します。Tmaxサービスはリモートノードに結果を転送するため、X25GWに`tpacall`の`TPNOREPLY`でサービスを再転送します。ユーザーは`get_channel_num`で、以前保存しておいたチャンネルを使用してリモートノードに応答を転送することができます。X25GWとTmaxエンジン間のチャンネルはブロックされず、サービスを要求したチャンネルに応答を転送することができます。

### 1.3.4. リセット処理方法

X25GWはリモートと任意のリセットをやり取りすることができます。ただし、システムの障害、プログラムの不具合、リモートとの環境設定に誤りがあり、異常に多いリセットをやり取りしてしまう問題が頻繁に発生しています。このように頻繁にリセットが発生するとゲートウェイはルーピング(Looping)され、業務処理ができなくなります。この問題を事前に防ぐため、ゲートウェイでは指定された連続リセット回数(100回)を超過したら、自動的に当該チャンネルのセッションを解除して再接続するようにし、リセットの異常問題を解決しています。なお、正常にリモートで読み込むと、リセット回数はクリアされます。



## 第2章 環境設定

本章では、Tmaxが提供するX.25ゲートウェイの環境構成および設定方法について説明します。

### 2.1. 概要

以下は、X25GWサーバーを構成する際に必要なファイルです。

- UNIX

下記は、UNIXで使用されるファイルです。

ディレクトリー	ファイル名
lib	lbx25gw.a、libx25gw.so、libtmaxgw.a、libtmaxgw.so
lib64	lbx25gw.a、libx25gw.so、libtmaxgw.a、libtmaxgw.so

- Windows

下記は、Windowsで使用されるファイルです。

ディレクトリー	ファイル名
lib	x25gw.dll、tmaxgw.dll
lib64	x25gw.lib、tmaxgw.lib

上記のファイル(X25GWライブラリー)は、Tmaxのインストール時に各ディレクトリーの下位に作成されます。ただし、custom.c、custom.hは別途に提供されるため、必要な場合は技術サポートの担当者にリクエストしてください。

X25GWライブラリーと適切に実装したcustom.c、custom.hをコンパイルしてX25GWを作成します。

### 2.2. Tmax環境構成

X25GWを使用するには、Tmax環境ファイルにX25GW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型サーバーと登録方法が類似していますが、SVRTYPEがUCSでCUSTOM\_GATEWAYという点が異なります。X25GWを使用するためにTmax環境ファイルを修正するときは、SERVER、SERVICEセクションのみ設定します。

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
x25gw     SVGNAME = svg1,
          MIN = 1,
          MAX = 1,
          CPC = 10,
          SVRTYPE=CUSTOM_GATEWAY,
          CLOPT="-- -F /home/tmax/config/x25gw.cfg -N 4096 -h 20"

*SERVICE
X25GW1    SVRNAME=x25gw, SVCTIME=20
X25GW2    SVRNAME=x25gw, SVCTIME=25

```

X25GWのプロセス数を指定する項目です。この値が1より大きい場合は、X25GW環境ファイルにプロセス別に異なるチャンネル情報を定義する必要があります。(「[2.3. X25GWチャンネル環境設定](#)」の内容を参照してください)

項目	説明
MIN	<p>X25GWのプロセス数を指定します。値が1より大きい場合は、X25GW環境ファイルにプロセス別に異なるチャンネル情報を定義する必要があります。</p> <p>詳細内容については、<a href="#">「2.3. X25GWチャンネル環境設定」</a>を参照してください</p>
CPC	<p>TmaxエンジンとX25GW間のチャンネル数を指定します。Tmaxのクライアントまたはサービスからリモートノードに要求する場合、同時に要求する数の分だけCPC数を指定します。非ブロック型の場合は、同時に要求する数の分だけ指定する必要はなく、適当に指定します。一方、X25GWは、Tmaxエンジンにtpacallで要求するため、多くのチャンネルを使用する必要はありません</p>
CLOPT	<p><a href="#">CLOPT項目(X25GWオプション)</a>の内容を参照してください</p>
SVCTIME	<p>X25GWは、1つのサーバー・プロセスに複数のサービスを指定し、サービスごとに異なるSVCTIME値を適用することができます。したがって、*SERVICEセクションに複数のサービスを指定し、サービスごとに異なるSVCTIME値を指定したら、X25GWは要求したサービス別に指定したサービス・タイム値を使用します</p>

## CLOPT項目(X25GWオプション)

X25GW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項目に多くのオプションを設定することができます。オプションによってX25GW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。

オプション	説明
[-r]	リモートノードとの接続が切断された場合、再接続の回数を指定します。  X25GWは、リモートノードとチャンネル別に指定した回数を超過して切断された場合は、再接続を行いません(デフォルト値: 無限)
[-N]	X25GWとリモートノード間に送受信するメッセージの最大サイズです。X25GWでは、オプションで指定したサイズのみだけメモリーを確保して受信するため、十分なサイズを指定する必要があります
[-n]	X25GWとリモートノード間に、一度で送受信できるサイズを指定します。  [-N]オプションで指定した全体メッセージをこのオプションで指定したサイズ単位で送受信します。オプションのサイズは[-N]オプションの値より大きくてはなりません(デフォルト値: [-N]オプションと同じです)
[-t]	X25GWで使用するチャンネルとリモートノードの接続が切断された場合、直ちにリモートノードとチャンネルを再接続するのではなく、設定時間ごとに接続できるように時間(秒)を指定するオプションです。  X25GWはチャンネルの接続が切断された場合、指定された時間間隔で再接続を行います(デフォルト値: 3秒)
[-k]	ユーザーが共有メモリーを確保して、X25GW環境ファイルで定義したチャンネルの状態が保存できるように共有メモリーのキー値を入力するオプションです。  X25GWは共有メモリーを確保せず、すべてをユーザーに渡して任意でできるようにします。X25GWが最初に実行される際、ユーザー・ルーチン(init_remote_info)の呼び出し時に渡し、ユーザーが共有メモリーを確保するときに使用されます
[-F]	X25GWのチャンネル情報が登録されている環境ファイル名を指定します
[-S]	非ブロック・モードでX25GWを使用するか、または非同期方式で使用する場合、X25GWが要求するサービス名を登録するオプションです。  非ブロック・モードで使用する時、Tmaxの送信サービスはX25GWにtpforwardし、X25GWはリモートノードで受信した応答を受信サービスにtprelayするため、relayされるサービス名を登録します。  非同期通信方式でのTmaxクライアントまたはサービスは、X25GWにサービスを要求(tpacall with TPNOREPLY)して終了します。その後、リモートノードから受信した応答は、通常TPNOREPLYで呼び出したためデータを捨てることになります。しかし、このオプションにサー

オプション	説明
	ビス名を指定すると、X25GWは指定したサービスにtpacallで応答データを転送します。このオプションで指定したサービスは登録されているサービス名である必要があります
[-H]	<p>TmaxクライアントまたはサービスとX25GWの間にユーザー情報データを送受信する場合に使用します。オプションで指定したデータはリモートノードに転送せず、一時的にX25GWで保存しておき、応答が受信されたらユーザー情報データに応答データを加えて呼び出したサービスに渡します。</p> <p>非ブロック・モードで使用する際、送受信サービスが分離されているため、送信サービスから受信サービスにデータを転送する場合に使用します</p>
[-h]	上記の[-H]と同じオプションです。ただし、上記のオプションは同期方式/非同期方式の両方とも適用できますが、このオプションは、tpforwardのtprelayで使用される呼び出しにのみ適用されます
[-a]	X25GWを非同期通信方式で使用するときに使われるオプションです。非同期通信方式はサービスを要求してから終了する形式です。X25GWはサービスに対する応答も別のサービス要求として認識します
[-u]	Tmaxクライアントまたはサービスから、X25GWを通じてリモートノードにサービスを要求する際、ユーザーがチャンネルを指定できるようにするオプションです。このオプションを使用するには、[-k]オプションを使用してチャンネルのステータス情報を保有している必要があります。X25GWは、ユーザーが指定したチャンネルが使用できない場合はエラーを返します
[-Y]	<p>リモートノードでTmaxのサービスを要求した後、サービスは正常に処理されたが結果をリモートノードに転送できない場合、以前処理したサービスに対してキャンセル処理ができるように、復旧サービスを指定するオプションです。</p> <p>リモートノードでTmaxのサービスを呼び出した後、リモートノードとセッションがすべて切断された場合、サービス処理結果はリモートノードに転送されず捨てられます。そのとき、サービスは正常に処理されたがリモートノード側ではエラーとして認識するため、問題が発生することもあります。この場合、ユーザーがこのオプションにサービス名を指定したら、ゲートウェイはリモートに結果が転送できないとき、リモートノードに転送するデータを、指定したサービスで呼び出すことになります</p>
[-y]	UIDとして利用できるメッセージ番号の最大値を設定するオプションです(デフォルト値: 50000、最大268435455まで指定可能)

## 2.2.1. サービス・ブロック型X25GW

X25GWはリモートノードと接続されたら、同期型/非同期型の両方とも使用できます。

\*DOMAIN

...



