

TmaxGrid ユーザガイド

Tmax v6.0



Copyright © 2016 TmaxSoft Co., Ltd. All Rights Reserved.

Copyright Notice

Copyright © 2016 TmaxSoft Co., Ltd. All Rights Reserved.

45, Jeongjail-ro, Bundang-gu, Seongnam-si, Gyeonggi-do, 13613, South Korea

Restricted Rights Legend

All TmaxSoft Software (Tmax®) and documents are protected by copyright laws and international convention. TmaxSoft software and documents are made available under the terms of the TmaxSoft License Agreement and this document may only be distributed or copied in accordance with the terms of this agreement. No part of this document may be transmitted, copied, deployed, or reproduced in any form or by any means, electronic, mechanical, or optical, without the prior written consent of TmaxSoft Co., Ltd.

Nothing in this software document and agreement constitutes a transfer of intellectual property rights regardless of whether or not such rights are registered) or any rights to TmaxSoft trademarks, logos, or any other brand features. This document is for information purposes only. The company assumes no direct or indirect responsibilities for the contents of this document, and does not guarantee that the information contained in this document satisfies certain legal or commercial conditions. The information contained in this document is subject to change without prior notice due to product upgrades or updates. The company assumes no liability for any errors in this document.

このソフトウェア(Tmax®)マニュアルの内容とプログラムは、日本国の著作権法および国際条約によって保護されています。マニュアルの内容とプログラムは、TmaxSoft Co., Ltd.との使用許諾契約書の下でのみ使用することができ、マニュアルは使用許諾契約で許可されている範囲を除いては、配布または複製することができません。TmaxSoftの書面による事前の承諾を得ることなく、このマニュアルの全部または一部を電子的または機械的な方法を問わず、転送、複製、配布したり、または二次的著作物を作成する等の行為を一切禁じます。

このソフトウェアのマニュアルとプログラムの使用許諾契約は、いかなる場合においても、マニュアル及びプログラムと関連する知的財産権(登録の有無を問わず)を譲渡するものと解釈されず、TmaxSoftのブランド、ロゴ、商標等の使用権限を与えるものではありません。マニュアルは、情報を提供する目的でのみ提供しており、これに伴う契約上の直接的ないしは間接的な責任を負わず、マニュアルの内容は法律上もしくは商業的な特定の条件が満たされることを保証しません。マニュアルの内容は、製品のアップグレード及び修正により、その内容が予告なく変更されることがあり、内容上の誤りがないことを保証しません。

Trademarks

Tmax®, Tmax WebtoB® and JEUS® are registered trademark of TmaxSoft Co., Ltd. Other products, titles or services may be registered trademarks of their respective companies.

Tmax®, Tmax WebtoB®, JEUS® は、TmaxSoft Co., Ltd.の登録商標です。その他、記載されている会社名、製品名などは、各社の商標または登録商標です。

Open Source Software Notice

Some modules or files of this product are subject to the terms of the following licenses. : openssl-0.9.7.m, zlib-1.1.4, expat-2.0.0, net-snmp, DCE1.0, pthread, google-diff-match-patch, libevent, getopt

Detailed Information related to the license can be found in the following directory :
\${INSTALL_PATH}/license/oss_licenses

この製品の一部ファイルまたはモジュールは、openssl-0.9.7.m、zlib-1.1.4、expat-2.0.0、net-snmp, DCE1.0、pthread、google-diff-match-patch、libevent、getoptライセンスを遵守します。

詳細情報については、製品ディレクトリーの\${INSTALL_PATH}/license/oss_licensesに記載されている事項を参照してください。

文書情報

文書名: TmaxGrid ユーザガイド

発行日: 2016年8月5日

ソフトウェアバージョン: Tmax v6.0

ガイドバージョン: v2.1.1

目次

このガイドについて	vii
第1章 はじめに	1
1.1. 概要	1
1.2. システム構成	3
第2章 使用および管理	5
2.1. 環境設定	5
2.1.1. DOMAIN節	5
2.1.2. NODE節	8
2.2. 状態情報の管理	12
2.3. データのモニタリング	13
2.3.1. tginfo	13
2.3.2. tglockinfo	14
2.4. 制御	15
第3章 API	17
3.1. C言語用のクライアント／サーバーAPI	17
3.1.1. tgsterror	18
3.1.2. tmax_grid_create	18
3.1.3. tmax_grid_create2	20
3.1.4. tmax_grid_destroy	22
3.1.5. tmax_grid_set	23
3.1.6. tmax_grid_get	25
3.1.7. tmax_grid_get2	26
3.1.8. tmax_grid_is_exist	28
3.1.9. tmax_grid_count	29
3.1.10. tmax_grid_lock	30
3.1.11. tmax_grid_unlock	31
3.1.12. tmax_grid_set_watcher	32
3.1.13. tmax_grid_wait_watcher	34
3.1.14. tmax_grid_enqueue	35
3.1.15. tmax_grid_dequeue	37
3.1.16. tmax_grid_dequeue2	38
3.1.17. tmax_grid_get_children	40
3.1.18. tmax_grid_get_child_with_index	41
3.1.19. tmax_grid_keylist_free	42
3.1.20. tmax_grid_init_key_browser	42
3.1.21. tmax_grid_get_next	43
3.1.22. tmax_grid_init_bulk_handle	44
3.1.23. tmax_grid_set_bulk_handle	45
3.1.24. tmax_grid_destroy_bulk_handle	46

3.1.25.	tmax_grid_bulk_set	47
3.1.26.	tmax_grid_bulk_get	48
3.2.	Java言語用のクライアント／サーバーAPI	50
3.2.1.	gq2Create	50
3.2.2.	gq2Destroy	51
3.2.3.	gq2Set	51
3.2.4.	gq2Get	52
3.2.5.	gq2IsExist	52
3.2.6.	gq2Count	52
3.2.7.	gq2Lock	53
3.2.8.	gq2Unlock	53
3.2.9.	gq2SetWatcher	54
3.2.10.	gq2Enqueue	54
3.2.11.	gq2Dequeue	55
3.2.12.	gq2GetChildren	55
第4章	例題	57
4.1.	C言語のクライアント	57
4.2.	C言語のサーバー	63
4.3.	Java言語のクライアント	70
索引	75

このガイドについて

対象読者

本書は、TmaxGrid[®]を使用してプログラムを開発しようとするユーザーやシステム管理者を対象に記述しています。

前提知識

本書はTmaxシステムへの知識をベースにし、Tmaxシステムが提供する各種の機能および特徴が習得できるように作成されました。

したがって、本書の内容を円滑に理解するには、以下の事項を事前に熟知している必要があります。

- ミドルウェアおよびUNIX系に対する基本的な理解
- Tmaxに対する基本的な理解
- JavaやCに対する基本的な理解

本書の構成

TmaxGridガイドは計5つの章と付録で構成されています。

各章の主要内容は以下のとおりです。

- 第1章: はじめに

TmaxGridの基本概念と特徴、そして構成要素について説明します。

- 第2章: 使用および管理

TmaxGridを運用するための設定項目と状態のモニタリング方法について説明します。

- 第3章: API

TmaxGridを利用して開発することができる、CやJAVA言語用のAPIについて説明します。

- 第4章: 例題

C言語のクライアントやサーバーとJAVA言語のクライアントの例題です。

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	参照事項
注	注意事項
[図 1.1]	図の名称
[表 1.1]	表の名称
AaBbCc123	Javaコード、XMLドキュメント
[<i>command argument</i>]	オプション・パラメータ
< xyz >	「<」と「>」の間の内容は実際に使用される特定の名称または値で置き換えられる
	構文の中の相互に排他的な選択項目の選択肢を示す
...	パラメータ、値、または他の情報が繰り返される

システム要件

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

お問合せ先

Korea

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

USA

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

Japan

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

China

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

Brazil

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

Russia

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

Singapore

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

United Kingdom

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

Canada

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

Australia

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

India

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

Turkey

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

第1章 はじめに

本章では、TmaxGridの基本概念と特徴、そして構成要素について説明します。

1.1. 概要

TmaxGridは、マルチ・ノード環境上でインメモリ・データの同期を取るために使います。簡単なAPIを実装するだけで、多数の業務プログラムが、分散ロックやキュー、共有機能などの機能をマルチ・ノード上で使用できます。

TmaxGridは、大きくサーバー、クライアント、およびTmaxGridサーバーで構成されています。サーバーとクライアントとは、TmaxGridを使用する業務プログラムのことをいいます。

- サーバー、クライアント

TmaxGridのAPIを利用する業務プログラムです。TmaxGridのAPIを実装するだけで、TmaxGridが利用できます。

- TmaxGridサーバー

Tmaxシステム・サーバーで構成します。データは共有メモリーに保存し、要求はTmaxエンジン(CLH)とサーバーが処理します。クラスターが構成されている場合は、TmaxGridサーバー間で直接通信を行います。同じマシン上に存在するプロセス間ではUNIXドメイン・ソケットを使用します。一方、TmaxGridサーバー間ではUDPソケットを使用して通信を行います。(パフォーマンス上の利点有り)

提供機能は以下のとおりです。

- 分散ロック

特定のキーを利用し、マルチ・ノード、マルチ・プロセス上で排他ロックを構成することができます。

- ウォッチャー(Watcher)

コールバック関数を登録すると、特定キーのデータに関連するイベントが発生したら、その通知を受けることができます。

- データ制御

マルチ・ノード上で同じキーのデータを入力、削除、および修正することができます。

- 障害/復旧

他のノードのシステムに障害が発生したら、それを検出して適切な対応をすることができます。そして、運用中にシステムが復旧されると、そのシステムは自動的にフェイル・バックされます。

- 管理

状態モニタリングとデータ・モニタリングを行うユーティリティを提供します。設定は、静的／動的に変更可能です。

特徴

以下は、TmaxGridの主な特徴です。

- キー・ベース・キュー

すべてのデータを「key:value(キー:値)」のペアで保持しています。

キーは以下の条件を満足しなければなりません。

- C言語ではchar *型であり、Java言語ではString型です。
- 127バイト以内で表現します。
- 「/」文字を使って、親キーと子キーを区分します。

TmaxGridは親子関係のキー構造をサポートします。つまり、UNIXのファイル・システムのように構造化してキーを設定することができます。

たとえば、為替情報を表現する場合、以下のように人が認識可能な論理的な構造化ができます。

```
/ExchangeRate/Korea/USA, /ExchangeRate/Korea/Japan
```

「ExchangeRate」の子キーは「Korea」あり、「Korea」の子キーは「USA」です。また、「ExchangeRate」配下の変化に関連する業務は、「ExchangeRate」というキーさえ分かれば、子キーにおける変化を検出することができます。値としては、tpallocで割り当てたデータのみを認めます。

- クラスタ方式

変化があれば、すべてのノードに複製を行い、ノード全体が同じメモリー空間を共有できるようにします。つまり、すべてのノードの「キー:値」のデータは同じでなければなりません。一方、クラスタ方式のシステムは、マルチ・ノード上で全体メンバーの中で正常状態にあるノードが半数以下であれば、システムが孤立しているか正常なのかが把握できないため、動作をしません。

もし、ノードが2台だとすれば、2つのノードが両方とも常に動作していないと、このシステムは動作しません。5台であれば、3台まで動作しないと正常動作しません。2台が相互につながっているとしても孤立していると判断し、処理を続行しません。

このような動作方式を取るのは、すべてのノードが同じ値を持つようにするためです。あるノードに障害が発生した後、再度それが回復された場合、既に動作しているシステムがあれば、進行中のキー:値の複製が行われ、同じTmaxGridを保持するようにします。

それぞれのノードは、最近の変更内容を保存する履歴機能を持ちます。マスター・ノードは、あるノードから復旧要求を受信したら、まず、そのノードのシーケンス番号を確認して自分の履歴内容を検索します。その後、履歴の変更内容を送信してそのノードを回復させます。もし、そのノードのシーケンス番号が自分の履歴にない場合には、全体のキー:値のデータを送信してノードを回復させます。

● モニタリングおよび管理機能のサポート

tmaxadminの追加コマンドを使い、現在の設定値と状態をモニタリングすることができます。また、簡単な制御機能も提供しています。設定値とは、環境設定ファイルに適用した値のことです。現在のGQSの状態と各キーの状態を確認することができます。制御機能としては、GQS(Global Queue Server)がクライアントの命令を実行せずに停止していただける「suspend」機能を提供しています。

制約事項

以下は、TmaxGridサービスの制約事項についての説明です。

- 1つのドメイン内でだけ有効な機能です。マルチ・ドメイン上で使用することはできません。
- 1つのドメイン内でこの機能を活性化させると、すべてのノードは、必ずこの機能を使用しなければなりません。一部のノードだけこの機能を使用することはできません。
- WindowsはGQSを提供しません。
- UDPマルチキャストを使用しているので、ノード間でUDPマルチキャストが可能なシステムでなければなりません。
- UDPマルチキャストは、ネットワークの負荷を増加させかねないので、別途のネットワークで構成することを推奨します。
- 一つの値は10,000バイトを超えることができません。

1.2. システム構成

TmaxGridシステムは、キー／値のデータを保存するシステム保存域、マルチ・ノード間でTmaxGridの同期を実行するサーバー・プロセスであるGQS、そして実際の機能を担当するサーバー／クライアント・ライブラリーの3つの要素で構成されます。

● システム保存域

TmaxGridを保存できる保存域です。管理者が設定したUNIXシステムの共有メモリーに、以下の形式で情報を保存します。

```
key:value
```

TmaxGridの情報を保存する際に、その他の必要な情報も一緒に保存します。実際に保存可能なキーの数や値のサイズは設定にて決まります。実際には、マシンが提供する共有メモリーのサイズまでも設定可能です。