```

*NODE
...

*SVRGROUP
...

*SERVER
x25gw    SVGNAME=svg1,
        MIN=1,
        MAX=1,
        CPC=10,
        SVRTYPE=CUSTOM_GATEWAY,
        CLOPT="-- -F /home/tmax/config/x25gw.cfg"

*SERVICE
X25GW    SVRNAME=x25gw, SVCTIME=30

```

上記のように環境ファイルを作成した場合、Tmaxのブート時にx25gwという名前のX25GWの1つが起動されます。X25GWのサービス名はX25GWで、タイムアウトが30秒なので、30秒以内に応答がない場合はタイムアウトを返します。ただし、Tmaxのクライアントまたはサービスからリモートノードに要求したサービスに対してのみタイムアウトをチェックし、リモートノードから要求したサービスに対しては要求した該当のサービスでタイムアウトを設定します。

Tmaxのクライアントまたはサーバーからリモートノードにサービスを要求する際、X25GWはUIDを使用します。UIDはメッセージ別の一意の値として要求に対する応答時に必ずX25GWに渡します。そうすると当該サービスを要求したサービスまたはクライアントに返されます。

リモートノードでサービスを要求する場合、X25GWはそれに対する応答をサービスを要求したチャンネルに転送します。ただし、要求したチャンネルが切断された場合は、データは消滅されます。

## 2.2.2. サービス非ブロック型X25GW

通常、Tmaxのサービスからリモートノードにサービスを要求する場合に使用します。そのため、一般のTmaxクライアントではこの方式で直接サービスを呼び出すことができません。Tmaxクライアントで直接使用できる方式は、ブロック型方式または非同期型方式です。

```

*DOMAIN
...

*NODE
...

*SVRGROUP
...

*SERVER

```

```

x25gw      SVGNAME=svg1,
           MIN=1,
           MAX=1,
           CPC=5,
           SVRTYPE=CUSTOM_GATEWAY,
           CLOPT="-- -F /home/tmax/config/x25gw.cfg -S RECVSVC -H 20"
sendsvr    MIN=1, MAX=1
secvsvr    MIN=1, MAX=1

*SERVICE
X25GW      SVRNAME=x25gw, SVCTIME=30
SENDSVC    SVRNAME=sendsvr
RECVSVC    SVRNAME=recvsvr

```

上記のように環境ファイルを作成した場合、Tmaxのブート時にx25gwという名前のX25GWの1つが起動されます。x25gwのサービス名はX25GWで、タイムアウトが30秒なので、30秒以内に応答がない場合はタイムアウトを返します。

Tmaxのクライアントまたはサーバーから先にSENDSVCサービスを呼び出すと、SENDSVCで事前作業を行った後、X25GWサービスに制御権を渡します。(tpforward)

X25GWは、まずTmaxエンジンにチャンネルの解除メッセージを転送した後、リモートノードにサービスを要求し、サービスで応答を受信したら[-S]オプションに指定したサービスに結果を渡します(tprelay)。RECVSVCが処理結果に対する作業を実行して返したら、SENDSVCを呼び出したクライアントまたはサービスに応答が渡されます。この方式を使用すると、少ないサービス(SENDSVC)でより多くの処理ができます。ブロック型のように必ずUIDを使用します。

リモートノードの応答が指定した時間(30秒)を超過した場合、X25GWはタイムアウトをRECVSVCに渡します(tprelay)。したがって、RECVSVCサービスでは必ずtpurcode値を確認し、値がゼロより大きいときはエラーが発生した場合なので、これに対する適切な処理を行う必要があります。タイムアウトのみならず、すべてのエラーに対してtpurcodeに値が渡されます。詳細内容については、「[付録 A. X25GWのエラーコード](#)」を参照してください。

## 2.2.3. リモート同期型と非同期型X25GW

リモートノードで要求するサービスの同期型/非同期型の区分は、X25GW環境ファイルの応答タイプに従って決めるか、またはユーザー定義関数(get\_msg\_info)のflagsにTPNOREPLYを設定して決めます。前者の場合、応答タイプを「no」に設定するとサービスに対する応答は別のチャンネルに転送されます。後者の場合は、リモートノードに応答を転送しません。

```

*DOMAIN
...
*NODE
...

```

```

*SVRGROUP
...

*SERVER
x25gw          SVGNAME=svg1,
               MIN=1,
               MAX=1,
               CPC=5,
               SVRTYPE=CUSTOM_GATEWAY,
               CLOPT="-- -F /home/tmax/config/x25gw.cfg"

*SERVICE
X25GW          SVRNAME=x25gw

```

ユーザー定義関数(get\_msg\_info)のflagsにTPNOREPLYを設定すると、サービス応答はリモートノードに転送されません。リモートノードでTmaxのサービスを要求する際、X25GWがtpacallを使用するため、応答を受けず同時にtpacallが使用できる個数が制限されます(デフォルト値:8個)。同時要求が8個以上の場合は、\*DOMAINセクションにMAXSACALL数を適切に指定する必要があります。

## 2.3. X25GWチャンネル環境設定

X25GWはリモートノードと通信するためのチャンネル情報を別途のファイルに登録し、当該ファイル名を[-F]オプションに登録する必要があります。X25GWチャンネル情報ファイルに登録する方法は、X25GWを使用するマシンによって少々異なります。そのため、リモートノードと円滑な通信を行うには、マシン別のチャンネル情報を正確に登録する必要があります。なお、チャンネル情報登録の項目について十分に理解する必要があります。

まず、各項目について説明した後、マシン別の相違点について記述します。

### 2.3.1. チャンネル環境ファイル

```

#####
#####
# x25gw_no | link_no | start_LCN_no | num_LCN | dir | reply_dedicated |
#          devpath [ | group_id]
#
# x25gw_no : x25gw process number. 0から開始します。
# link_no  : x25 link number
# start_LCN_no : LCN (LU) 開始番号
# num_LCN   : LCN (LU) 個数
# (start_LCN_no ~ (start_LCN_no + num_LCN))
# dir      : LCN チャンネルの方向
#   in - 外部マシン to tmax request 専用 (Send)
#   out - tmax to 外部マシン request 専用 (Receive)
#   any - any direction

```

# reply\_dedicated : yesの場合は、このLCNからrequestされたメッセージの応答は必ずこのLCNで出力されます。

noの場合は、空きのLCNで出力されます。

# devpath : x25 device path

# group\_id : チャンネル・グループ名(省略可能)

#####

0 0 3 2 out no /dev/x25pkt

0 0 1 2 in no /dev/x25pkt

下記は、基本的なチャンネル環境ファイルの項目に関する説明です。

項目名	説明
ゲートウェイ番号	Tmax設定ファイルのMINが2以上の場合、1つのX25GWチャンネル情報ファイルにすべて登録できます。X25GWプロセスを区分するためのゲートウェイ番号です。X25GWプロセスは、自身のゲートウェイ番号に該当のチャンネル情報をロードしてリモートノードと接続します。ゼロからスタートし、MIN値の-1まで表現します
リンク番号	下記のLCNが属しているリンク番号を登録します。1つのリンクに多くのLCN(LU)を有することができます。すなわち、多くのLCNが1つのリンクを共有してリモートノードと通信します
開始LCN番号	論理的な番号です。1つの物理的なリンクを多くのロジカルLUで分割して、リモートノードとLU別に通信することができます。LUの開始番号を入力します
LCN個数	開始LCN番号からロジカルLUの個数を指定します。X25GWは、開始LCNからLCN個数の分だけのチャンネルをリモートノードと接続します
チャンネル・タイプ	指定したチャンネル・タイプが、IN_CHANNEL、OUT_CHANNEL、またはANY_CHANNELなのかを区分する項目です  – IN_CHANNEL: リモートノードからX25GWに要求が受信されるチャンネルです  – OUT_CHANNEL: X25GWからリモートノードに要求が送信されるチャンネルです  – ANY_CHANNEL: 指定すればIN/OUT両方とも使用可能なチャンネルです。使用できる値は、in、out、anyです
応答タイプ	リモートノードでTmaxサービスを呼び出した後、応答が送信されるチャンネルを指定します  – yes: 要求を送信したチャンネルに応答が転送されます  – no: OUT_CHANNELまたはANY_CHANNELに指定したチャンネルのうち、使用可能なチャンネルに応答が転送されます
デバイス名	X25GWのデバイス名を指定します。マシン別に若干異なります

項目名	説明
チャンネル・グループ名	チャンネルをグループ別に区分して使用する際に登録して使うことができます。2つ以上のチャンネルが同グループに属すると、ラウンドロビン方式の負荷分散をサポートします

## 2.3.2. マシン別のチャンネル環境ファイル

下記は、マシン別のチャンネル環境ファイルに関する説明です。

マシン	項目名	説明
IBM、SUN		<a href="#">「2.3.1. チャンネル環境ファイル」</a> で説明した方法を参照して登録してください
HP	デバイス名	HPマシンでは、デバイス名の項目にデバイス名を入力せず、インターフェース名(リンク名)を入力します。以外の項目はIBMおよびSUNと同様です
NCR	開始LCN番号	NCRマシンは、LCN(LU)別にローカル名が存在するため、LCN別にローカル名を別途指定する必要があります。常に「1」に指定します
	LCN個数	常に「1」に指定します
	デバイス名	NCRは、ローカル名とリモート名で相互接続するため、ローカル名とリモート名をLCN別に登録する必要があります。各LCNに該当するローカル名を登録する項目です
	デバイス名2	NCRの場合のみ使用する項目です。リモートLCN名を入力します

## 2.4. ユーザー・ヘッダー環境設定

X25GWは、ユーザー・ヘッダーを設定して使用できます。ユーザー・ヘッダーは、Tmaxクライアントまたはサービスからリモートノードにサービスを要求する場合のみ使用可能で、逆の場合は使用できません。ユーザー・ヘッダーとして指定したデータはリモートノードに転送されず一時のX25GWでUID別に保存され、応答が受信されたら応答データからUIDを見つけ、当該UIDのユーザー・ヘッダー・データと応答データを渡すこととなります。ユーザー・ヘッダー・データの長さは、最大256バイトまで使用できます。

ユーザー・ヘッダーは、全モードで利用できるオプションで、2つに分けて指定することが可能です。tpforward方式で使用する場合(非ブロック型)と、その他のサービス(ブロック型、非同期型)を使用する場合で、異なるユーザー・ヘッダー長が指定できます。

```
*DOMAIN
...

*NODE
...

*SVRGROUP
```

```

...
*SERVER
x25gw          SVGNAME=svg1,
               MIN=1,
               MAX=1,
               CPC=10,
               SVRTYPE=CUSTOM_GATEWAY,
               CLOPT="-- -F /home/tmax/config/x25gw.cfg -H 9 -h 10"

*SERVICE
X25GW          SVRNAME=x25gw

```

[-H]オプションは、全タイプのサービス要求に対してユーザー・ヘッダーを指定する際に使用するオプションです。一方、[-h]オプションは、tpforward方式(非ブロック型)でのみ使用できるオプションです。X25GWは、ユーザー・ヘッダーが指定された場合、リモートノードに転送するデータの中で、一般のサービスは9バイトを、tpforwardの場合は10バイトを保存し、残りのデータのみリモートノードに転送します。(上記の設定の場合)

リモートノードから応答が受信されたら、保存されているユーザー・ヘッダーに応答データを加えて返します。ユーザー・ヘッダーは、ユーザーが任意で利用できるデータなので、様々な方法で 사용할 ことができます。

下記は、ユーザー・ヘッダーを使用する一般的な方法です。

- tprelayおよび非同期サービス名を指定

ユーザー・ヘッダーに非ブロック型で使用する場合、tprelayされるサービスを指定することができます。この場合、[-S]オプションで指定したサービスより優先されます。ユーザー・ヘッダーにもサービスを指定し、[-S]オプションでサービスを指定した場合、X25GWは先にユーザー・ヘッダーからユーザー関数(get\_service\_name)を呼び出してサービスを検索します。サービスが見つからない場合は、[-S]オプションで指定したサービスを使用します。

非同期型方式(tpacallのTPNOREPLAY)で呼び出した場合も、応答データに対して上記と同様に処理します。

- キー・データの保存

非ブロック・モードで使用する場合、送受信サービスで分離されます。送信サービスでデータベースに適切な作業を処理し、リモートノードにデータを転送して送信サービスは終了します。以後、受信サービスで、送信サービスで処理したデータベースのキー情報が確認したい場合や、受信サービスのリモートノードでエラーが発生した場合、データベースを復旧するための重要なデータを保存するために使用されます。

# 第3章 ユーザー・プログラムと関数

本章では、X.25ゲートウェイでユーザー・プログラムを作成する方法とAPIについて説明します。

## 3.1. 概要

Tmaxの環境を設定してX25GWを登録した後、X25GWを使用するためにcustom.h, custom.cを適切に修正します。ユーザー・プログラムを修正したcustom.h, custom.cとX25GWライブラリーおよびregister.cをリンクしてコンパイルしたら、X25GWが完成されます。このとき作成された実行ファイル名は、Tmax環境ファイルのSERVERセクションに登録されている名前と同じである必要があります。custom.clには様々なユーザー定義関数が存在しています。詳細説明については、各節を参照してください。

## 3.2. custom.h

X25GWで使用されるmsg\_info\_t構造体を含むファイルです。ユーザーがリモートノードおよびX25GWと通信する、その他のメッセージ構造体またはヘッダーを修正することができます。しかし、msg\_info\_tは、X25GWライブラリー内部とユーザー関数が使用する構造体なので、メンバー変数、タイプ、長さを修正してはなりません。構造体は、X25GWライブラリー内部とユーザーの修正が必要なcustom.cファイルの関数で使用されます。custom.cファイルでは、リモートノードからのメッセージを受信した直後に呼び出されるget\_msg\_info()関数で、この構造体の値を適切に設定します。さらに、リモートノードにメッセージを送信する直前に呼び出されるput\_msg\_info()関数で、この構造体の値を利用してリモートノードに送信したデータを適切に構成する必要があります。

X25GWで使用されるmsg\_info\_t構造体は下記のとおりです。

```
typedef struct msg_info {
    char svc[20];
    int err;
    int len;
    int uid;
    int flags; /* flagsを設定します。(TPNOREPLYなど) */
    int msgtype;
    int channel_id;
    char sys_id[20]; /* チャンネル・グループ名 */
} msg_info_t;
```

下記は、msg\_info構造体のメンバーに関する説明です。

メンバー	説明
char svc[20]	サービス名を指定します。リモートノードからサービスを要求する際、この項目にサービス名を与えるとTCPGWはこれを利用してサービスを呼び出します
int err	サービスを要求し、応答データを受信した場合のエラー有無を示します
int len	送受信データの長さを示します
int uid	同期型通信の場合、TCPGWが作成したUIDを有する項目です
int flags	リモートノードからTmaxにサービスを要求するときに使用します <ul style="list-style-type: none"> <li>– NOFLAGS : リモートノードで要求したサービスの応答を受ける場合</li> <li>– TPNOREPLY : リモートノードで要求したサービスの応答を受けず、サービス要求のみする場合(TmaxのtpacallのTPNOREPLYと同じ)</li> </ul>
int msgtype	未使用項目です
char sys_id[20]	送受信チャンネル番号です

### 3.3. custom.c

custom.cは、リモートノードとX25GWが通信するために開発者が実装するファイルです。X25GWライブラリー(libx25gw.a、libx25gw.so)と一緒にコンパイルして使用します。関数の使用例については、「[第3章 ユーザー・プログラムと関数](#)」で説明します。

下記は、custom.cで実装する関数一覧です。

関数	説明
<a href="#">init_remote_info</a>	リモートノードと接続する前に呼び出される関数です
<a href="#">remote_connected</a>	リモートノードと接続された後に呼び出される関数です
<a href="#">remote_closed</a>	リモートノードとの接続を終了した後に呼び出される関数です
<a href="#">get_msg_info</a>	リモートノードから要求または応答が受信され、データを読み込んだ後、TCP/IPゲートウェイ・ライブラリーとcustom.cとのインターフェースの役割をする、infoを参照、または加工する関数です
<a href="#">get_channel_num</a>	Tmaxサービスまたはクライアントから要求したデータをリモートノードに転送する際、ユーザーがチャンネルを選択できるようにする関数です
<a href="#">put_msg_info</a>	リモートノードにメッセージを転送する際に呼び出される関数です
<a href="#">get_service_name3</a>	tpreply()またはtpacall()するサービス名のエラーコードに従って設定する関数です
<a href="#">prepare_shutdown</a>	TCPGWが終了する直前に呼び出される関数です
<a href="#">inmsg_recovery</a>	リモートノードからの要求をtpacall(..., TPBLOCK)で処理中にエラーが発生した場合に呼び出される関数です
<a href="#">outmsg_recovery</a>	リモートノードに要求を転送する際、エラーが発生した場合に呼び出されます



### 3.3.1. init\_remote\_info

リモートノードと接続する前に呼び出されます。X25GWが自身の初期化作業を完了した直後に呼び出す関数で、1回のみ呼び出されます。Tmax環境ファイルのCLOPTに[-k]オプションを利用して共有メモリー・キーを設定した場合、接続に関する情報などを保存するため、共有メモリーを作成するロジックが実装できます。場合によっては、内部ロジックを実装しなくても構いません。

- プロトタイプ