マルチ・ノードの場合、最も小さい値に合わせます。

- **GQS(Global Queue Server)**

Tmaxのシステム・プロセスです。一つのノードに一つのプロセスだけが起動します。CLHとサーバー・プロセスは、他のGQSとの通信を行います。このプロセスは、tmbootによってノード全体が起動するときと一緒に起動し、全体がtmdownされるときと一緒にダウンされます。

- **サーバー/クライアント・ライブラリー**

C言語用の一般クライアント・ライブラリーとスレッド用のクライアント・ライブラリー、そしてTCS、UCS用のサーバー・ライブラリーを提供します。また、Java言語用のWebTライブラリーも提供します。

このライブラリーを利用し、開発者は、キーのcreate、destory、put、get、update、delete、enqueue、dequeue、およびwatcherの登録を行うことができます。

第2章 使用および管理

本章では、TmaxGridを運用するための設定項目と状態のモニタリング方法について説明します。

2.1. 環境設定

TmaxGrid機能を活性化させるには、Tmax環境設定のDOMAIN節とNODE節を構成する必要があります。

2.1.1. DOMAIN節

以下は、DOMAIN節の設定項目についての説明です。

```
*DOMAIN
domain
:
TGSHMSIZE = numeric,
TGMAX     = numeric,
TGMAX_CHILD = numeric ,
TGMAX_WATCHER = numeric,
TGHEARTBEAT = numeric,
TGTIMEOUT  = numeric,
TGLOGLVL  = string,
TGSHMKEY = numeric,
TGPORTNO  = numeric,
TGMCAST_IP = literal,
TGHISTORY = numeric,
TGSTANDBY = numeric,
TGMAXBUFFERSIZE = numeric,
TGDOWNWAITTIME = numeric
```

必須項目

- TGSHMSIZE = numeric
 - 範囲:32768〜262143
 - キー情報や値、管理情報を保存する共有メモリーの最大サイズです。
- TGMAX = numeric

- 範囲 : 1 ~ MAX_INT
- 最大で作成可能なキーの数です。
- TGMAX_CHILD = numeric
 - 範囲 : 1 ~ MAX_INT
 - 一つのキーが作成可能な最大の子キーの数(直系の子のみ)です。このキーの数を、TGMAXの数より大きく設定することはできません。
- TGMAX_WATCHER = numeric
 - 範囲 : 1 ~ MAX_INT
 - クラスター上の一つのTmaxGridサーバーに最大で作成可能なウォッチャーの数です。
- TGHEARTBEAT = numeric
 - 範囲 : 1 ~ MAX_INT
 - クラスター上でのハートビート(heartbeat)のチェック周期です。(単位:ms)
- TGTIMEOUT = numeric
 - 範囲 : 1 ~ MAX_INT
 - クラスター上の他のGQSから要求がない場合に、障害が発生したと判断する時間(障害検出待ち時間)です。(単位:ms)
- TGLOGLVL = string
 - 範囲:TMMLOGLVLと同じ
 - GQSのログレベルを設定します。ログレベルはTMMLOGLVL項目で設定した値と同じ方式で指定可能です。DEBUG1~DEBUG4の場合、GQSの実行ファイルを、デバッグログを出力するgqsdに交替したら、付加的なデバッグログを出力します。
- TGSHMKEY = numeric
 - 範囲:32768~262143
 - キー情報や値、管理情報を保存する共有メモリーのキーです。

- TGPORNO = numeric
 - クラスタ上のあるGQSが他のGQSに送信するデータを受信するポート番号です。GQSはUDPマルチキャストとUDPユニキャストを使用するため、ポートも2つを使用します。
 - GQPORNOはUDPマルチキャストのためのポート番号です。GQPORNO + 1は、ユニキャストのために使用します。
- TGMCAST_IP = literal
 - クラスタ上でGQSが他のGQSに送信するためのマルチキャストIPです。
 - UDPマルチキャストを使用するため、IPは標準にしたがって224.0.1.0〜239.255.255.255から選択する必要があります。

選択項目

- TGHISTORY = numeric
 - 範囲: 1〜MAX_INT、デフォルト: 50
 - 迅速な復旧のために、クラスタ上でGQSがメモリーに保存している同期要求の数です。
- TGSTANDBY = numeric
 - 範囲: 1〜MAX_INT、デフォルト: 50
 - クラスタ上でマスターではないGQSからマスターであるGQSに同時に要求を送る場合、最大で受信可能な要求の数です。
- TGMAXBUFFERSIZE = numeric
 - 範囲: 1〜MAX_INT、デフォルト: 65000
 - GQS間で回復のため、大容量サイズのデータを送信するときに、分けて送信できる単位です。回復のためのデータのサイズがこの値より大きい場合は、このサイズずつ分けて送信します。(単位: Byte)
- TGDOWNWAITTIME = numeric
 - 範囲: 1〜MAX_INT、デフォルト: 30

- メモリー不足を理由にリポートをしようとする場合、メモリー不足の状態を認知してからリポートするまでの待機時間です。(単位:秒)

使用例

以下は、DOMAIN節の設定例です。以下の例で、イタリック体で設定されている部分は、各マシンに合わせて入力する必要があります。

```
*DOMAIN
domain
  SHMKEY      = @@DOM1_NA.SHMKEY@@,
  MAXUSER     = 10000,
  MINCLH      = 1, MAXCLH = 1,
  CLHLOGLVL   = DEBUG4, TMMLOGLVL=DEBUG4,
  TPORTNO     = @@DOM1_NA.PORTNO@@,
  BLOCKTIME   = 30,
  MAXSACALL   = 1024,    MAXCACALL = 1024,
  TGSHMSIZE    = 40960, TGMAX = 10000,
  TGMAX_CHILD = 100,    TGMAX_WATCHER = 10,
  TGHEARTBEAT = 1000,   TGTIMEOUT = 5000,
  TGLOGLVL    = DEBUG3,
  TGPORTNO    = @@DOM1_NA.GQPORTNO@@, TGMCAST_IP="224.0.0.101",
  TGHISTORY   = 100, TGSTANDBY = 10,
  TGMAXBUFFERSIZE = 1000, RACPORT=11111
```

2.1.2. NODE節

以下は、NODE節の設定項目についての説明です。

```
*NODE
:
  TGSHMKEY = numeric,
  TGID     = numeric,
  TGMCAST_TTL = numeric,
  TGMCAST_IF = string,
  TGLOGLVL  = string
```

必須項目

- TGID = numeric
 - 範囲 : 1 ~ MAX_INT

- マスターを選定する際に、同じシーケンス番号を持つGQSの間で加重値を与えるための項目です。
- TGID番号が高いとマスターになれます。すべてのGQSはGQIDが異なる必要があります。
- TGMCAST_TTL = numeric
 - 範囲 : 1 ~ MAX_INT
 - クラスタ上での通信時にUDPマルチキャストを使用するため、マルチキャスト・グループへのメンバーシップ加入が必要です。加入時にはルーティングの範囲を指定する必要があります。すべてのマシンの間で通信が可能な値を設定します。

たとえば、以下のように設定します。

区分	説明
0	ホストの内部
1	同じサブネット
<32	同じオンサイト(団体や部署)
<64	同じ地域
<128	同じ大陸

選択項目

- TGSHMKEY = numeric
 - 範囲 : 32768 ~ 262143
 - キー情報や値、管理情報を保存する共有メモリーのキーです。
 - NODE節に設定すると、マシンごとに異なるキーが持てるように設定可能です。
- TGMCAST_IF = string
 - GQS間で通信するインターフェースを指定します。指定しない場合、OSのルーティング・テーブルによって動作を実行します。
- TGLOGLVL = string
 - 範囲 : TMMLOGLVLと同じ

- GQSのログレベルを設定します。ログレベルはTMMLOGLVL項目で設定した値と同じ方式で指定可能です。DEBUG1~DEBUG4の場合、GQSの実行ファイルを、デバッグログを出力するgqsdに交替したら、付加的なデバッグログを出力します。

使用例

以下は、3台のノードにGQSを設定する例です。以下の例で、イタリック体で設定されている部分は、各マシンに合わせて入力する必要があります。

```
*DOMAIN
domain
  SHMKEY      = @@DOM1_NA.SHMKEY@@,
  MAXUSER     = 10000,
  MINCLH      = 1, MAXCLH = 1,
  CLHLOGLVL   = DEBUG4, TMMLOGLVL=DEBUG4,
  TPORTNO     = @@DOM1_NA.PORTNO@@,
  BLOCKTIME   = 30,
  MAXSACALL   = 1024,    MAXCACALL = 1024,
  TGSHMSIZE    = 40960, TGMAX = 10000,
  TGMAX_CHILD  = 100,    TGMAX_WATCHER = 10,
  TGHEARTBEAT  = 1000,  TGTIMEOUT = 5000,
  TGLOGLVL    = DEBUG3,
  TGPORTNO    = @@DOM1_NA.GQPORTNO@@, TGMCAST_IP="224.0.0.101",
  TGHISTORY    = 100, TGSTANDBY = 10,
  TGMAXBUFFERSIZE = 1000, RACPORT=11111

*NODE
DOM1_NA
  TMAXDIR     = " @@DOM1_NA.TMAXDIR@@ ",
  APPDIR      = " @@DOM1_NA.TMAXDIR@@/appbin ",
  PATHDIR     = " @@DOM1_NA.TMAXDIR@@/path ",
  TLOGDIR     = " @@DOM1_NA.TMAXDIR@@/log/tlog ",
  ULOGDIR     = " @@DOM1_NA.TMAXDIR@@/log/ulog ",
  SLOGDIR     = " @@DOM1_NA.TMAXDIR@@/log/slog ",
  CLHOPT      = "-o @@DOM1_NA.TMAXDIR@@/log/slog/clh.log
-e @@DOM1_NA.TMAXDIR@@/log/slog/clh.log ",
  TMMOPT      = "-o @@DOM1_NA.TMAXDIR@@/log/slog/tmm.log
-e @@DOM1_NA.TMAXDIR@@/log/slog/tmm.log ",
  HOSTNAME    = @@DOM1_NA.TMAXDIR@@,
  TGSHMKEY    = @@DOM1_NA.TMAXDIR@@,
  TGID        = 1,  MAXRSTART = 1,
  IP          = " @@DOM1_NA.V4_IP@@ ",
  TPORTNO     = @@DOM1_NA.PORTNO@@
```



```

DOM1_NB
TMAXDIR = "@@DOM1_NB.TMAXDIR@@",
APPDIR = "@@DOM1_NB.TMAXDIR@@/appbin",
PATHDIR = "@@DOM1_NB.TMAXDIR@@/path",
TLOGDIR = "@@DOM1_NB.TMAXDIR@@/log/tlog",
ULOGDIR = "@@DOM1_NB.TMAXDIR@@/log/ulog",
SLOGDIR = "@@DOM1_NB.TMAXDIR@@/log/slog",
CLHOPT = "-o @@DOM1_NB.TMAXDIR@@/log/slog/clh.log
-e @@DOM1_NB.TMAXDIR@@/log/slog/clh.log ",
TMMOPT = "-o @@DOM1_NB.TMAXDIR@@/log/slog/tmm.log
-e @@DOM1_NB.TMAXDIR@@/log/slog/tmm.log ",
HOSTNAME = @@DOM1_NB.NODENAME@@,
TGSHMKEY = @@DOM1_NB.GQSHMKEY@@,
TGID = 2,
MAXRSTART = 1,
IP="@@DOM1_NB.V4_IP@@",
TPORTNO=@@DOM1_NB.PORTNO@@

DOM1_NC
TMAXDIR = "@@DOM1_NC.TMAXDIR@@",
APPDIR = "@@DOM1_NC.TMAXDIR@@/appbin",
PATHDIR = "@@DOM1_NC.TMAXDIR@@/path",
TLOGDIR = "@@DOM1_NC.TMAXDIR@@/log/tlog",
ULOGDIR = "@@DOM1_NC.TMAXDIR@@/log/ulog",
SLOGDIR = "@@DOM1_NC.TMAXDIR@@/log/slog",
CLHOPT = "-o @@DOM1_NC.TMAXDIR@@/log/slog/clh.log
-e @@DOM1_NC.TMAXDIR@@/log/slog/clh.log ",
TMMOPT = "-o @@DOM1_NC.TMAXDIR@@/log/slog/tmm.log
-e @@DOM1_NC.TMAXDIR@@/log/slog/tmm.log ",
HOSTNAME = @@DOM1_NC.NODENAME@@,
TGSHMKEY = @@DOM1_NC.GQSHMKEY@@,
TGID = 3,
MAXRSTART = 1,
IP = "@@DOM1_NC.V4_IP@@",
TPORTNO = @@DOM1_NC.PORTNO@@

*SVRGROUP
svg1 NODENAME = DOM1_NA, COUSIN = "svg2,svg3"
svg2 NODENAME = DOM1_NB
svg3 NODENAME = DOM1_NC

*SERVER
svr1 SVGNAME = svg1, LOGLVL=DEBUG4, CLOPT="-o $(SVR).$(SPRI).out
-e $(SVR).$(SPRI).out"

```

```
*SERVICE
ECHO SVRNAME = svr1
TEST_GQS_BASE SVRNAME = svr1
```

2.2. 状態情報の管理

tmaxadminの以下のコマンドを使って設定と状態を照会します。

```
st -tg
```

- 設定情報

コマンドを実行すると、以下の項目を確認することができます。

項目	説明
shmkey	環境設定のTGSHMKEY項目です
shmsize	環境設定のTGSHMSIZE項目です
max data	環境設定のTGMAX項目です
max watcher	環境設定のTGMAX__WATCHER項目です
id	環境設定のTGID項目です
portno	環境設定のTGPORTNO項目です
heartbeat	環境設定のTGHEARTBEAT項目です
timeout	環境設定のTGTIMEOUT項目です
historycount	環境設定のTGHISTORY項目です
standbycount	環境設定のTGSTANDBY項目です
gqmaxbuffersize	環境設定のTGMAXBUFFERSIZE項目です
gqdownwaittime	環境設定のTGDOWNWAITTIME項目です

- 状態

項目	説明
status	TmaxGridサーバーの現在の状態です – RUN: 正常状態(サービス可能状態) – NOT_READY: 起動していない状態 – READY: 起動はしたものの、同期作業やマスター選定作業が進行中
suspend status	gqsuspendコマンドによって停止された状態なのかどうかを確認できます – NOT_SUSPENDED: gqsuspendを実行していない状態

項目	説明
	– SUSPENDED: ggsuspendコマンドによって停止された状態
created node count	現在まで作成されたノードの数です
deleted ndoe count	現在まで削除されたノードの数です
set data count	現在までsetでデータが入力されたノードの数です
get data count	現在までgetでデータを取得したノードの数です
current node count	現在作成されているノードの数です
temporary node count	現在作成されているノードのうち、Temporaryタイプのノードの数です
watcher count	現在作成されているウォッチャーの数です
current data count	現在入力されているデータの数です
total data size	現在入力されているすべてのデータのサイズの合計です

2.3. データのモニタリング

本節では、データをモニタリングするためのコマンドについて説明します。

2.3.1. tginfo

tmadminのgqiコマンドを使ってTmaxGridが管理しているデータを確認することができます。

- 使用方法

```
$ $1 tmax1 (tmadm): tginfo (tgi)
```

- 出力情報

各キーから以下の情報を確認することができます。

項目	説明
name	キーの名前です
stime	キーが生成された時間です
node_type	キーのタイプです <ul style="list-style-type: none"> – Persistent: キーを作成したクライアントが終了しても削除されないキー – Temporary: 作成したクライアントが終了すると削除されるキー – Queue: enqueue APIによって作成したキー – Queue Child: enqueueで自動生成されるキー

項目	説明
	<ul style="list-style-type: none"> – Lock : Lock APIによって作成したキー – Lock Child: Lock APIによって自動生成されるキー
data_type	保持している値のタイプ (STRING、CARRAY、STRUCT、FIELD) です
datalen	保持している値の長さです
watcher	当該キーにウォッチャーを登録したクライアントの数です
child	当該キーの子キーの数 (すぐ配下のレベルの子キーの数だけ確認可能) です
create	キーを作成したクライアントIDです
get	キーのデータを最後に取得したクライアントIDです
last	最後にアクションを実行したクライアントIDです
lock	ロックの状態を表示します

2.3.2. tglockinfo

tmaxadminのtglockinfo(tgli)コマンドを使ってTmaxGridが管理しているロック情報を確認することができます。

● 使用方法

```

$ $1 tmax1 (tmaxadm): tglockinfo (tgli) [ -o [ソートする条件] ] [ -k [キーの名前] ]
[ -c [子キーの数] ] [ -d [待機時間 (秒)] ]

```

項目	説明
[-o [ソートする条件]]	<p>[-o]オプションは、特定条件を基準にして照会結果をソートして出力するためのオプションです。ソートは降順と昇順の両方で設定可能です</p> <ul style="list-style-type: none"> – [-o st]: キーが生成された時間の降順 – [-o st-]: キーが生成された時間の昇順 – [-o ch]: 子キーの数の降順 – [-o ch-]: 子キーの数の昇順 – [-o du]: 待機時間の降順 – [-o du-]: 待機時間の昇順
[-k [キーの名前]]	当該キーの情報を照会します。当該キーの子キーまで表示されます
[-c [子キーの数]]	子キーの数が指定した値以上であるキーに対して照会を行います。この際、条件にマッチするキーの子キーも一緒に表示されます

項目	説明
[-d [待機時間(秒)]]	待機時間が入力した値以上であるキーを照会します

- 出力情報

以下の情報を確認することができます。

項目	説明
name	キーの名前です
stime	キーが生成された時間です
acquired	ロックを取得した時間です
duration	ロックを保持している時間です
node_type	キーのタイプです <ul style="list-style-type: none"> – Lock : Lock APIによって作成したキー – Lock Child: Lock APIによって自動生成されるキー
child	当該キーの子キーの数(すぐ配下のレベルの子キーの数だけ確認可能)です
create	キーを作成したクライアントIDです
lock	ロックの状態を表示します

2.4. 制御

tmadminのtgsuspend、tgresume、tgpurgeコマンドを使って制御します。

- tgsuspend

運用中のTmaxGridサーバーのサービスを停止します。クライアントの要求を受信はしないものの、データの同期は取ります。

```
> tgsuspend
```

- tgresume

tgsuspendで停止したTmaxGridサーバーがクライアントの要求を再度受信するようにします。

```
> tgresume
```

- tgpurge

TmaxGridのすべてのキーと値を削除して初期化します。

```
> tgpurge
```

第3章 API

本章では、TmaxGridを利用して開発することができる、CやJAVA言語用のAPIについて説明します。

3.1. C言語用のクライアント／サーバーAPI

以下は、C言語用のクライアント／サーバーAPIについての説明です。

API	説明
tgsterror	TmaxGrid APIを使用するときに発生するggqerrno番号のメッセージを出力します
tmax_grid_create	キーを作成します
tmax_grid_create2	キーを作成し、キーの値を設定します
tmax_grid_destory	キーを削除します
tmax_grid_set	キーに値を設定します
tmax_grid_get	値を取得し、当該キーの値は削除します
tmax_grid_get2	プリミティブ型のAPIです。データ型と一緒に値を取得し、当該キーの値は削除します
tmax_grid_is_exist	キーが存在するかどうかをチェックします
tmax_grid_count	全体のキーの数を照会します
tmax_grid_lock	キーの名前でロックを実行します
tmax_grid_unlock	キーの名前でロックを解除します
tmax_grid_set_watcher	当該キーのイベントが発生する際に呼び出す関数を登録します
tmax_grid_wait_watcher	イベントが発生するまで「timeout」に指定した時間の間、待機します
tmax_grid_enqueue	キーの名前でデータを入力します
tmax_grid_dequeue	tmax_grid_enqueue()で入力した値の中で最初の値を照会します
tmax_grid_dequeue2	プリミティブ型のAPIです。tmax_grid_enqueue()で入力した値の中で最初の値をデータ型とともに照会します
tmax_grid_get_children	キーの名前を使い、子キーの名前のリストを照会します
tmax_grid_get_child_with_index	子キーの情報を格納しているgrid_KEYLIST_Tハンドラーからn番目のキーの情報をgrid_KEYINFO_Tの構造体に保持します
tmax_grid_keylist_free	子キーの情報を格納しているgrid_KEYLIST_Tハンドラーのリソースを解放します

API	説明
tmax_grid_init_key_browser	キーをブラウズするためのAPIです。当該キーとキーの長さを使ってブラウザーを初期化します
tmax_grid_get_next	ブラウズをして次のキーを取得します
tmax_grid_init_bulk_handle	複数のキー／値のペアをゲット(get)／セット(set)するためにハンドラーを初期化します
tmax_grid_set_bulk_handle	一ペアのキー、またはキー／値をハンドラーに登録します
tmax_grid_destroy_bulk_handle	ハンドラーのリソースを解放して戻します
tmax_grid_bulk_set	ハンドラーに登録されているすべてのキー／値のペアを設定します
tmax_grid_bulk_get	ハンドラーに登録されているすべてのキーの値を取得し、そのキーの値は削除します

3.1.1. tgstrerror

TmaxGrid APIを使用時に発生するtgerrnoに該当する番号のメッセージを出力します。

- プロトタイプ

```
#include <usrinc/tmaxapi.h>
char *tgstrerror(int tgerrno)
```

- パラメータ

パラメータ	説明
tgerrno	出力するエラー番号です

- 戻り値

戻り値	説明
エラー・メッセージ	エラーコードのメッセージがある場合です

3.1.2. tmax_grid_create

キーを作成します。キーを作成するときに、キー・タイプを指定する必要があります。

作成したノードに対してenqueue、dequeue、またはLock、Unlockを使用するには、TG_QUEUEまたはTG_LOCKタイプを指定する必要があります。当該タイプで作成されたノードには、enqueue、dequeue、Lock、Unlock APIを使用する方法でのみ子ノードを作成することができます。さらに、これらのタイプは常に

TG_PERSISTENTタイプで作成されます。TG_TEMPORARYタイプで作成されたノードの子ノードを作成するときは、TG_PERSISTENTタイプで作成してはなりません。

注

TG_TEMPORARYタイプのノードを削除すると、すべての子ノードが一緒に削除されるので注意が必要です。

● プロトタイプ

```
#include <usrinc/tmaxapi.h>
int tmax_grid_create(char *key, int keylen, int flags)
```

● パラメータ

パラメータ	説明
key	作成するキーの名前です。127バイトを超えることはできません
keylen	キーのサイズです
flags	動作方式を指定します – TG_PERSISTENT クライアントが終了しても削除されません – TG_TEMPORARY クライアントが終了する場合やタイムアウトが過ぎた場合に自動で削除されます – TG_QUEUE enqueue、dequeueを使用するキーを作成します – TG_LOCK Lock、Unlockを使用するキーを作成します

● 戻り値

戻り値	説明
>= 0	関数の呼び出しに成功した場合です
< 0	関数の呼び出しに失敗した場合です。tgerrnoにエラーコードが設定されます

● エラー

tmax_grid_create()が正常に実行されなかった場合、tgerrnoに以下の値のいずれかが設定されます。

エラーコード	説明
[TGEINVAL]	キーがNULLであるか、key_len <=0または128バイト以上である場合、あるいはフラグに適切でない値が指定された場合に発生します
[TGEOS]	Tmaxシステムにエラーが発生した場合です。詳細情報はログファイルに記録されます
[TGENOMEM]	メモリーが不足して割り当てに失敗した場合に発生します
[TGEKEY]	キーを正常に認識できない場合に発生します
[TGEMAXCHL]	環境設定のMax Child制限に達して子キーを作成できなくなった場合に発生します
[TGEPROTO]	TG_QUEUEまたはTG_LOCKタイプのキーの子キーを作成した場合に発生します
[TGESHMFULL]	TmaxGridの共有メモリー空間が不足したり、あるいは環境設定のTGMAXの最大値までキーが作成されてキーが作成できなくなった場合に発生します
[TGEDUPKEY]	キーの名前が重複する場合です
[TGECONF]	TmaxGridが設定されていない場合です
[TGEENABLE]	TmaxGridが動作しない場合です

3.1.3. tmax_grid_create2

キーを作成します。キーを作成するときに、値を入力し、Watcherも一緒に登録することができます。詳細については、[tmax_grid_create](#)、[tmax_grid_set](#)、[tmax_grid_set_watcher](#) APIを参照してください。データがNULLでない場合に値を入力します。コールバック関数がNULLでない場合にWatcherが設定されます。

● プロトタイプ

```
#include <usrinc/tmaxapi.h>
int tmax_grid_create2(char *key, int keylen, char *data, int len,
TG_WATHCER_CALLBACK callback, void *args, int flags)
```

● パラメータ

パラメータ	説明
key	作成するキーの名前です。127バイトを超えることはできません
keylen	キーのサイズです
data	値です。tpallocで割り当てられたポインターです。NULLの場合は、値を保存せずにキーだけ作成します
len	データのサイズです

パラメータ	説明
callback	Watcherを登録後、イベントが発生したときに呼び出すコールバック関数のポインターです
args	コールバック関数を呼び出す場合に渡す引数のポインターです
flags	<p>動作方式を指定します</p> <ul style="list-style-type: none"> – TG_PERSISTENT クライアントが終了しても削除されません – TG_TEMPORARY クライアントが終了する場合、自動で削除されます – TG_QUEUE enqueue、dequeueを使用するキーを作成します – TG_LOCK Lock、Unlockを使用するキーを作成します – TG_WATCHER_PERSISTENT Watcherを登録した場合、クライアントに通知を送り続けます。キーが削除されたら、Watcherは自動で削除されます – TG_WATCHER_ONCE Watcherを登録した場合、一度だけ通知し、以降は通知を送りません。もし情報を継続して受信したい場合は、Watcherを再登録する必要があります – TG_EVENT_DELETE_SELF 自身のキーが削除されたときにイベントを受信します。コールバック関数で当該イベントを受信した場合、登録されたWatcherは削除されるので、Watcherを再登録する必要があります – TG_EVENT_CREATE_CHILD 子キーが作成されたときにイベントを受信します – TG_EVENT_SET_SELF 自身のキーにデータが設定されたときにイベントを受信します – TG_EVENT_GET_SELF 自身のキーにデータが取得されたときにイベントを受信します

- 戻り値

戻り値	説明
≥ 0	関数の呼び出しに成功した場合です
< 0	関数の呼び出しに失敗した場合です。tgerrnoにエラー・コードが設定されます

- エラー

tmax_grid_create2()が正常に実行されなかった場合、tgerrnoに以下の値のいずれかが設定されます。

エラーコード	説明
[TGEINVAL]	キーがNULLであるか、key_len ≤ 0 または128バイト以上である場合、あるいはフラグに適切でない値が指定された場合に発生します
[TGEOS]	Tmaxシステムにエラーが発生した場合です。詳細情報はログファイルに記録されます
[TGENOMEM]	メモリーが不足して割り当てに失敗した場合に発生します
[TGEKEY]	キーを正常に認識できない場合に発生します
[TGEMAXCHL]	環境設定のMax Child制限に達して子キーを作成できなくなった場合に発生します
[TGEPROTO]	TG_QUEUEまたはTG_LOCKタイプのキーの子キーを作成した場合に発生します
[TGESHMFULL]	TmaxGridの共有メモリー空間が不足したり、あるいは環境設定のTGMAXの最大値までキーが作成されてキーが作成できなくなった場合に発生します
[TGEDUPKEY]	キーの名前が重複する場合です
[TGECONF]	TmaxGridが設定されていない場合です
[TGEENABLE]	TmaxGridが動作しない場合です