```
int init_remote_info (char *myname, int mynumber, int num_channel, int key)
```

- パラメータ

パラメータ	説明
myname	X25GWサーバー名です
mynumber	同じX25GWが同時に複数実行される場合、それぞれのプロセスが区別できるX25GWプロセス番号です。ゼロから始まります
num_channel	X25GWが接続している最大チャンネル数で、環境ファイルに定義したチャンネルの総数です
key	Tmax環境ファイルで、[-k]オプションを使って設定した共有メモリーのキー値です

### 3.3.2. remote\_connected

リモートノードと接続した後に呼び出される関数です。リモートノードと接続した後に必要な作業を実行します。関数はチャンネル数の分だけ呼び出されます。また、チャンネルが解除されてから再接続される際にも呼び出されます。

- プロトタイプ

```
int remote_connected(int index, int linkno, int lcnno, int type)
```

- パラメータ

パラメータ	説明
index	X25GWが接続している各チャンネルに対する自身の索引値です
linkno	接続されているチャンネルのリンク番号です
lcnno	接続されているチャンネル番号で、1つのリンクに複数のlcnが存在します。通常、ロジカルな番号として認識してください
type	リモートノードと接続されているチャンネル・タイプです。IN_CHANNELまたはOUT_CHANNELを設定します

### 3.3.3. remote\_closed

リモートノードとの接続を終了した後に呼び出される関数です。リモートノードとの接続が切断された後、必要な作業があれば行います。init\_remote\_info関数で共有メモリーを作成するロジックを実装した場合、解除する業務ロジックを実装します。チャンネル数の分だけ呼び出されます。

- プロトタイプ

```
int remote_closed(int index, int type)
```

- パラメータ

パラメータ	説明
index	それぞれのチャンネルに対する自身の索引値です
type	リモートノードと接続されているチャンネル・タイプです。IN_CHANNELまたはOUT_CHANNELを設定します

### 3.3.4. get\_msg\_info

リモートノードから要求または応答が受信され、データを読み込んだ後、データをTmaxサービス・プログラムに再要求するか、応答を転送する前に当該データ値を加工します。さらに、情報を転送するための様々な情報(uid、len、flags、service名など)、X25ゲートウェイ・ライブラリーとcustom.cとのインターフェースの役割をする、infoを参照または加工する関数です。

- プロトタイプ

```
int get_msg_info(char *data,msg_info_t *info)
```

- パラメータ

パラメータ	説明
data	リモートノードから読み込んだデータです
info	X.25ゲートウェイ・ライブラリー(libx25gw.a、libx25gw.so)とcustom.cとのインターフェースの役割をする構造体の構造です。ユーザーが受信したデータをベースにして、関数でinfo構造体の項目に各種情報を設定します

- 戻り値

Tmaxサービスに転送するタイプを定義します。X25GWはこの値をベースにして、Tmaxにどのような処理を行うかを判断します。

例えば、REMOTE\_REQUESTはリモートノードから要求が発生したと判断し、REMOTE\_REPLYはTmaxサービスから要求が発生し、リモートノードから応答が受信される場合に返す値です。リモートノードから応答を受信した場合は、必ずUID値をinfo構造体のuid項目に指定します。なお、その他の値も状況に合わせて指定します。

### 3.3.5. get\_channel\_num

Tmaxサービスまたはクライアントから要求したデータをリモートノードに転送する際、ユーザーがチャンネルを選択できるようにします。ユーザーは与えられたデータの特性に合わせて、転送するチャンネルを指定することができます。こちらで指定するのは、リモートノードと接続されているソケット番号ではなく、単純なチャンネル番号です。X25GWは、ユーザーが指定したチャンネルが使用できない場合は、エラーを返します。

- プロトタイプ

```
int get_channel_num(char *data)
```

- パラメータ

パラメータ	説明
data	リモートノードに転送するためのデータです

- 戻り値

チャンネル番号を返します。

### 3.3.6. put\_msg\_info

リモートノードにメッセージを転送する際に呼び出します。同期型通信の場合は、ユーザーがこの関数でUIDをメッセージに保存する必要があります。UIDは、info構造体のuid項目の値を使用するか、またはユーザーが任意のUIDを作成して使用した後、infoのuid項目に加えます。

- プロトタイプ

```
int put_msg_info(char *data, msg_info_t *info)
```

- パラメータ

パラメータ	説明
data	リモートノードに転送するデータです
info	リモートノードに転送するデータの情報です

- 戻り値

実際にリモートノードに転送するデータの全長を返します。メッセージ・ヘッダーと実データを足した長さを返します。

### 3.3.7. get\_service\_name3

Tmaxからリモートノードに要求を転送する際、要求を送信するサーバーと結果を受信するサーバーが異なる、非ブロック型または非同期型のX25GWを構成する場合、tpreply()あるいはtpacall()するサービス名をエラーコードに従って設定します。

- プロトタイプ

```
int get_service_name(char *header, int err, char *svc)
```

- パラメータ

パラメータ	説明
header	[-H]、[-h]オプションを利用して設定した、X25GWで保存しているユーザー・ヘッダーのポインターです
err	エラーコードです
svc	tpreply()またはtpacall()を受けるサービス名を設定します

### 3.3.8. prepare\_shutdown

X25GWが終了する直前に呼び出される関数です。通常、init\_remote\_info()関数で作成した共有メモリーを解除する役割をします。

- プロトタイプ

```
int prepare_shutdown(int code)
```

- パラメータ

パラメータ	説明
code	シャットダウン・コードで、現在は使われていません

### 3.3.9. inmsg\_recovery

リモートノードからの要求をtpacall(..., TPBLOCK)で処理した場合、サーバーが有効でないときはエラーを返しますが、その際に呼び出されます。ユーザーは関数内で適当に新しいデータを作成し、データのサイズを返します。

- プロトタイプ

```
int inmsg_recovery(char *data, msg_info_t *info)
```

- パラメータ

パラメータ	説明
data	リモートノードから読み込んだデータです
info	リモートノードから読み込んだデータの情報は、 有効な値は下記のとおりです – info→svc : tpacall()したサービス名 – info→len : tpacall()したデータ長 – info→err : エラーが発生する場合のtperrno – info→uid : 以前のget_msg_info()時に作成されていたUID値

- 戻り値

ユーザーは、関数で新しいデータの長さを返します。データ長がゼロより大きい場合、info→svcに値が存在したら該当のサービスにtpacall(..., TPNOREPLY)し、以外はリモートノードに応答を転送します。データ長が負の数の場合はデータを捨てます。

### 3.3.10. outmsg\_recovery

リモートノードに要求を転送する際、エラーが発生する場合に呼び出されます。ユーザーは、関数内で適当に新しいデータを作成し、サービス名およびUIDなどを設定した後、データのサイズを返します。

- プロトタイプ

```
int outmsg_recovery(char *data, msg_info_t *info)
```

- パラメータ

パラメータ	説明
data	リモートノードに転送するデータです
info	<p>リモートノードに転送するデータの情報です。</p> <p>有効な値は下記のとおりです</p> <ul style="list-style-type: none"> <li>– info→len : データ長</li> <li>– info→uid : X25GWが任意で作成したUID値</li> <li>– info→msgtype: 1000以上の場合、外部へ応答を送信する際にエラーが発生し、1000未満の場合は外部へ要求を送信する際にエラーが発生します</li> <li>– info→err : TPECLOSEの場合、応答を待っている間にチャンネルが切断され、エラーが発生します。以外の場合は、要求を送信する際にエラーが発生します</li> </ul>

- 戻り値

ユーザーは新しいデータの長さを返します。データの長さがゼロより大きい場合、info→svclに値が存在したら該当のサービスにtpacall(..., TPNOREPLY)します。以外は、info->msgtypeが1000以上の場合はデータを捨て、info→msgtypeが1000未満の場合はデータの呼び出し元に返します。

## 3.4. register.c

ユーザー関数登録ファイルです。ユーザーが登録した関数のみゲートウェイ・ライブラリーから呼び出されます。ゲートウェイのコンパイル時に必ずregister.cを含ませます。使用しない関数は、NULL登録します。

下記は、register.cの例です。

```
#include <stdio.h>
#include "custom.h"

extern int init_remote_info(char *name, int index, int n, int key);
extern int prepare_shutdown();
#ifdef _NCR_X25
extern int remote_connected(int index, int pcid, char *lname, char *rname, int type);
#else
extern int remote_connected(int index, int linkno, int lcno, int type);
#endif
extern int remote_closed(int index, int type);
extern int get_msg_info(char *data, msg_info_t *info);
extern int put_msg_info(char *data, msg_info_t *info);
extern int get_service_name(char *header, int err, char *svc);
extern int get_channel_num(char *data);
extern int inmsg_recovery(char *data, msg_info_t *info);
```

```

extern int outmsg_recovery(char *data, msg_info_t *info);

/*****
 * int
 * _register_custom()
 *
 * returns no used
 * [function number]
 * 1. init_remote_info
 * 2. prepare_shutdown
 * 3. remote_connected
 * 4. remote_closed
 * 5. get_msg_info
 * 6. put_msg_info
 * 7. get_service_name
 * 8. get_channel_num
 * 9. inmsg_recovery
 * 10. outmsg_recovery
 *****/
int
_register_custom()
{
    _x25gw_regfn(1, init_remote_info);
    _x25gw_regfn(2, prepare_shutdown);
    _x25gw_regfn(3, remote_connected);
    _x25gw_regfn(4, remote_closed);
    _x25gw_regfn(5, get_msg_info);
    _x25gw_regfn(6, put_msg_info);
    _x25gw_regfn(7, get_service_name);
    _x25gw_regfn(8, get_channel_num);
    _x25gw_regfn(9, NULL);
    _x25gw_regfn(10, NULL);

    return 1;
}

```





# 第4章 例

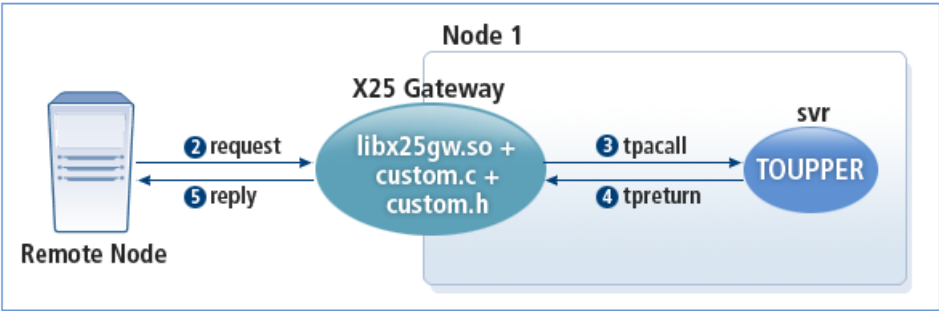
本章では、X25GWのサービス・タイプ別の例について説明します。

## 4.1. アウトバウンドX25GW

Tmaxのブート時にX25GWが起動されており、リモートノードの要求が受信されたら、ユーザーが指定したサービス呼び出した後、リモートノードに処理結果を渡します。なお、リモートノードの状況に合わせて custom.cを修正し、X25GWを構成します。

下記は、アウトバウンドX25GWの動作構造です。

[図 4.1] アウトバウンドX25GWの動作構造



下記は、アウトバウンドX25GWのプログラム構成です。

区分	ファイル名
環境ファイル	x25gw.m、x25gw.cfg
X25GW	custom.c、custom.h
サーバー	svr.c

### 4.1.1. 環境ファイル

#### <x25gw.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
x25gw          SVGNAME=svg1,
               MIN=1,
               MAX=1,
               CPC=5,
               SVRTYPE=CUSTOM_GATEWAY,
               CLOPT="-- -F /home/tmax/config/x25gw.cfg"

svr            SVGNAME=svg1,
               MIN=1, MAX=1

*SERVICE
TOUPPER        SVRNAME=svr

```

## <x25gw.cfg>

```

# gwno | link_no | start_LCN_no | num_LCN | dir | reply_dedicated| dev_path
#
0 1 1 4 any no /dev/x25pkt

```

## 4.1.2. X25GW

### <custom.h>

```

#ifndef _CUSTOM_H_
#define _CUSTOM_H_

/* ----- */
/*           Fixed structures and macros           */

#define MSG_MAGIC          "Tmax"
#define REMOTE_REQUEST     0
#define REMOTE_REPLY       1
#define SVC_NAME_LENGTH   20
#define SYSID_LENGTH       20

```

```

/* このmsg_info_tは、開発者が再定義してはいけない構造体です。 */
typedef struct msg_info {
    char    svc[SVC_NAME_LENGTH];
    int     err;
    int     len;
    int     uid;
    int     flags;
    int     msgtype;
    int     channel_id;
    char    sys_id[SYSID_LENGTH];
} msg_info_t;

typedef struct msg_body {
    char     name[16];
    char     data[100];
} msg_body_t;

#endif

```

## <custom.c>

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <sys/types.h>
#include <sys/timeb.h>
#include "custom.h"

/* 下記の関数は、不要な場合は内部ロジックを実装しなくても構いません。 */
/* ただし、X25GWライブラリー内で使用されるため定義する必要があります。 */
int init_remote_info(char *myname, int mynumber, int num_channel, int key)
{
    return 1;
}

int remote_connected(int index, int linkno, int lcnno, int type)
{
    return 1;
}

int get_msg_info(char *data, msg_info_t *info)
{
    msg_body_t *body;

    if ((info == NULL) || (data == NULL))
        return -1;
}

```

```

    body = (msg_body_t *)data;

    info->err    = 0;
    info->flags  = 0;

    memset(info->svc, 0x00, SVC_NAME_LENGTH);
    strncpy(info->svc, body->name, 8);

    /* リモートノードから要求が受信されるため、REMOTE_REQUESTを返します。 */
    return REMOTE_REQUEST;
}

int get_service_name(char *header, int err, char *svc)
{
    return -1;
}

int put_msg_info(char *data, msg_info_t *info)
{
    msg_body_t *body;

    if ((info == NULL) || (data == NULL))
        return -1;

    body = (msg_body_t *)data;

    /* body->nameを利用してエラー有無を転送します。 */
    if (info->err) /* error */
        strcpy(body->name, "Fail");
    else
        strcpy(body->name, "Success");

    /* リモートノードに要求の結果を転送するためのデータ長を返します。 */
    return info->len;
}

int get_channel_num(char *data)
{
    return -1;
}

int remote_closed(int index, int type)
{
    return 1;
}

```

```

int prepare_shutdown(int code)
{
    return 1;
}

int outmsg_recovery(char *data, msg_info_t *info)
{
    return -1;
}

int inmsg_recovery(char *data, msg_info_t *info)
{
    return -1;
}

```

## <Makefile>

```

# このMakefileはlibm 32bit用です。
#TARGETはTmax環境ファイルで定義したSERVER名と同じである必要があります。
TARGET = x25gw
APOBJS = $(TARGET).o

#X25GWを作成するためには
#libX25GW.aあるいはlibX25GW.soをリンクさせます。
LIBS = -lx25gw -ltmaxgw
OBS = custom.o register.o