3.1.4. tmax_grid_destroy

キーを削除します。値も一緒に削除されます。基本的に子キーが存在すれば削除されませんが、フラグを設定すると、persistent typeに限って子キーまで一緒に削除することができます。またキーが削除されるときに、Watcherが登録されている場合には、設定されたイベントが通知された後Watcherも一緒に削除されます。

- プロトタイプ

```
#include <usrinc/tmaxapi.h>
int tmax_grid_destroy(char *key, int keylen, int flags)
```

- パラメータ

パラメータ	説明
key	作成するキーの名前です。127バイトを超えることはできません
keylen	キーのサイズです
flags	動作方式を指定します <ul style="list-style-type: none"> – TG_CHILDREN - 子キーも一緒に削除します

- 戻り値

戻り値	説明
≥ 0	関数の呼び出しに成功した場合です
< 0	関数の呼び出しに失敗した場合です。tgerrnoにエラーコードが設定されます

- エラー

tmax_grid_destroy()が正常に実行されなかった場合、tgerrnoに以下の値のいずれかが設定されます。

エラーコード	説明
[TGEINVAL]	キーがNULLであるか、key_len ≤ 0 または128バイト以上である場合、あるいはフラグに適切でない値が指定された場合に発生します
[TGEOS]	Tmaxシステムにエラーが発生した場合です。詳細情報はログファイルに記録されます
[TGENOMEM]	メモリーが不足して割り当てに失敗した場合に発生します
[TGEKEY]	キーを正常に認識できない場合に発生します
[TGESHMFULL]	TmaxGridの共有メモリー空間が不足したり、あるいは環境設定のGQMAXの最大値までキーが作成されてキーが作成できなくなった場合に発生します
[TGECONF]	TmaxGridが設定されていない場合です
[TGEENABLE]	TmaxGridが動作しない場合です
[TGEDELCHL]	子キーが存在して削除に失敗した場合です
[TGEPROTO]	子キーにTG_QUEUEまたはTG_LOCKタイプのノードが存在するか、これらのタイプの子キーを削除する場合に発生します
[TGENOKEY]	該当するキーがない場合です

3.1.5. tmax_grid_set

キーに値を入力します。キーに値が存在する場合は、既存の値は削除され、新しい値に更新されます。キーにWatcherが登録されている場合、既存の値が削除されるときにイベントが発生しません。

TG_EVENT_SET_SELFイベントが登録されている場合は、イベントを通知します。TG_QUEUEまたはTG_LOCKタイプのキーにも値を入力できます。しかし、このタイプの子キーには値を入力できません。

- プロトタイプ

```
#include <usrinc/tmaxapi.h>

int tmax_grid_set(char *key, int keylen, int type, void *data, int len, int flags)
```

- パラメータ

パラメータ	説明
key	作成するキーの名前です。127バイトを超えることはできません
keylen	キーのサイズです
type	キー値のタイプです。 tpallocで割り当てたデータの場合は0を入力し、プリミティブ型の場合は以下を入力します。 – TMAX_GRID_INT – TMAX_GRID_FLOAT – TMAX_GRID_LONG – TMAX_GRID_SHORT – TMAX_GRID_DOUBLE – TMAX_GRID_STRING プリミティブ型を使用する場合は、tpallocを実行しません
data	入力する値です。tpallocで割り当てたポインターまたは実際のプリミティブ型データです
len	データのサイズです
flags	使用しません

- 戻り値

戻り値	説明
>= 0	関数の呼び出しに成功した場合です
< 0	関数の呼び出しに失敗した場合です。tgerrnoにエラーコードが設定されます

- エラー

tmax_grid_set()が正常に実行されなかった場合、tgerrnoに以下の値のいずれかが設定されます。

エラーコード	説明
[TGEINVAL]	キーがNULLであるか、key_len <=0または128バイト以上である場合、あるいはフラグに適切でない値が指定された場合に発生します
[TGEOS]	Tmaxシステムにエラーが発生した場合です。詳細情報はログファイルに記録されます
[TGENOMEM]	メモリーが不足して割り当てに失敗した場合に発生します
[TGEKEY]	キーを正常に認識できない場合に発生します
[TGPROTO]	TG_QUEUEまたはTG_LOCKタイプのノードの子ノードにデータを入力する場合に発生します
[TGESHMFULL]	TmaxGridの共有メモリー空間が不足したり、あるいは環境設定のTGMAXの最大値までキーが作成されてキーが作成できなくなった場合に発生します
[TGEITYPE]	データが正しくないSDLまたはFDLである場合です
[TGECONF]	TmaxGridが設定されていない場合です
[TGEENABLE]	TmaxGridが動作しない場合です

3.1.6. tmax_grid_get

キーの値を取得し、当該キーの値を削除します。値を削除せずに取得するには、フラグを指定します。キーを削除するには、[tmax_grid_destroy](#)関数を呼び出します。

TG_QUEUEまたはTG_LOCKタイプの子キーの場合は値を取得できません。

● プロトタイプ

```
#include <usrinc/tmaxapi.h>
int tmax_grid_get(char *key, int keylen, int *type, char **data, int *len, int flags)
```

● パラメータ

パラメータ	説明
key	作成するキーの名前です。127バイトを超えることはできません
keylen	キーのサイズです
type	関数の実行後、データのタイプを戻します
data	関数の実行後、値を戻します。tpallocまたはfballocであらかじめ割り当てる必要があります
len	関数の実行後、値のサイズを割り当てます

パラメータ	説明
flags	動作方式を指定します – TG_GET_PEEK データを取得し、当該ノードのデータを削除しません

- 戻り値

戻り値	説明
≥ 0	関数の呼び出しに成功した場合です
< 0	関数の呼び出しに失敗した場合です。tgerrnoにエラーコードが設定されます

- エラー

tmax_grid_get()が正常に実行されなかった場合、tgerrnoに以下の値のいずれかが設定されます。

エラーコード	説明
[TGEINVAL]	キーがNULLであるか、key_len ≤ 0 または128バイト以上である場合、あるいはフラグに適切でない値が指定された場合に発生します
[TGEOS]	Tmaxシステムにエラーが発生した場合です。詳細情報はログファイルに記録されます
[TGENOMEM]	メモリーが不足して割り当てに失敗した場合に発生します
[TGEKEY]	キーを正常に認識できない場合に発生します
[TGECONF]	TmaxGridが設定されていない場合です
[TGEENABLE]	TmaxGridが動作しない場合です
[TGEOTYPE]	data tpallocで割り当てていない場合や、取得したデータが正しくないSDLまたはFDLタイプである場合に発生します

3.1.7. tmax_grid_get2

キーのプリミティブ型の値を取得し、当該キーの値は削除します。TG_GET_TYPEフラグを設定して先にタイプを取得することもできます。値を削除せずに取得するには、フラグの指定が必要です。一方、キーを削除するには、tmax_grid_destroy関数を呼び出す必要があります。

TG_QUEUE、またはTG_LOCKタイプの子キーの場合は、値を取得することができません。

- プロトタイプ


```
#include <usrinc/tmaxapi.h>
int tmax_grid_get2(char *key, int keylen, int *type, void *data, int *len, int flags)
```

● パラメータ

パラメータ	説明
key	作成するキーの名前です。127Byteを超えることはできません
keylen	キーの長さです
type	当該キーの値のタイプを戻します – TMAX_GRID_INT TMAX_GRID_FLOAT TMAX_GRID_LONG TMAX_GRID_SHORT TMAX_GRID_DOUBLE TMAX_GRID_STRING
data	関数を実行した後、値を戻します。タイプによっては、型変換(キャスト)をしてから使用します
len	関数を実行した後、値の長さを割り当てます
flags	動作方式を決定します – TG_GET_PEEK データを取得します。そのノードのデータは削除しません – TG_GET_TYPE データのタイプを取得します。そのノードのデータは削除しません

● 戻り値

戻り値	説明
>= 0	関数の呼び出しに成功した場合です
< 0	関数呼び出しに失敗した場合です。tgerrnoにエラーコードが設定されます

● エラー

tmax_grid_get2()が正常に実行されない場合、tgerrnoに以下の値のうち1つが設定されます。

エラーコード	説明
[TGEINVAL]	キーがNULLであるか、key_len <=0またはキーの長さが128バイト以上である場合、あるいはフラグに適切でない値が指定された場合に発生します
[TGEOS]	Tmaxシステムにエラーが発生した場合です。詳細については、ログファイルに記録されます

エラーコード	説明
[TGENOMEM]	メモリーが不足し、割り当てに失敗した場合に発生します
[TGEKEY]	キーが正常に認識されない場合です
[TGECONF]	TmaxGridが設定されていない場合です
[TGEENABLE]	TmaxGridが動作しない場合です
[TGEOTYPE]	データがtpallocで割り当てられていない場合、または取得したデータが正しくないSDLやFDLタイプである場合です

3.1.8. tmax_grid_is_exist

キーが存在するかチェックします。

- プロトタイプ

```
#include <usrinc/tmaxapi.h>
int tmax_grid_is_exist(char *key, int keylen, int flags)
```

- パラメータ

パラメータ	説明
key	チェックするキーの名前です。127バイトを超えることはできません
keylen	キーのサイズです
flags	使用しません

- 戻り値

戻り値	説明
0	キーが存在しない場合です
1	キーだけ存在し、値は存在しない場合です
2	キーと値が両方存在する場合です
< 0	関数の呼び出しに失敗した場合です。tgererrnoにエラーコードが設定されます

- エラー

tmax_grid_is_exist()が正常に実行されなかった場合、tgererrnoに以下の値のいずれかが設定されます。

エラーコード	説明
[TGEINVAL]	キーがNULLであるか、key_len <=0または128バイト以上である場合、あるいはフラグに適切でない値が指定された場合に発生します
[TGEOS]	Tmaxシステムにエラーが発生した場合です。詳細情報はログファイルに記録されます
[TGENOMEM]	メモリーが不足して割り当てに失敗した場合に発生します
[TGEKEY]	キーを正常に認識できない場合に発生します
[TGECONF]	TmaxGridが設定されていない場合です
[TGEENABLE]	TmaxGridが動作しない場合です

3.1.9. tmax_grid_count

キーの全体数を照会します。

- プロトタイプ

```
#include <usrinc/tmaxapi.h>
int tmax_grid_count(int flags)
```

- パラメータ

パラメータ	説明
flags	使用しません

- 戻り値

戻り値	説明
>= 0	関数の呼び出しに成功した場合です。戻り値はキーの全体数です
< 0	関数の呼び出しに失敗した場合です。tgerrnoにエラーコードが設定されます

- エラー

tmax_grid_count()が正常に実行されなかった場合、tgerrnoに以下の値のいずれかが設定されます。

エラーコード	説明
[TGEOS]	Tmaxシステムにエラーが発生した場合です。詳細情報はログファイルに記録されます
[TGENOMEM]	メモリーが不足して割り当てに失敗した場合に発生します
[TGECONF]	TmaxGridが設定されていない場合です

エラーコード	説明
[TGEENABLE]	TmaxGridが動作しない場合です

3.1.10. tmax_grid_lock

キーの名前でロックを実行します。他のクライアントまたはサーバーで同じキー名ですでにロックを実行している場合は、これらのプロセスでアンロックが行われ、自分の番が回ってくるまでタイムアウトで指定した時間の間待機します。ロックに成功した場合は、必ずアンロックする必要があります。もしプログラムがアンロックせずに終了すると、自動でアンロック処理が行われ、次のプロセスがロックを取得します。キーは必ずTG_LOCKで作成する必要があり、TG_LOCKで作成されたキーはデータを取得、設定することができません。キーを作成せずにキー名でロックを実行した場合は、自動で当該名前のキーが作成されます。ロックを実行すると、自動で振られる番号を持つ子キーが作成されます。この子キーはアンロックが呼び出されるときに削除されます。キーの名前はサイズを小さく指定することを推奨します。

1つのプロセス内で同じキー名で再帰的にロックを呼び出すことができます。そのような場合には、必ず同じ回数でアンロックを実行する必要があります。またその場合には、単一の子キーが作成されます。共有ロックを作成するには、フラグをTG_SHARED_LOCKに設定します。共有ロックを作成する場合、同時に複数のノードがロックを取得できます。以前の子ノードがアンロックされると、以降連続する子ノードが共有ロックを使用する場合、そのすべての子ノードがロックを取得します。その後、すべての共有ロックがアンロックされると、待機中のTG_EXCLUSIVE_LOCKに設定された子ノードがロックを取得します。

● プロトタイプ

```
#include <usrinc/tmaxapi.h>
int tmax_grid_lock(char *key, int keylen, int timeout, int flags)
```

● パラメータ

パラメータ	説明
key	作成するキーの名前です。127バイトを超えることはできません
keylen	キーのサイズです
timeout	ロックを取得できる待機時間です(単位：秒)
flags	TG_EXCLUSIVE_LOCK、TG_SHARED_LOCKを設定します