CFLAGS = -q32 -O -I$(TMAXDIR) -D_DBG
LDFLAGS = -brtl

APPDIR = $(TMAXDIR)/appbin
LIBDIR = $(TMAXDIR)/lib

.SUFFIXES : .c

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

$(TARGET): $(OBS)
    $(CC) $(CFLAGS) $(LDFLAGS) -L$(LIBDIR) -o $(TARGET) $(OBS) $(LIBS)
    mv $(TARGET) $(APPDIR)/.
    rm -f $(OBS)

$(APOBJS): $(TARGET).c
    $(CC) $(CFLAGS) $(LDFLAGS) -c $(TARGET).c

```

### 4.1.3. サーバー

#### <SVR.C>

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

TOUPPER(TPSVCINFO *msg)
{
    int i;

    printf("TOUPPER service is started!\n");
    printf("INPUT : len=%d, data='%s'\n", msg->len, msg->data);

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

    printf("OUTPUT: len=%d, data='%s'\n",
        strlen(msg->data), msg->data);

    tpreturn(TPSUCCESS, 0, (char *)msg->data, msg->len, 0);
}
```

## 4.2. 同期型インバウンドX25GW

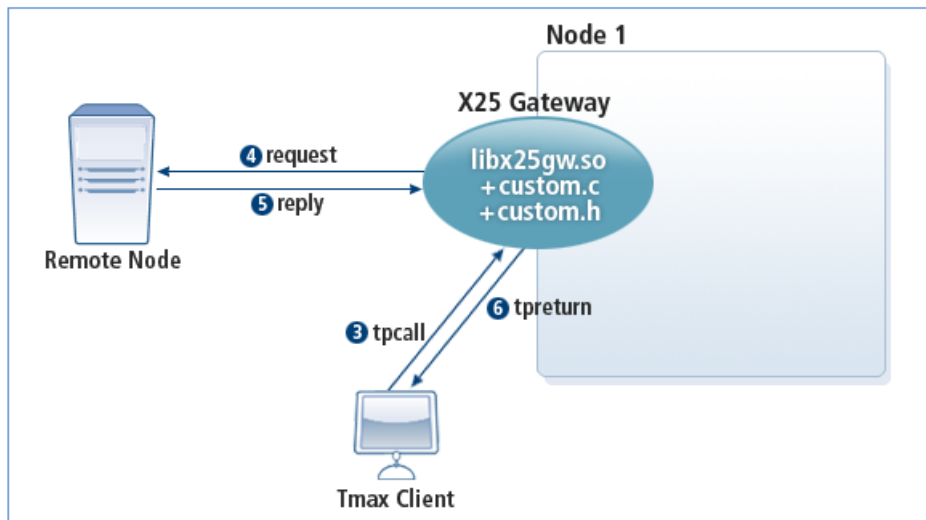
Tmaxクライアントで(または、Tmaxサービス)X25GWのサービスである「X25GW」をtpcallすると、X25GWはリモートノードにデータを転送します。リモートノードから処理結果が受信されたら、該当のクライアントに返します。

Tmaxクライアントからリモートノードにサービスを要求する際に重要なのはUIDを設定することです。UIDはX25GWライブラリー内部で指定する値(info->uid)をput\_msg\_infoの構文に保存するか、あるいはユーザーがUIDを作成して構文に保存した後、UID値をuid項目に加えます。このように、UIDを構文に保存してリモートノードに要求を転送したら、リモートノードでは当該UIDを変更せずにそのまま返します。

X25GWは応答を受信した後、get\_msg\_info関数を呼び出してUID値を取得し、X25GWの呼び出し元を判断して応答を返します。

下記は、同期型インバウンドX25GWの動作構造です。

[図 4.2] 同期型インバウンドX25GWの動作構造



下記は、同期型X25GWのプログラム構成です。

区分	ファイル名
環境ファイル	x25gw.m、x25gw.cfg
X25GW	custom.c、custom.h
クライアント	cli_x25gw.c

## 4.2.1. 環境ファイル

### <x25gw.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
x25gw    SVGNAME=svg1,
          MIN=1,

```

```

        MAX=1,
        CPC=5,
        SVRTYPE=CUSTOM_GATEWAY,
        CLOPT="-- -F /home/tmax/config/x25gw.cfg"

*SERVICE
X25GW      SVRNAME=x25gw

```

## <x25gw.cfg>

```

# gwno | link_no | start_LCN_no | num_LCN | dir | reply_dedicated | dev_path
#
0 1 1 4 any no /dev/x25pkt

```

## 4.2.2. X25GW

### <custom.h>

```

#ifndef _CUSTOM_H_
#define _CUSTOM_H_

/* ----- */
/*          Fixed structures and macros          */

#define MSG_MAGIC          "Tmax"
#define REMOTE_REQUEST    0
#define REMOTE_REPLY      1
#define SVC_NAME_LENGTH   20
#define SYSID_LENGTH      20

/*   msg_info_tは、開発者が再定義してはいけない構造体です。   */
typedef struct msg_info {
    char   svc[SVC_NAME_LENGTH];
    int    err;
    int    len;
    int    uid;
    int    flags;
    int    msgtype;
    int    channel_id;
    char   sys_id[SYSID_LENGTH];
} msg_info_t;

/* ----- */
/*          Modifiable structures and macros          */

```



```

/* このmsg_header_tとmsg_body_tは、開発者が再定義できる構造体です。 */

#define UID_FIELD 98
#define SVC_NAME_FIELD 92
#define UID_LENGTH 4
#define SVC_LENGTH 6

typedef struct msg_body {
    char      data[92];
    char      name[6];
    char      uid[4];
} msg_body_t;

#endif /* _CUSTOM_H_ */

```

## <custom.c>

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <sys/types.h>
#include <sys/timeb.h>
#include "custom.h"

/* 下記の関数は、不要な場合は内部ロジックを実装しなくても構いません。 */
/* ただし、x25gwライブラリー内で使用されるため定義する必要があります。 */
int get_channel_num(char *data)
{
    return -1;
}

int put_msg_complete(char *hp, char *data, msg_info_t *info)
{
    return 1;
}

int get_service_name(char *header, int err, char *svc)
{
    return -1;
}

int init_remote_info(char *myname, int mynumber, int num_channel, int key)
{
    return 1;
}

```

```

int remote_connected(int index, int addr, int type, int fd)
{
    return 1;
}

int get_msg_info(char *data, msg_info_t *info)
{
    info->flags = 0;

    if ((info == NULL) || (data == NULL))
        return -1;

    info->err = 0;
    info->flags = 0;
    /* info->uidを設定します。 */
    memcpy((char *)&(info->uid), (char *)&data[UID_FIELD], UID_LENGTH);

    strncpy(info->svc, (char *)&data[SVC_NAME_FIELD], SVC_LENGTH);
    info->svc[SVC_LENGTH] = 0;

    if (info->uid == 0) {
        return REMOTE_REQUEST;
    }
    else {
        return REMOTE_REPLY;
    }
}

int put_msg_info(char *data, msg_info_t *info)
{
    int nSize = 0;

    if ((info == NULL) || (data == NULL))
        return -1;

    if (info->err) {
        printf("info->err = %d\n", info->err);
        return -1;
    }
    else
        /* リモートノードに要求する場合は、必ずこのuidを */
        /* ライブラリー内部で設定した値で設定する必要があります。 */
        memcpy((char *)&data[UID_FIELD], (char *)&(info->uid),
                UID_LENGTH);

    memcpy((char *)&data[SVC_NAME_FIELD], info->svc, SVC_LENGTH);
    return info->len;
}

```

```

}

int remote_closed(int index, int type)
{
    return 1;
}

int prepare_shutdown(int code)
{
    return 1;
}

int outmsg_recovery(char *data, msg_info_t *info)
{
    return -1;
}

int inmsg_recovery(char *data, msg_info_t *info)
{
    return -1;
}

```

## <Makefile>

```

#このMakefileはibm 32bit用です。
#TARGETはTmax環境ファイルで定義したSERVER名と同じである必要があります。
TARGET = x25gw
APOBJS = $(TARGET).o

#X25GWを作成するためには、libX25GW.aあるいはlibX25GW.soをリンクさせます。
LIBS = -lx25gw -ltmaxgw
OBS = custom.o register.o

CFLAGS = -q32 -O -I$(TMAXDIR) -D_DBG
LDFLAGS = -brtl

APPDIR = $(TMAXDIR)/appbin
LIBDIR = $(TMAXDIR)/lib

.SUFFIXES : .c

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

$(TARGET): $(OBS)
    $(CC) $(CFLAGS) $(LDFLAGS) -L$(LIBDIR) -o $(TARGET) $(OBS) $(LIBS)
    mv $(TARGET) $(APPDIR)/.

```

```

rm -f $(OBJJS)

$(APOBJS): $(TARGET).c
    $(CC) $(CFLAGS) $(LDFLAGS) -c $(TARGET).c

```

## 4.2.3. クライアント

### <cli\_x25gw.c>

```

/* このプログラムは、X25GWにサービスを要求するTmaxクライアントです。
X25GWとリモートノードが接続された後、このプログラムを実行します。 */
#include <stdio.h>
#include <usrinc/atmi.h>
#include "custom.h"

int main(int argc, char **argv)
{
    int ret;
    msg_body_t *body;
    char *buf;
    long rlen;

    ret = tmaxreadenv("tmax.env", "TMAX");
    if (ret < 0) {
        printf("tmaxreadenv fail...[%s]\n", tpstrerror(tperrno));
    }

    ret = tpstart((TPSTART_T *)NULL);
    if (ret < 0) {
        printf("tpstart fail...[%s]\n", tpstrerror(tperrno));
        return -1;
    }

    buf = tpalloc("STRING", 0, 0);
    if (buf == NULL){
        printf("buf tpalloc fail...[%s]\n", tpstrerror(tperrno));
        tpend();
        return -1;
    }

    body = (msg_body_t *)buf;
    memcpy(body->data, argv[1], strlen(argv[1]));
    body->data[51] = 0;

    /* X25GWサービスを呼び出します。 */
    ret = tpcall("X25GW", buf, sizeof(msg_body_t), &buf, &rlen, 0);
    if (ret < 0){

```

```

    printf("tpcall fail...[%s]\n", tpstrerror(tperrno));
    tpfree((char *)buf);
    tpend();
    return -1;
}

body = (msg_body_t *)buf;
printf("return value = %s\n", body->data);
tpfree((char *)buf);
tpend();
}

```

### 4.3. サービス非ブロック型X25GW

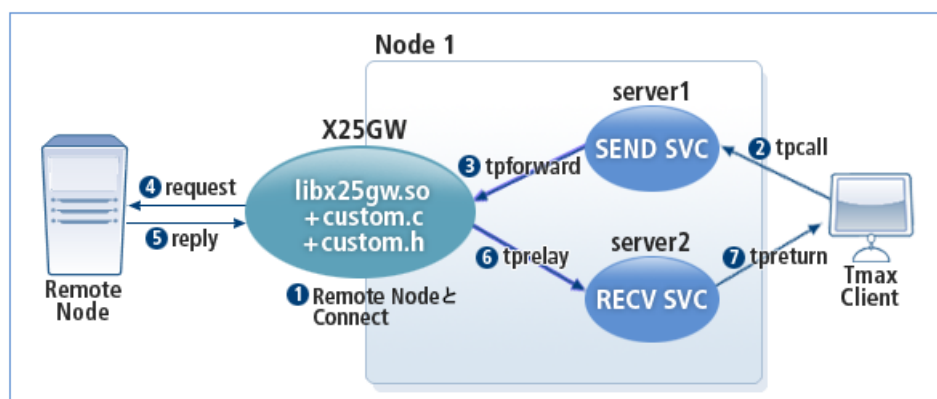
非ブロック型X25GW方式は、通常、外部と通信する場合に多く使用されます。この方式を利用すると、少ないプロセス数でもシステムへの負荷を最小限にし、多くの作業が処理できます。

ブロック型方式は、Tmaxクライアントまたはサービスで直接X25GWを呼び出して応答を受けますが、非ブロック型方式では、送信プロセスとX25GWから応答を処理する受信プロセスを分離して処理します。X25GWを呼び出そうとするすべてのプロセスが先に送信プロセスを呼び出すと、送信プロセスはX25GWを呼び出す前に事前作業を完了し、X25GWにサービス制御権を渡します。(tpforward)

次に、X25GWは渡されたサービスを処理する前に、Tmaxエンジンと接続されているチャンネルのブロック状態を解除し、リモートノードにサービスを送信します。リモートノードから応答を受信したら、X25GWは受信メッセージを処理する受信サービスを検索順に従って検索し、受信プロセスにサービス制御権を渡します。受信プロセスは応答を処理した後、最初にサービス呼び出したプロセスに結果を渡します。実際にX25GWを呼び出すサービスまたはtprelayを受けるサービスは、非同期的に動作するため、実行時間に対する負荷はありません。最初に送信サービス呼び出すプロセスは、応答が受信されるまで待機するため、ブロックされている状態です。

下記は、リモートノードがサーバーになり、X25GWがクライアントとなって接続する構造です。

[図 4.3] 非ブロック型X25GWの動作構造



下記は、非ブロック型X25GWのプログラム構成です。

区分	ファイル名
環境ファイル	x25gw.m、x25gw.cfg
X25GW	custom.c、custom.h
サーバー	sndsvr.c、rcvsvr.c
クライアント	cli_x25gw.c、custom.h

## 4.3.1. 環境ファイル

### <x25gw.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
x25gw     SVGNAME=svg1,
          MIN=1,
          MAX=1,
          CPC=5,
          SVRTYPE=CUSTOM_GATEWAY,
          CLOPT="-- -F /home/tmax/config/x25gw.cfg -S RECVSVC -H 9"
sndsvr    SVGNAME=svg1,
          MIN=1,
          MAX=1

rcvsvr    SVGNAME=svg1,
          MIN=1,
          MAX=1

*SERVICE
X25GW     SVRNAME=x25gw
```

SENDSVC	SVRNAME=sndsvr
RECVSVC	SVRNAME=rcvsvr

## <x25gw.cfg>

```
# gwno | link_no | start_LCN_no | num_LCN | dir | reply_dedicated | dev_path
#
0      1      1      4      any      no      /dev/x25pkt
```

## 4.3.2. X25GW

### <custom.h>

```
/* ----- custom.h ----- */

#ifndef _CUSTOM_H_
#define _CUSTOM_H_

/* ----- */
/*          Fixed structures and macros                      */
/* Common of Agent Define */
#define MSG_MAGIC          "Tmax"
#define REMOTE_REQUEST     0
#define REMOTE_REPLY       1
#define SVC_NAME_LENGTH   20
#define SYSID_LENGTH       20

/*   msg_info_tは、開発者が再定義してはいけない構造体です。 */
typedef struct msg_info {
    char   svc[SVC_NAME_LENGTH];
    int    err;
    int    len;
    int    uid;
    int    flags;
    int    msgtype;
    int    channel_id;
    char   sys_id[SYSID_LENGTH];
} msg_info_t;

/* ----- */
/*          Modifiable structures and macros                  */

#define UID_FIELD          58
```

```

#define SVC_NAME_FIELD          52
#define UID_LENGTH 4
#define SVC_LENGTH 6
#define MSG_KEEP_SVC_SIZE      9

typedef struct msg_body {
    char    retsvcname[9]; /* -Hオプションの分だけ内部的に保存するサービス名 */
    char    data[52];
    char    name[6];
    char    uid[4];
} msg_body_t;

/* リモートノードと通信する際は、body内部のヘッダーであるretsvcnameを除いた構造体 */
typedef struct remote_body{
    char    data[52];
    char    name[6];
    char    uid[4];
}remote_body_t;

#endif

```

## <custom.c>

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <sys/types.h>
#include <sys/timeb.h>
#include "custom.h"

/* 下記の関数は、不要な場合は内部ロジックを実装しなくても構いません。*/
/* ただし、X25GWライブラリー内で使用されるため定義する必要があります。 */
int get_channel_num(char *data)
{
    return -1;
}

/* この関数でtprelayする関数名を設定します。 */
int get_service_name(char *header, int err, char *svc)
{
    /*内部的に保存しているユーザー・ヘッダーはretsvcnameなので直ちにstrcpyします。*/
    strcpy(svc, header);
    svc[8] = 0;
    return 1;
}