● 戻り値

戻り値	説明
>= 0	関数の呼び出しに成功した場合は
< 0	関数の呼び出しに失敗した場合は。tgerrnoにエラーコードが設定されます

- エラー

tmax_grid_lock()が正常に実行されなかった場合、tgermnoに以下の値のいずれかが設定されます。

エラーコード	説明
[TGEINVAL]	キーがNULLであるか、key_len <=0または128バイト以上である場合、あるいはフラグに適切でない値が指定された場合に発生します
[TGEOS]	Tmaxシステムにエラーが発生した場合です。詳細情報はログファイルに記録されます
[TGENOMEM]	メモリーが不足して割り当てに失敗した場合に発生します
[TGEKEY]	キーを正常に認識できない場合に発生します
[TGEMAXCHL]	環境設定のMax Child制限に達して子キーを作成できなくなった場合に発生します
[TGEPROTO]	キーの名前がTG_LOCKで作成されたキーでない場合に発生します
[TGESHMFULL]	環境設定のTGMAXの最大値までノードが作成され、ロックを行うための子ノードを作成できなくなった場合に発生します
[TGEMAXCHL]	環境設定のTGMAX_CHILD制限に達してロックを行うための子ノードを作成できなくなった場合に発生します
[TGETIME]	タイムアウトの間ロックの要求に失敗した場合です
[TGECONF]	TmaxGridが設定されていない場合です
[TGEENABLE]	TmaxGridが動作しない場合です

3.1.11. tmax_grid_unlock

キーの名前でロックを解除します。次のロックを実行できます。同じキー名で再帰的にロックを実行した場合は、必ずアンロックを同じ回数実行する必要があります。

- プロトタイプ

```
#include <usrinc/tmaxapi.h>
int tmax_grid_unlock(char *key, int keylen, int flags)
```

- パラメータ

パラメータ	説明
key	ロックするキーの名前です。127バイトを超えることはできません
keylen	キーのサイズです
flags	使用しません

- 戻り値

戻り値	説明
≥ 0	関数の呼び出しに成功した場合です
< 0	関数の呼び出しに失敗した場合です。tgerrnoにエラーコードが設定されます

- エラー

tmax_grid_unlock()が正常に実行されなかった場合、tgerrnoに以下の値のいずれかが設定されます。

エラーコード	説明
[TGEINVAL]	キーがNULLであるか、key_len ≤ 0 または128バイト以上である場合、あるいはフラグに適切でない値が指定された場合に発生します
[TGEOS]	Tmaxシステムにエラーが発生した場合です。詳細情報はログファイルに記録されます
[TGENOMEM]	メモリーが不足して割り当てに失敗した場合に発生します
[TGEKEY]	キーを正常に認識できない場合に発生します
[TGECONF]	TmaxGridが設定されていない場合です
[TGEENABLE]	TmaxGridが動作しない場合です
[TGENOKEY]	ロックを実行時に設定したキーが存在しない場合です

3.1.12. tmax_grid_set_watcher

指定したキーのイベントが発生したときに呼び出す関数を登録します。flagsは、Bitwise OR演算で複数の設定を同時に適用できます。

Watcherが設定された状態で再設定を行うと、既存のイベント設定は新しい値に代替されます。コールバックをNULLに指定すると、Watcherを解除し、当該キーに対してそれ以上イベントを受信しません。

コールバック関数で受信できるイベントの種類は以下のとおりです。

イベント	説明
TG_EVENT_DELETE_CHILD	下のフラグと同じです
TG_EVENT_DELETE_SELF	下のフラグと同じです
TG_EVENT_CREATE_CHILD	下のフラグと同じです
TG_EVENT_SET_SELF	下のフラグと同じです
TG_EVENT_GET_SELF	下のフラグと同じです
TG_EVENT_RECOVERED	GQサーバーに障害が発生し、復旧されたことを表します。このイベントを受信した場合、ユーザーはWatcherを再登録する必要があります

● プロトタイプ

```
#include <usrinc/tmaxapi.h>

int tmax_grid_set_watcher(char *key, int keylen, TG_WATCHER_CALLBACK callback,

void *args, int flags)
```

● パラメータ

パラメータ	説明
key	キーの名前です。127バイトを超えることはできません
keylen	キーのサイズです
callback	キーのイベントが発生したとき、イベントを処理する関数のポインターです。ユーザーは「int CALLBACK(char *key, int keylen, int event_type, void *args);」タイプで関数を実装します。それぞれのキーごとに別々のコールバック関数を指定することができます
args	当該キーのイベントが発生してコールバック関数が呼び出されたときに一緒に渡されるユーザー定義引数です
flags	動作方式を指定します <ul style="list-style-type: none">– TG_WATCHER_PERSISTENT Watcherを登録した後、イベントが発生した場合に、クライアントに通知を送り続けます– TG_WATCHER_ONCE Watcherを登録した後、一度だけ通知し、それ以降は通知を送りません。通知をさらに受信したい場合は、コールバック関数を呼び出した後にWatcherを再登録する必要があります– TG_EVENT_DELETE_SELF 自身のキーが削除されたときにイベントを受信します。コールバック関数で当該イベントを受信すると、登録されたWatcherが削除されるので、Watcherを再登録する必要があります– TG_EVENT_CREATE_CHILD 子キーが作成されたときにイベントを受信します– TG_EVENT_SET_SELF 自身のキーにtmax_grid_set APIによりデータが設定されたときにイベントを受信します。またtmax_grid_enqueue APIによりデータが入力されたときにもイベントを受信します

パラメータ	説明
	– TG_EVENT_GET_SELF 自身のキーのデータが取得されたときにイベントを受信します。tmax_grid_get APIを呼び出すときに、フラグにTG_GET_PEEKを設定するとイベントは発生しません [注意] TG_WATCHER_PERSISTENTとTG_WATCHER_ONCEは単独で使うことができません。必ずその他のイベントとBitwise OR演算で組み合わせて適用する必要があります

- 戻り値

戻り値	説明
>= 0	関数の呼び出しに成功した場合です
< 0	関数の呼び出しに失敗した場合です。tgerrnoにエラーコードが設定されます

- エラー

tmax_grid_set_watcher()が正常に実行されなかった場合、tgerrnoに以下の値のいずれかが設定されます。

エラーコード	説明
[TGEINVAL]	キーがNULLであるか、key_len <=0または128バイト以上である場合、あるいはフラグに適切でない値が指定された場合に発生します
[TGEOS]	Tmaxシステムにエラーが発生した場合です。詳細情報はログファイルに記録されます
[TGENOMEM]	メモリーが不足して割り当てに失敗した場合に発生します
[TGENOKEY]	keyに指定した名前で作成されたキーが存在しない場合や、Watcherを解除するときにkeyの名前で指定しなかった場合に発生します
[TGECONF]	TmaxGridが設定されていない場合です
[TGEENABLE]	TmaxGridが動作しない場合です

3.1.13. tmax_grid_wait_watcher

イベントが発生するまでタイムアウトに指定した時間の間待機します。1つのイベントを受信すると、受信したイベントのノードに指定されたコールバック関数が呼び出され、戻されます。

- プロトタイプ

```
#include <usrinc/tmaxapi.h>
int tmax_grid_wait_watcher(int timeout, int flags)
```

- パラメータ

パラメータ	説明
timeout	待機する時間です(単位 : 秒)
flags	使用しないパラメータです

- 戻り値

戻り値	説明
>= 0	関数の呼び出しに成功した場合です
< 0	関数の呼び出しに失敗した場合です。tgerrnoにエラーコードが設定されます

- エラー

tmax_grid_wait_watcher()が正常に実行されなかった場合、tgerrnoに以下の値のいずれかが設定されます。

エラーコード	説明
[TGEOS]	Tmaxシステムにエラーが発生した場合です。詳細情報はログファイルに記録されます
[TGENOMEM]	メモリーが不足して割り当てに失敗した場合に発生します
[TGETIME]	イベントが発生しなかった場合に発生します

3.1.14. tmax_grid_enqueue

キーの名前でデータを入力します。当該キーの子キーの最後の番号でデータが追加されます。キーの名前は必ずTG_QUEUEタイプで作成する必要があります。キーを作成していない場合は、自動でキーを作成してデータを入力します。データの入力順序は保証されます。[tmax_grid_get](#)、[tmax_grid_set](#) APIで直接子キーのデータにアクセスすることは許容されません。しかし、キーの名前ではアクセスが許容されます。

- プロトタイプ

```
#include <usrinc/tmaxapi.h>
int tmax_grid_enqueue(char *key, int keylen, int type, void *data, int len, int flags)
```

- パラメータ

パラメータ	説明
key	作成するキーの名前です。127バイトを超えることはできません
keylen	キーのサイズです
type	<p>キー値のタイプです。</p> <p>tpallocで割り当てたデータの場合は0を入力し、プリミティブ型の場合は以下を入力します。</p> <ul style="list-style-type: none"> – TMAX_GRID_INT – TMAX_GRID_FLOAT – TMAX_GRID_LONG – TMAX_GRID_SHORT – TMAX_GRID_DOUBLE – TMAX_GRID_STRING <p>プリミティブ型を使用する場合はtpallocを実行しません</p>
data	キーに入力する値です。tpallocまたはfballocで割り当てたデータのみ入力できます。あらかじめデータを割り当てる必要があります
len	データのサイズです
flags	<p>動作方式を指定します</p> <ul style="list-style-type: none"> – TG_TEMPORARY <p>クライアントが終了すると、自動で削除します。「/test」というキーでenqueueをした場合、enqueueの対象となる「/test/0」だけ削除され、親ノードの「/test」は削除されません</p>

- 戻り値

戻り値	説明
≥ 0	関数の呼び出しに成功した場合です
< 0	関数の呼び出しに失敗した場合です。tgerrnoにエラーコードが設定されます

- エラー

tmax_grid_enqueue()が正常に実行されなかった場合、tgerrnoに以下の値のいずれかが設定されます。

エラーコード	説明
[TGEINVAL]	キーがNULLであるか、key_len <=0または128バイト以上である場合、あるいはフラグに適切でない値が指定された場合に発生します
[TGEOS]	Tmaxシステムにエラーが発生した場合です。詳細情報はログファイルに記録されます
[TGENOMEM]	メモリーが不足して割り当てに失敗した場合に発生します
[TGEKEY]	キーを正常に認識できない場合に発生します
[TGEMAXCHL]	環境設定のMax Child制限に達して子キーを作成できなくなった場合に発生します
[TGEPROTO]	キーの名前がTG_QUEUEタイプで作成されたノードでない場合に発生します
[TGESHMFULL]	TmaxGridの共有メモリー空間が不足したり、あるいは環境設定のGQMAXの最大値までキーが作成されてキーが作成できなくなった場合に発生します
[TGEDUPKEY]	キーの名前が重複する場合です
[TGECONF]	TmaxGridが設定されていない場合です
[TGEENABLE]	TmaxGridが動作しない場合です
[TGEITYPE]	データが正しくないSDLまたはFDLである場合です

3.1.15. tmax_grid_dequeue

指定したキーの子キーの最初の番号の値を取得します。当該子キーは削除されます。[tmax_grid_get](#)、[tmax_grid_set](#) APIで子キーの値にアクセスすることは許容されません。しかし、キーの名前ではアクセスが許容されます。

● プロトタイプ

```
#include <usrinc/tmaxapi.h>
int tmax_grid_dequeue(char *key, int keylen, int *type, char **data, int *len,
int flags)
```

● パラメータ

パラメータ	説明
key	キーの名前です。127バイトを超えることはできません
keylen	キーのサイズです
type	関数を実行した後、データのタイプを戻します
data	キーで取得する値のバッファです(tpallocまたはfballocで割り当てられたデータのみ可能です。あらかじめデータを割り当てる必要があります)。取得したデータがバッファのサイズより大きい場合は、内部で再度割り当てられます

パラメータ	説明
len	ノードで取得するデータのサイズが保存されます
flags	使用しません

- 戻り値

戻り値	説明
>= 0	関数の呼び出しに成功した場合です
< 0	関数の呼び出しに失敗した場合です。tgerrnoにエラーコードが設定されます

- エラー

tmax_grid_dequeue()が正常に実行されなかった場合、tgerrnoに以下の値のいずれかが設定されます。

エラーコード	説明
[TGEINVAL]	キーがNULLであるか、key_len <=0または128バイト以上である場合、あるいはフラグに適切でない値が指定された場合に発生します
[TGEOS]	Tmaxシステムにエラーが発生した場合です。詳細情報はログファイルに記録されます
[TGENOMEM]	メモリーが不足して割り当てに失敗した場合に発生します
[TGEKEY]	キーを正常に認識できない場合に発生します
[TGENOKEY]	指定したキーがない場合です
[TGENODATA]	入力したデータが存在しない場合です
[TGECONF]	TmaxGridが設定されていない場合です
[TGEENABLE]	TmaxGridが動作しない場合です
[TGEOTYPE]	data tpallocでデータを割り当てていない場合や、取得したデータが正しくないSDLまたはFDLデータである場合です

3.1.16. tmax_grid_dequeue2

キーの名前に該当する子キーの最初の番号のプリミティブ型の値を取得します。当該子キーは削除されます。[tmax_grid_get](#)や[tmax_grid_set](#) APIで子キーの値にアクセスすることはできません。しかし、キーの名前に対してはアクセス可能です。

- プロトタイプ

```
#include <usrinc/tmaxapi.h>
int tmax_grid_dequeue2(char *key, int keylen, int *type, void *data, int *len,
int flags)
```

- パラメータ

パラメータ	説明
key	キーの名前です。127バイトを超えることはできません
keylen	キーの長さです
data	関数を実行した後、値を返します。タイプによっては、型変換(キャスト)をしてから使用します
len	ノードから取得するデータの長さが保存されます
flags	動作方式を決定します <ul style="list-style-type: none"> – TG_GET_TYPE: データのタイプを取得します。そのノードのデータは削除しません

- 戻り値

戻り値	説明
≥ 0	関数の呼び出しに成功した場合です
< 0	関数呼び出しに失敗した場合です。tgerrnoにエラーコードが設定されます

- エラー

tmax_grid_dequeue()が正常に実行されない場合、tgerrnoに以下の値のうち1つが設定されます。

エラーコード	説明
[TGEINVAL]	キーがNULLであるか、key_len ≤ 0 またはキーの長さが128バイト以上である場合、あるいはフラグに適切でない値が指定された場合に発生します
[TGEOS]	Tmaxシステムにエラーが発生した場合です。詳細については、ログファイルに記録されます
[TGENOMEM]	メモリーが不足し、割り当てに失敗した場合に発生します
[TGEKEY]	キーが正常に認識されない場合です
[TGENOKEY]	該当するキーが存在しない場合です
[TGENODATA]	入力されたデータが存在しない場合です
[TGECONF]	TmaxGridが設定されていない場合です
[TGEENABLE]	TmaxGridが動作しない場合です
[TGEOTYPE]	データがtpallocで割り当てられていない場合、または取得したデータが正しくないSDLやFDLタイプである場合です

3.1.17. tmax_grid_get_children

キーの名前で子キーの名前リストを照会します。子キーの情報を含めているTG_KEYLIST_Tハンドラーを返します。失敗した場合は、NULLが返されます。

- プロトタイプ

```
#include <usrinc/tmaxapi.h>
TG_KEYLIST_T tmax_grid_get_children(char *key, int keylen, int *child_count)
```

- パラメータ

パラメータ	説明
key	キーの名前です。127バイトを超えることはできません
keylen	キーのサイズです
child_count	動作方式を指定します

- 戻り値

戻り値	説明
NULLでない場合	TG_KEYLIST_Tでキーの名前リストの開始ポインターを返します。使用しなくなった場合は、freeを実行する必要があります
NULLの場合	関数の呼び出しに失敗した場合です。tgerrnoにエラーコードが設定されます

- エラー

tmax_grid_get_children()が正常に実行されなかった場合、tgerrnoに以下の値のいずれかが設定されます。

エラーコード	説明
[TGEINVAL]	キーがNULLであるか、key_len <=0または128バイト以上である場合、あるいはフラグに適切でない値が指定された場合に発生します
[TGEOS]	Tmaxシステムにエラーが発生した場合です。詳細情報はログファイルに記録されます
[TGENOMEM]	メモリーが不足して割り当てに失敗した場合に発生します
[TGEKEY]	キーを正常に認識できない場合に発生します
[TGECONF]	TmaxGridが設定されていない場合です
[TGEENABLE]	TmaxGridが動作しない場合です
[TGENOKEY]	指定したキーがない場合です

3.1.18. tmax_grid_get_child_with_index

子キーの情報を含めているTG_KEYLIST_Tハンドラーでnth番目のキーの情報をTG_KEYINFO_T構造体に保存します。

- プロトタイプ

```
#include <usrinc/tmaxapi.h>
int tmax_grid_get_child_with_index(TG_KEYLIST_T keylist, int nth,
                                   TG_KEYINFO_T * keyinfo)
```

- パラメータ

パラメータ	説明
keylist	tmax_grid_get_children()を呼び出したときに戻されるハンドラーです
nth	参照する子キーの番号です
keyinfo	子ノードの情報を保存する構造体。割り当てられたバッファを使用する必要があります

- 戻り値

戻り値	説明
>= 0	関数の呼び出しに成功した場合です
< 0	関数の呼び出しに失敗した場合です。tgerrnoにエラーコードが設定されます

- エラー

tmax_grid_get_child_with_index()が正常に実行されなかった場合、tgerrnoに以下の値のいずれかが設定されます。

エラーコード	説明
[TGEINVAL]	keylistがNULLであるか、nth < 0またはkeyinfoがNULLの場合に発生します
[TGEOS]	Tmaxシステムにエラーが発生した場合です。詳細情報はログファイルに記録されます
[TGENOMEM]	メモリーが不足して割り当てに失敗した場合に発生します
[TGEITYPE]	正しくないTG_KEYLIST_Tハンドラーの値が入力された場合です
[TGELIMIT]	nthが子ノードの数より大きい場合です

3.1.19. tmax_grid_keylist_free

子キーの情報を含めているTG_KEYLIST_Tハンドラーのリソースを解除します。

- プロトタイプ

```
#include <usrinc/tmaxapi.h>
int tmax_grid_keylist_free(TG_KEYLIST_T keylist)
```

- パラメータ

パラメータ	説明
keylist	tmax_grid_get_children()を呼び出したときに戻されるハンドラーです

- 戻り値

戻り値	説明
>= 0	関数の呼び出しに成功した場合です
< 0	関数の呼び出しに失敗した場合です。tgerrnoにエラーコードが設定されます

- エラー

tmax_grid_keylist_free()が正常に実行されなかった場合、tgerrnoに以下の値のいずれかが設定されます。

エラーコード	説明
[TGEINVAL]	keylistがNULLの場合です
[TGEOS]	Tmaxシステムにエラーが発生した場合です。詳細情報はログファイルに記録されます
[TGEITYPE]	正しくないTG_KEYLIST_Tハンドラーの値が入力された場合に発生します

3.1.20. tmax_grid_init_key_browser

キーをブラウジングするためのハンドラーを初期化します。初期化パラメータとしてメモリーを割り当てたtg_key_browser_t構造体のポインターとキー、キーの長さを使用します。その後、tmax_grid_get_nextを呼び出してブラウジングします。

- プロトタイプ

```
#include <usrinc/tmaxapi.h>
int tmax_grid_init_key_browser(tg_key_browser_t *browser, char *key, int keylen)
```


- パラメータ

パラメータ	説明
browser	ブラウジングのための構造体です。メモリーを直接割り当てて使用します。使用後には解除しておきます
key	ブラウジングを開始するキー
keylen	キーの長さ

- 戻り値

戻り値	説明
>= 0	関数の呼び出しに成功した場合です
< 0	関数呼び出しに失敗した場合です。tgerrnoにエラーコードが設定されます

- エラー

tmax_grid_init_key_browser()が正常に実行されない場合、tgerrnoに以下の値のうち1つが設定されます。

エラーコード	説明
[TGEINVAL]	ブラウザーにメモリーが割り当てられていないか、キーが存在しない場合、あるいはキーの長さが0以下の場合に発生します。
[TGENOKEY]	当該キーがTmaxGridに存在しない場合に発生します
[TGENOMEM]	メモリーが不足し、割り当てに失敗した場合に発生します
[TGEOS]	Tmaxシステムにエラーが発生した場合です。詳細については、ログファイルに記録されます

3.1.21. tmax_grid_get_next

ブラウジングで次のポジションに位置するキーとその長さを取得します。関数の呼び出しに成功すれば、tg_key_browser_tのnext_keyに次のポジションのキーがnext_key_lenにその長さを戻します。それ以上キーが存在しない場合には、next_keyは空白を戻し、next_key_lenは0を戻します。

- プロトタイプ

```
#include <usrinc/tmaxapi.h>
int tmax_grid_get_next(tg_key_browser_t *browser, int flags)
```

- パラメータ

パラメータ	説明
browser	tmax_grid_init_key_browserを使って初期化したブラウジング構造体です
flags	使用しません

- 戻り値

戻り値	説明
>= 0	関数の呼び出しに成功した場合です
< 0	関数呼び出しに失敗した場合です。tgerrnoにエラーコードが設定されます

- エラー

tmax_grid_get_next()が正常に実行されない場合、tgerrnoに以下の値のうち1つが設定されます。

エラーコード	説明
[TGEINVAL]	ブラウザーにメモリーが割り当てられていないか、または、開始キーが存在しない場合に発生します
[TGENOMEM]	メモリーが不足し、割り当てに失敗した場合に発生します
[TGEOS]	Tmaxシステムにエラーが発生した場合です。詳細については、ログファイルに記録されます

3.1.22. tmax_grid_init_bulk_handle

複数のキー／値のペアを一括でget/setするためのハンドラーを初期化します。初期化パラメータとしてメモリーを割り当てたtg_bulk_handler_t構造体のポインターを使用します。

- プロトタイプ

```
#include <usrinc/tmaxapi.h>
int tmax_grid_init_bulk_handle(tg_bulk_handler_t *lptr)
```

- パラメータ

パラメータ	説明
lptr	bulk get / setのための構造体です。メモリーを直接割り当てて使用します

- 戻り値

戻り値	説明
>= 0	関数の呼び出しに成功した場合です

戻り値	説明
< 0	関数呼び出しに失敗した場合です。tgerrnoにエラーコードが設定されます

- エラー

tmax_grid_init_bulk_handle()が正常に実行されない場合、tgerrnoに以下の値のうち1つが設定されます。

エラーコード	説明
[TGEINVAL]	lptrにメモリーが割り当てられていない場合に発生します

3.1.23. tmax_grid_set_bulk_handle

bulk get/setのためにキー、またはキー／値を設定します。

- プロトタイプ

```
#include <usrinc/tmaxapi.h>
int tmax_grid_set_bulk_handle(tg_bulk_handler_t *lptr, char *key, int keylen,
int type, int subtype, void *data, int len)
```

- パラメータ

パラメータ	説明
lptr	tmax_grid_set_bulk_handleで初期化した構造体です
key	キーの名前です。127バイトを超えることはできません
keylen	キーの長さです
type	該当キーの値のタイプです tpallocしたデータの場合は、0を入力し、 プリミティブ型の場合は、以下のように入力します – TMAX_GRID_INT TMAX_GRID_FLOAT TMAX_GRID_LONG TMAX_GRID_SHORT TMAX_GRID_DOUBLE TMAX_GRID_STRING

パラメータ	説明
	プリミティブ型を使用するときは、tpallocを行いません
subtype	
data	入力しようとする値です。tpallocで割り当てたポインターや、実際のプリミティブ・データです
len	データの長さです

- 戻り値

戻り値	説明
≥ 0	関数の呼び出しに成功した場合です
< 0	関数呼び出しに失敗した場合です。tgerrnoにエラーコードが設定されます

- エラー

tmax_grid_set_bulk_handle()が正常に実行されない場合、tgerrnoに以下の値のうち1つが設定されます。

エラーコード	説明
[TGEINVAL]	キーがNULLであるか、key_len ≤ 0 またはキーの長さが128バイト以上である場合、あるいはフラグに適切でない値が指定された場合に発生します
[TGEOS]	Tmaxシステムにエラーが発生した場合です。詳細については、ログファイルに記録されます
[TGENOMEM]	メモリーが不足し、割り当てに失敗した場合に発生します
[TGEKEY]	キーが正常に認識されない場合です
[TGPROTO]	TG_QUEUE、またはTG_LOCKタイプのノードの子ノードにデータを入力しようとする場合です
[TGESHMFULL]	TmaxGridの共有メモリーの容量が不足した場合、または環境設定のTGMAXの最大値までキーが生成され、もはやキーを作成することができない場合です
[TGEITYPE]	データが正しくないSDLまたはFDLである場合です
[TGECONF]	TmaxGridが設定されていない場合です
[TGEENABLE]	TmaxGridが動作しない場合です

3.1.24. tmax_grid_destroy_bulk_handle

複数のキー／値のペアを一括でget/setするためのハンドラー・リソースを戻します。

- プロトタイプ

```
#include <usrinc/tmaxapi.h>
void tmax_grid_destroy_bulk_handle(tg_bulk_handler_t *lptr)
```

- パラメータ

パラメータ	説明
lptr	bulk get / setに使用した構造体です

- 戻り値

該当なし

- エラー

該当なし

3.1.25. tmax_grid_bulk_set

tg_bulk_handler_tに登録されているキー／値を一括で作成します。該当するキーに値が存在する場合は、既存の値は削除され、新しい値で更新されます。キーにウォッチャーが登録されている場合は、既存の値が削除される際にイベントが発生しません。TG_EVENT_SET_SELFイベントが登録されていれば、イベントを通知します。

- プロトタイプ

```
#include <usrinc/tmaxapi.h>
int tmax_grid_bulk_set(tg_bulk_handler_t *lptr, int flags)
```

- パラメータ

パラメータ	説明
lptr	キー／値のペアが1つ以上登録されているハンドラーです
flags	使用しません

- 戻り値

戻り値	説明
>= 0	関数の呼び出しに成功した場合です
< 0	関数呼び出しに失敗した場合です。tgerrnoにエラーコードが設定されます

- エラー

tmax_grid_bulk_set()が正常に実行されない場合、tgerrnoに以下の値のうち1つが設定されます。

エラーコード	説明
[TGEINVAL]	キーがNULLであるか、key_len <=0またはキーの長さが128バイト以上である場合、あるいはフラグに適切でない値が指定された場合に発生します
[TGEOS]	Tmaxシステムにエラーが発生した場合です。詳細については、ログファイルに記録されます
[TGENOMEM]	メモリーが不足し、割り当てに失敗した場合に発生します
[TGEKEY]	キーが正常に認識されない場合です
[TGPROTO]	TG_QUEUE、またはTG_LOCKタイプのノードの子ノードにデータを入力しようとする場合です
[TGESHMFULL]	TmaxGridの共有メモリーの容量が不足した場合、または環境設定のTGMAXの最大値までキーが生成され、もはやキーを作成することができない場合です
[TGEITYPE]	データが正しくないSDLまたはFDLである場合です
[TGECONF]	TmaxGridが設定されていない場合です
[TGEENABLE]	TmaxGridが動作しない場合です

3.1.26. tmax_grid_bulk_get

tg_bulk_handler_tに登録されているキーの値を取得し、当該キーの値は削除します。TG_GET_TYPEフラグを設定して先にタイプを取得することもできます。値を削除せずに取得するには、フラグの指定が必要です。一方、キーを削除するには、[tmax_grid_destroy](#)関数を呼び出す必要があります。