```



```

int init_remote_info(char *myname, int mynumber, int num_channel, int key)
{
    return 1;
}

int remote_connected(int index, int linkno, int lcnno, int type)
{
    return 1;
}

/* get_msg_infoとput_msg_infoは同期型クライアントX25GWと同様に実装します。 */
int get_msg_info(char *data, msg_info_t *info)
{
    info->flags = 0;

    if ((info == NULL) || (data == NULL))
        return -1;

    info->err = 0;
    info->flags = 0;
    /* info->uidを設定します。 */
    memcpy((char *)&(info->uid), (char *)&data[UID_FIELD], UID_LENGTH);

    strncpy(info->svc, (char *)&data[SVC_NAME_FIELD], SVC_LENGTH);
    info->svc[8] = 0;

    if (info->uid == 0) {
        return REMOTE_REQUEST;
    }
    else {
        return REMOTE_REPLY;
    }
}

int put_msg_info(char *data, msg_info_t *info)
{
    int nSize = 0;

    if ((info == NULL) || (data == NULL))
        return -1;

    if (info->err) {
        printf("info->err = %d\n", info->err);
        return -1;
    }
    else
        /* リモートノードに要求する場合は必ずこのuidを */

```

```

        /* ライブラリー内部で設定した値で設定します。*/
        memcpy((char *)&data[UID_FIELD], (char *)&(info->uid), 4);

        memcpy((char *)&data[SVC_NAME_FIELD], info->svc, 6);

        return info->len;
    }

    int remote_closed(int index, int type)
    {
        return 1;
    }

    int prepare_shutdown(int code)
    {
        return 1;
    }

    int outmsg_recovery(char *data, msg_info_t *info)
    {
        return -1;
    }

    int inmsg_recovery(char *data, msg_info_t *info)
    {
        return -1;
    }

```

## <Makefile>

「[4.2. 同期型インバウンドX25GW](#)」のMakefileの例と同じです。

## 4.3.3. サーバー

### <sndsvr.c>

```

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

SENDSVC(TPSVCINFO *msg)
{
    char *sndbuf;
    long len;

    printf("[%s] Service Started!\n", msg->name);

```

```

len = (msg->len);
printf("len is [%d]\n", len);

if ((sndbuf=(char *)tpalloc("CARRAY", NULL, len)) == NULL) {
    printf("sndbuf alloc failed !\n");
    tpreturn(TPFAIL, -1, NULL, 0, 0);
}

memcpy(sndbuf, msg->data, msg->len);
/* X25GWのサービスをtpforwardで呼び出します。 */
tpforward("X25GW", (char *)sndbuf, len, TPNOREPLY);
}

```

## <rcvsvr.c>

```

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

/* X25GWで動的にこのサービスにtprelayをします。 */
RECVSVC(TPSVCINFO *msg)
{
    char *rcvbuf;
    long len;

    printf("[%s] Service Started!\n", msg->name);

    len = msg->len;
    rcvbuf = msg->data;

    if(tpurcode != 0) {
        printf("tpurcode is [%d] tperrmsg is [%s]\n", tpurcode,
            tpstrerror(tpurcode));
        tpreturn(TPFAIL, -1, (char *)rcvbuf, len, 0);
    }

    tpreturn(TPSUCCESS, 0, (char *)rcvbuf, len, TPNOFLAGS);
}

```

## 4.3.4. クライアント

### <cli\_x25gw.c>

/\* このサンプル・プログラムは、TmaxクライアントとしてSENDSVCを呼び出すようになっています。  
SENDSVCを呼び出す前に、X25GWでtprelayするサービスを構文ヘッダーに設定します。 \*/

```
#include <stdio.h>
#include <usrinc/atmi.h>
#include "../server/custom.h"

int main(int argc, char **argv)
{
    int ret;
    msg_body_t *body;
    char *buf;
    long rlen;

    ret = tmaxreadenv("tmax.env", "TMAX");
    if (ret < 0) {
        printf("tmaxreadenv fail...[%s]\n", tpstrerror(tperrno));
    }

    ret = tpstart((TPSTART_T *)NULL);
    if (ret < 0) {
        printf("tpstart fail...[%s]\n", tpstrerror(tperrno));
        return 0;
    }

    buf = tpalloc("STRING", 0, 0);
    if (buf == NULL){
        printf("buf tpalloc fail...[%s]\n", tpstrerror(tperrno));
        tpend();
        return 0;
    }

    body = (msg_body_t *)buf;
    memcpy(body->data, argv[1], strlen(argv[1]));
    body->data[51] = 0;
    memcpy(body->retsvcname, "RECVSVC", 9);
    body->retsvcname[8] = 0;
    memcpy(buf, (char *)body, sizeof(msg_body_t));

    /* X25GWを呼び出すサービスを呼び出します。 */
    ret=tpcall("SENDSVC",buf,sizeof(msg_body_t), &buf, &rlen, 0);
    if (ret < 0){
        printf("tpcall fail...[%s]\n", tpstrerror(tperrno));
    }
}
```

```
        tpfree((char *)buf);
        tpend();
        return 0;
    }

    body = (msg_body_t *)buf;

    printf("return value = %s\n", body->data);
    tpfree((char *)buf);
    tpend();
}
```



# 付録 A. X25GWのエラーコード

X25GWのエラーコードは下記のとおりです。

エラーコード	説明
TPEINVAL	ゲートウェイ・ヘッダーを使用する際、X25GWを呼び出したデータ長がゲートウェイ・ヘッダーより小さい場合に発生します
TPEPROTO	tpforwardでX25GWを呼び出す際、X25GWが同期型モードではなく非同期型モードの場合に発生します。tpforward&tprelay方式は、必ず同期型方式である必要があります
TPENOREADY	リモートノードとの接続が切断され、使用できるチャンネルが存在しない場合です
TPEOS	X25GW内部でメモリーの確保ができない場合です
TPESYSTEM	ユーザー関数のput_msg_infoで負の数を返した場合、リモートノードにデータを転送する際にエラーが発生します
TPETIME	リモートノードにサービスを要求し、指定されている時間内に応答がない場合です
TPECLOSE	リモートノードにサービスを要求した後、リモートノードとの接続が切断された場合です





# 索引

## C

CLOPT項目(X25GWオプション), 13

## G

get\_channel\_num(), 25

get\_msg\_info(), 24

get\_service\_name3(), 26

## I

init\_remote\_info(), 23

inmsg\_recovery(), 27

## O

outmsg\_recovery(), 27

## P

prepare\_shutdown(), 26

put\_msg\_info(), 25

## R

remote\_closed(), 24

remote\_connected(), 23

## T

Tmax環境ファイル, 11

Tmax環境構成

    CLOPT項目(X25GWオプション), 13

    サービス・ブロック型X25GW, 14

    サービス非ブロック型X25GW, 15

    リモート同期型と非同期型X25GW, 16

Tmax環境構成(CLOPT項目)

    [-a], 14

    [-F], 13

    [-H], 14

    [-h], 14

    [-k], 13

    [-N], 13

    [-n], 13

    [-r], 13

    [-S], 13

    [-t], 13

    [-u], 14

    [-Y], 14

    [-y], 14

Tmax要求の非同期型X25GW, 6

## X

X.25ゲートウェイ, 1

X25GW, 1

X25GWの動作構造, 1

## あ

アウトバウンドX25GWの例, 31

アウトバウンド・サービス, 2

インバウンド・サービス, 1

## か

ゲートウェイ・ヘッダー, 8

## さ

サービス・ブロック型X25GWの環境構成, 14

サービス・ブロック型の同期X25GW, 3

サービス非ブロック型X25GWの例, 43

サービス非ブロック型X25GWの環境構成, 15

サービス非ブロック型の同期X25GW, 4

## た

同期型X25GW, 36

    サービス・ブロック型の同期X25GW, 3

    サービス非ブロック型の同期X25GW, 4

    リモート同期型X25GW, 5

## は

非同期型X25GW

    Tmaxからのサービス要求方式, 6

    非同期型X25GW, 6

リモート要求の非同期型X25GW, 7

## ら

例

同期型インバウンドX25GW, 36

アウトバウンドX25GW, 31

サービス非ブロック型X25GW, 43

リモート同期型X25GW, 5

リモート同期型と非同期型X25GWの環境構成, 16

リモート要求の非同期型X25GW, 7