TG_QUEUE、またはTG_LOCKタイプの子キーの場合は、値を取得することができません。

- プロトタイプ

```
#include <usrinc/tmaxapi.h>
int tmax_grid_bulk_get(tg_bulk_handler_t *lptr, int flags)
```

- パラメータ

パラメータ	説明
lptr	キーが1つ以上登録されているハンドラーです
flags	動作方式を決定します <ul style="list-style-type: none"> – TG_GET_PEEK: データを取得します。そのノードのデータは削除しません

- 戻り値

戻り値	説明
≥ 0	関数の呼び出しに成功した場合です
< 0	関数呼び出しに失敗した場合です。tgerrnoにエラーコードが設定されます

- エラー

tmax_grid_bulk_get()が正常に実行されない場合、tgerrnoに以下の値のうち1つが設定されます。

エラーコード	説明
[TGEINVAL]	キーがNULLであるか、key_len ≤ 0 またはキーの長さが128バイト以上である場合、あるいはフラグに適切でない値が指定された場合に発生します
[TGEOS]	Tmaxシステムにエラーが発生した場合です。詳細については、ログファイルに記録されます
[TGENOMEM]	メモリーが不足し、割り当てに失敗した場合に発生します
[TGEKEY]	キーが正常に認識されない場合です
[TGECONF]	TmaxGridが設定されていない場合です
[TGEENABLE]	TmaxGridが動作しない場合です
[TGEOTYPE]	データがtpallocで割り当てられていない場合、または取得したデータが正しくないSDLやFDLタイプである場合です

3.2. Java言語用のクライアント／サーバーAPI

以下は、Java言語用のクライアント／サーバーAPIについての説明です。

API	説明
gq2Create	キーを作成します
gq2Destroy	キーを削除します
gq2Set	キーの値を設定します
gq2Get	値を取得し、当該キーの値は削除します
gq2IsExist	キーが存在するかどうかをチェックします
gq2Count	全体のキーの数を照会します
gq2Lock	キーの名前でロックを設定します
gq2Unlock	キーの名前でロックを解除します
gq2SetWatcher	当該キーのイベントが発生する際に呼び出す関数を登録します
gq2Enqueue	キーの名前で値を入力します
gq2Dequeue	GQ2Enqueueで入力した値の中で最初の値を照会します
gq2GetChildren	キーの名前を使い、子キーの名前のリストを照会します

3.2.1. gq2Create

キーを作成します。

- プロトタイプ

```
public void gq2Create(String key, WebtGQ2Attribute attr, WebtGQ2Handler handler)
    throws WebtException
```

- パラメータ

パラメータ	説明
key	作成するキーの名前です。127バイトを超えることはできません
attr	C API(tmax_grid_create)と同じです
handler	キーのイベントが発生する際に呼び出すインタフェースです

- エラー

問題が発生したら、WebtExceptionを投げます。

3.2.2. gq2Destroy

キーを削除します。キーの名前でキーと値と一緒に削除します。attrを設定すれば、子キーまで一緒に削除できます。

- プロトタイプ

```
public void gq2Destroy(String key, WebtGQ2Attribute attr)
throws WebtException
```

- パラメータ

パラメータ	説明
key	作成するキーの名前です。127バイトを超えることはできません
attr	C API(tmax_grid_destory)と同じです

- エラー

問題が発生したら、WebtExceptionを投げます。

3.2.3. gq2Set

キーを作成します。

- プロトタイプ

```
public void gq2Set(String key, WebtBuffer data, WebtGQ2Attribute attr)
throws WebtException
```

- パラメータ

パラメータ	説明
key	キーの名前です。127バイトを超えることはできません
data	キーに入力する値です
attr	C API(tmax_grid_set)と同じです

- エラー

問題が発生したら、WebtExceptionを投げます。

3.2.4. gq2Get

キーの値を照会します。この関数を実行すれば、そのキーの値は削除されます。

- プロトタイプ

```
public WebtBuffer gq2Get(String key, WebtGQ2Attribute attr)
throws WebtException
```

- パラメータ

パラメータ	説明
key	照会するキーの名前です。127バイトを超えることはできません
attr	C API(tmax_grid_get)と同じです

- エラー

問題が発生したら、WebtExceptionを投げます。

3.2.5. gq2IsExist

キーが存在するかどうかをチェックします。attrによって、キー、値、子キーや子キーの値までその存在有無をチェックすることができます。

- プロトタイプ

```
public int gq2IsExist(String key, WebtGQ2Attribute attr)
throws WebtException
```

- パラメータ

パラメータ	説明
key	作成するキーの名前です。127バイトを超えることはできません
attr	C API(tmax_grid_is_exist)と同じです

- エラー

問題が発生したら、WebtExceptionを投げます。

3.2.6. gq2Count

全体のキーの数を照会します。戻り値は、GQSが保持しているキーの数です。

- プロトタイプ

```
public int gq2Count()  
throws WebtException
```

- エラー

問題が発生したら、WebtExceptionを投げます。

3.2.7. gq2Lock

キーの名前でロックを実行します。自分の順番が来るまで待機します。

- プロトタイプ

```
public void gq2Lock(String key, int timeout, WebtGQ2Attribute attr)  
throws WebtException
```

- パラメータ

パラメータ	説明
key	キーの名前です。127バイトを超えることはできません
timeout	ロックを取得するまでの待機時間です(単位:秒)
attr	C API(tmax_grid_lock)と同じです

- エラー

問題が発生したら、WebtExceptionを投げます。

3.2.8. gq2Unlock

キーの名前でロックを解除します。次の順番がロックを実行できます。

- プロトタイプ

```
public void gq2Unlock(String key, WebtGQ2Attribute attr)  
throws WebtException
```

- パラメータ

パラメータ	説明
key	キーの名前です。127バイトを超えることはできません

パラメータ	説明
attr	C API(tmax_grid_unlock)と同じです。

- エラー

問題が発生したら、WebtExceptionを投げます。

3.2.9. gq2SetWatcher

当該キーのイベントが発生する際に呼び出す関数を登録します。

- プロトタイプ

```
public void gq2SetWatcher(String key, WebtGQ2Attribute attr, WebtGQ2Handler handler)
throws WebtException
```

- パラメータ

パラメータ	説明
key	作成するキーの名前です。127バイトを超えることはできません
attr	C API(tmax_grid_set_watcher)と同じです。
handler	キーのイベントが発生する際に呼び出すインタフェースです

- エラー

問題が発生したら、WebtExceptionを投げます。

3.2.10. gq2Enqueue

キーの名前で値を入力します。当該キーの子キーの最後の番号に値が追加され、gq2Dequeueで照会できるようになります。

- プロトタイプ

```
public void gq2Enqueue(String key, WebtBuffer data, WebtGQ2Attribute attr)
throws WebtException
```

- パラメータ

パラメータ	説明
key	作成するキーの名前です。127バイトを超えることはできません
data	入力する値です
attr	C API(tmax_grid_enqueue)と 同じです

- エラー

問題が発生したら、WebtExceptionを投げます。

3.2.11. gq2Dequeue

gq2Enqueueで入力した値の中で最初に入力した値を照会します。照会した値は GQSから削除されるので、他のクライアントが照会することはできません。

- プロトタイプ

```
public WebtBuffer gq2Dequeue(String key, WebtGQ2Attribute attr)
throws WebtException
```

- パラメータ

パラメータ	説明
key	作成するキーの名前です。127バイトを超えることはできません
attr	C API(tmax_grid_dequeue)と 同じです

- エラー

問題が発生したら、WebtExceptionを投げます。

3.2.12. gq2GetChildren

キーの名前で、子キーの名前のリストを照会します。すぐ配下の子のリストだけ照会可能です。

戻された配列を使い、各キーの情報を照会することができます。

- プロトタイプ

```
public GQ2KeyList[] gq2GetChildren(String key, WebtGQ2Attribute attr)
throws WebtException
```

- パラメータ

パラメータ	説明
key	作成するキーの名前です。127バイトを超えることはできません
attr	C API(tmax_grid_get_children)と 同じです

- エラー

問題が発生したら、WebtExceptionを投げます。

第4章 例題

本章は、C言語のクライアントやサーバーとJAVA言語のクライアントの例題で構成されています。

4.1. C言語のクライアント

各関数をすべて使用する例題です。

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

void tperror(char *msg)
{
    printf("%s error, [%d:%s]\n", msg, tperrno, tpstrerror(tperrno));
    exit(1);
}

void tgerror(char *msg)
{
    printf("%s error, [%d:%s]\n", msg, tgerrno, tgstrerror(tgerrno));
    exit(1);
}

int my_callback_func(char *key, int keylen, int event_type, void *args)
{
    if (event_type & TG_EVENT_DELETE_CHILD)
        printf("callback key(%s) received TG_EVENT_DELETE_CHILD event\n", keylen, key,
            event_type);

    if (event_type & TG_EVENT_DELETE_SELF)
        printf("callback key(%s) received TG_EVENT_DELETE_SELF event\n", keylen, key,
            event_type);

    if (event_type & TG_EVENT_CREATE_CHILD)
        printf("callback key(%s) received TG_EVENT_CREATE_CHILD event\n", keylen, key,
            event_type);

    if (event_type & TG_EVENT_SET_SELF)
        printf("callback key(%s) received TG_EVENT_SET_SELF event\n", keylen, key,
            event_type, (char*)args);
}
```

```

        if (event_type & TG_EVENT_GET_SELF)
            printf("callback key(%s) received TG_EVENT_GET_SELF event\n",keylen, key,
event_type);

        if (event_type & TG_EVENT_LOCK)
            printf("callback key(%s) received TG_EVENT_LOCK event\n",keylen, key,
event_type);

        if (event_type & TG_EVENT_RECOVERED)
            printf("callback key(%s) received TG_EVENT_RECOVERED event\n",keylen, key,
event_type);
        return 0;
    }
}

int main(int argc, char *argv[])
{
    TG_WATHCER_CALLBACK callback;
    TPSTART_T *start;
    char *sndbuf, *rcvbuf, sndbuf2[1024], rcvbuf2[1024];
    int sndlen, rcvlen;
    char *key, *key1,*lock_key, *queue_key, *parent_key;
    int keylen,keylen1, lock_keylen, queue_keylen, parent_keylen;
    int nth, count, timeout;
    TG_KEYLIST_T keylist;
    TG_KEYINFO_T keyinfo;
    tg_bulk_handler_t *lptr;
    tg_key_browser_t *browser;
    int type;

    if (tmaxreadenv("tmax.env", "TMAX") < 0)
        tperror("tmaxreadenv");
    start = (TPSTART_T*)tpalloc("TPSTART", NULL, 0);
    if (start == NULL)
        tperror("tpalloc");

    if (tpstart(start) < 0)
        tperror("tpstart");

    key = "/keytest/mee";
    key1 = "/key1/message";
    keylen = strlen(key);
    keylen1 = strlen(key1);
    lock_key = "/key/lock";
    lock_keylen = strlen(lock_key);
    queue_key = "/data01";
    queue_keylen = strlen(queue_key);
    parent_key = "/key/message";
    //parent_key = "/key11";
    parent_keylen = strlen(parent_key);

```



```

    /* tmax_grid_create */
    if (tmax_grid_create(key, keylen, TG_TEMPORARY) < 0)
        tgerror("tmax_grid_create");

    if (tmax_grid_create(key1, keylen1, TG_TEMPORARY) < 0)
        tgerror("tmax_grid_create");

#if _CASE1
    /* tmax_grid_is_exist */
    if (tmax_grid_is_exist(key, keylen, TG_NOFLAGS) < 0)
        tgerror("tmax_grid_is_exist");
    printf("tmax_grid_is_exis(%s) exist\n",key);

    if (tmax_grid_is_exist(key1, keylen1, TG_NOFLAGS) < 0)
        tgerror("tmax_grid_is_exist");
    printf("tmax_grid_is_exis(%s) exist\n",key1);

    /* tmax_grid_set_watcher */
    callback = my_callback_func;
    if (tmax_grid_set_watcher(key, keylen, callback, NULL, (TG_WATCHER_PERSISTENT
|
    TG_EVENT_SET_SELF)) < 0)
        tgerror("tmax_grid_set_watcher");
    printf("tmax_grid_set_watcher(%s) ok!\n",key);
#endif

#if _CASE2
    /* tmax_grid_set */
    if ((sndbuf = tpalloc("STRING", NULL, 1024)) == NULL)
        tperror("tpalloc sndbuf");
    sndlen = sprintf(sndbuf, "get pid(%d) data(%s)", getpid(),argv[0]);
    if (tmax_grid_set(key, keylen, 0, sndbuf, sndlen, TG_NOFLAGS) < 0){
        tgerror("tmax_grid_set key");
        tpfree(sndbuf);
    }

    /* tmax_grid_get */
    if ((rcvbuf = tpalloc("STRING", NULL, 1024)) == NULL)
        tperror("tpalloc rcvbuf");
    if(tmax_grid_get(key, keylen, &type , &rcvbuf, &rcvlen, TG_GET_PEEK) < 0){
        tgerror("tmax_grid_get");
        tpfree(rcvbuf);
        tpfree(sndbuf);
        exit(1);
    }
    printf("recv data = [%s] len= [%d]\n", rcvbuf,rcvlen);

```

```

        tpfree(sndbuf);
        tpfree(rcvbuf);
    #endif

    #if _CASE3
        /* tmax_grid_set */
        sndlen = sprintf(sndbuf2, "get2 pid(%d) data(%s)", getpid(),argv[0]);
        if (tmax_grid_set(key1, keylen1,TMAX_GRID_STRING , (void *)sndbuf2, sndlen,
TG_NOFLAGS) < 0){
            tgerror("tmax_grid_set key1");
        }

        /* tmax_grid_get2 */
        type = TMAX_GRID_STRING;
        if (tmax_grid_get2(key1, keylen1, &type, &rcvbuf2, &rcvlen, TG_NOFLAGS) < 0)
            tgerror("tmax_grid_get2");
        printf("recv data = [%s] len= [%d]\n", rcvbuf2,rcvlen);
    #endif

    #if _CASE4
        if (tmax_grid_create(lock_key, lock_keylen, TG_LOCK) < 0)
            tgerror("tmax_grid_create");

        /* grid lock set */
        printf("lock exist\n");
        if (tmax_grid_is_exist(lock_key, lock_keylen, TG_NOFLAGS) < 0)
            tgerror("tmax_grid_is_exist");

        printf("tmax_grid_lock\n");
        /* tmax_grid_lock */
        if (tmax_grid_lock(lock_key, lock_keylen, timeout, TG_SHARED_LOCK) < 0)
            tgerror("tmax_grid_lock");
        printf("tmax_grid_lock ok key[%s]!!\n",lock_key);

        /*if (tmax_grid_lock(lock_key, lock_keylen, timeout, TG_EXCLUSIVE_LOCK) < 0)
            tgerror("tmax_grid_lock");
        printf("tmax_grid_lock ok key[%s]!!\n",lock_key);
        */
        system("echo tgli |tmadmin");

        /* tmax_grid_unlock */
        if (tmax_grid_unlock(lock_key, lock_keylen, TG_NOFLAGS) < 0)
            tgerror("tmax_grid_unlock");
        printf("tmax_grid_unlock ok key[%s]!!\n",lock_key);
        sleep(1);
        system("echo tgli |tmadmin");
    #endif

```

```

    if (tmax_grid_lock(lock_key, lock_keylen, timeout, TG_EXCLUSIVE_LOCK) < 0)
        tgerror("tmax_grid_lock");
    printf("tmax_grid_lock ok key[%s]!!\n", lock_key);

    system("echo tgli |tmadmin");

    /* tmax_grid_unlock */
    if (tmax_grid_unlock(lock_key, lock_keylen, TG_NOFLAGS) < 0)
        tgerror("tmax_grid_unlock");
    printf("tmax_grid_unlock ok key[%s]!!\n", lock_key);
    sleep(1);
    system("echo tgli |tmadmin");

#endif

#if CASE5

    if (tmax_grid_create(queue_key, queue_keylen, TG_QUEUE) < 0)
        tgerror("tmax_grid_create");

    /* tmax_grid_dequeue */
    if ((sndbuf = (char *)tpalloc("STRING", NULL, 1024)) == NULL)
        tperror("tpalloc sndbuf");
    sndlen = sprintf(sndbuf, "pid(%d) date()", getpid());
    type = TMAX_GRID_STRING;

    if (tmax_grid_enqueue(queue_key, queue_keylen, 0, sndbuf, sndlen, TG_NOFLAGS) <
0)
        tgerror("tmax_grid_enqueue");

    if ((rcvbuf = (char *)tpalloc("STRING", NULL, 1024)) == NULL)
        tperror("tpalloc rcvbuf");
    type=TMAX_GRID_STRING;
    if (tmax_grid_dequeue(queue_key, queue_keylen, &type, &rcvbuf, &rcvlen,
TG_NOFLAGS) < 0)
        tgerror("tmax_grid_dequeue");
    printf("tmax_grid_dequeue recv data = [%s]\n", rcvbuf);
    tpfree(sndbuf);
    tpfree(rcvbuf);

    /* tmax_grid_dequeue2*/
    sndlen = sprintf(sndbuf2, "get2 pid(%d) ", getpid());
    if (tmax_grid_enqueue(queue_key, queue_keylen, TMAX_GRID_STRING, (void *)sndbuf2,
sndlen, TG_NOFLAGS) < 0)
        tgerror("tmax_grid_enqueue");

    type=TMAX_GRID_STRING;
    if (tmax_grid_dequeue2(queue_key, queue_keylen, &type, &rcvbuf2, &rcvlen,

```

```

TG_NOFLAGS) < 0)
    tgerror("tmax_grid_dequeue");
    printf("tmax_grid_dequeue2 recv data = [%s]\n", rcvbuf2);

#endif

#if _CASE6
    if (tmax_grid_create(parent_key, parent_keylen, TG_PERSISTENT) < 0)
        tgerror("tmax_grid_create");
    /* tmax_grid_get_children */

    keylist = tmax_grid_get_children(parent_key, parent_keylen, &count);
    if (keylist == NULL)
        tgerror("tmax_grid_get_children");
    for (nth = 0; nth < count; nth++) {
        if (tmax_grid_get_child_with_index(keylist, nth, &keyinfo) == -1)
            break;
        printf("nth(%d) key(%s) datalen(%d)\n", nth, keyinfo.keylen,
            keyinfo.key, keyinfo.datalen);
    }
    tmax_grid_keylist_free(keylist);
    printf("keylist_free = [%s]\n",parent_key);

    system("echo tgi|tmadmin");
    printf("destroy children [%s]\n\n",parent_key);
    if (tmax_grid_destroy(parent_key, parent_keylen, TG_CHILDREN) < 0)
        tgerror("tmax_grid_destroy");
    sleep(1);
    system("echo tgi|tmadmin");
#endif

#if _CASE7
    printf("tg_bulk_handler_t\n");
    lptr=(tg_bulk_handler_t *)malloc(sizeof(tg_bulk_handler_t));

    printf("tg_bulk_handler init\n");
    if(tmax_grid_init_bulk_handle(lptr)<0)
        tgerror("tmax_grid_init_bulk_handle");

    if ((sndbuf = (char *)tpalloc("STRING", NULL, 1024)) == NULL)
        tperror("tpalloc sndbuf");

    sndlen = sprintf(sndbuf, "(%d) date()", getpid());

    printf("grig_set_bulk_handle\n");
    if(tmax_grid_set_bulk_handle(lptr,key,strlen(key), -1,0,sndbuf,sndlen)<0)
        tgerror("tmax_grid_set_bulk_handle");

    sndlen = sprintf(sndbuf2, "13(%d) ", getpid());

```

```

        type=TMAX_GRID_STRING;

        printf("grig_set_bulk_handle\n");
        if(tmax_grid_set_bulk_handle(lp_ptr,key1,strlen(key1),type,0,(void
*)sndbuf2,sndlen)<0)
            tgerror("tmax_grid_set_bulk_handle");

        printf(" bulk count = %d \n",lp_ptr->count);

        printf("grig_set_bulk\n");
        if(tmax_grid_bulk_set(lp_ptr,0)<0)
            tgerror("tmax_grid_bulk_set");

        if(tmax_grid_bulk_get(lp_ptr,0)<0)
            tgerror("tmax_grid_bulk_get");
        printf("get bulk data = %s , key = %s \n",lp_ptr->tail->data, lp_ptr->tail->key);
        system("\necho tgi | tadmin");

    #endif

    #if _CASE8

        browser=(tg_key_browser_t *)malloc(sizeof(tg_key_browser_t));
        tmax_grid_init_key_browser(browser,key,strlen(key));
        rcvlen=tmax_grid_get_next(browser,0);
        printf("browser keylen = %d \n",rcvlen);
        printf("start_key = %s , start_key_len =%d\n",
browser->start_key,browser->start_key_len);
        printf("next_key = %s , next_key_len =%d\n",
browser->next_key,browser->next_key_len);
        count=tmax_grid_count(0);
        printf("tmax_grid conut = %d \n",count);
        system("\necho tgi | tadmin");
    #endif

    tpend();
    return 0;
}

```

4.2. C言語のサーバー

TmaxGridTESTというサービスが呼び出されると、各TmaxGridのAPIを呼び出してみる例題です。

```

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

```

```

void tperror(char *msg)
{
    printf("%s error, [%d:%s]\n", msg, tperrno, tpstrerror(tperrno));
    tpreturn(TPFAIL, tperrno, NULL, 0, TPNOFLAGS);
}

void tgerror(char *msg)
{
    printf("%s error, [%d:%s]\n", msg, tgerrno, tgstrerror(tgerrno));
    tpreturn(TPFAIL, tgerrno, NULL, 0, TPNOFLAGS);
}

int my_callback_func(char *key, int keylen, int event_type, void *args)
{
    if (event_type & TG_EVENT_DELETE_CHILD)
        printf("callback key(%.s) received TG_EVENT_DELETE_CHILD event\n",
keylen, key, event_type);
    if (event_type & TG_EVENT_DELETE_SELF)
        printf("callback key(%.s) received TG_EVENT_DELETE_SELF event\n",
keylen, key, event_type);
    if (event_type & TG_EVENT_CREATE_CHILD)
        printf("callback key(%.s) received TG_EVENT_CREATE_CHILD event\n",
keylen, key, event_type);
    if (event_type & TG_EVENT_SET_SELF)
        printf("callback key(%.s) received TG_EVENT_SET_SELF event\n",
keylen, key, event_type);
    if (event_type & TG_EVENT_GET_SELF)
        printf("callback key(%.s) received TG_EVENT_GET_SELF event\n",
keylen, key, event_type);
    if (event_type & TG_EVENT_LOCK)
        printf("callback key(%.s) received TG_EVENT_LOCK event\n",
keylen, key, event_type);
    if (event_type & TG_EVENT_RECOVERED)
        printf("callback key(%.s) received TG_EVENT_RECOVERED event\n",
keylen, key, event_type);

    return 0;
}

TmaxGridTEST(TPSVCINFO *msg)
{
    T        TG_WATHCER_CALLBACK callback;
    TPSTART_T *start;
    char *sndbuf, *rcvbuf, sndbuf2[1024], rcvbuf2[1024];
    int sndlen, rcvlen;
    char *key, *key1, *lock_key, *queue_key, *parent_key;
    int keylen, keylen1, lock_keylen, queue_keylen, parent_keylen;
    int nth, count, timeout;
    TG_KEYLIST_T keylist;

```

```

TG_KEYINFO_T keyinfo;
tg_bulk_handler_t *lptr;
tg_key_browser_t *browser;
int type;

if (tmaxreadenv("tmax.env", "TMAX") < 0)
tperror("tmaxreadenv");
start = (TPSTART_T*)tpalloc("TPSTART", NULL, 0);
if (start == NULL)
    tpperror("tpalloc");

if (tpstart(start) < 0)
    tpperror("tpstart");

key = "/keytest/mee";
key1 = "/key1/message";
keylen = strlen(key);
keylen1 = strlen(key1);
lock_key = "/key/lock";
lock_keylen = strlen(lock_key);
queue_key = "/data01";
queue_keylen = strlen(queue_key);
parent_key = "/key/message";
//parent_key = "/key11";
parent_keylen = strlen(parent_key);

/* tmax_grid_create */
if (tmax_grid_create(key, keylen, TG_TEMPORARY) < 0)
tgerror("tmax_grid_create");

if (tmax_grid_create(key1, keylen1, TG_TEMPORARY) < 0)
tgerror("tmax_grid_create");

#if _CASE1
/* tmax_grid_is_exist */
if (tmax_grid_is_exist(key, keylen, TG_NOFLAGS) < 0)
tgerror("tmax_grid_is_exist");
printf("tmax_grid_is_exis(%s) exist\n",key);

if (tmax_grid_is_exist(key1, keylen1, TG_NOFLAGS) < 0)
tgerror("tmax_grid_is_exist");
printf("tmax_grid_is_exis(%s) exist\n",key1);

/* tmax_grid_set_watcher */
callback = my_callback_func;
if (tmax_grid_set_watcher(key, keylen, callback, NULL, (TG_WATCHER_PERSISTENT
|
TG_EVENT_SET_SELF)) < 0)
tgerror("tmax_grid_set_watcher");

```

```

        printf("tmax_grid_set_watcher(%s) ok!\n",key);
#endif

#if _CASE2
    /* tmax_grid_set */
    if ((sndbuf = tpalloc("STRING", NULL, 1024)) == NULL)
        tperror("tpalloc sndbuf");
    sndlen = sprintf(sndbuf, "get pid(%d) data(%s)", getpid(),argv[0]);
    if (tmax_grid_set(key, keylen, 0, sndbuf, sndlen, TG_NOFLAGS) < 0){
        tgerror("tmax_grid_set key");
        tpfree(sndbuf);
    }

    /* tmax_grid_get */
    if ((rcvbuf =tpalloc("STRING", NULL, 1024)) == NULL)
        tperror("tpalloc rcvbuf");
    if(tmax_grid_get(key, keylen, &type , &rcvbuf, &rcvlen, TG_GET_PEEK) < 0){
        tgerror("tmax_grid_get");
        tpfree(rcvbuf);
        tpfree(sndbuf);
        exit(1);
    }
    printf("recv data = [%s] len= [%d]\n", rcvbuf,rcvlen);

    tpfree(sndbuf);
    tpfree(rcvbuf);
#endif

#if _CASE3
    /* tmax_grid_set */
    sndlen = sprintf(sndbuf2, "get2 pid(%d) data(%s)", getpid(),argv[0]);
    if (tmax_grid_set(key1, keylen1,TMAX_GRID_STRING , (void *)sndbuf2, sndlen,
TG_NOFLAGS) < 0){
        tgerror("tmax_grid_set key1");
    }

    /* tmax_grid_get2 */
    type = TMAX_GRID_STRING;
    if (tmax_grid_get2(key1, keylen1, &type, &rcvbuf2, &rcvlen, TG_NOFLAGS) < 0)
        tgerror("tmax_grid_get2");
    printf("recv data = [%s] len= [%d]\n", rcvbuf2,rcvlen);
#endif

#if _CASE4
    if (tmax_grid_create(lock_key, lock_keylen, TG_LOCK) < 0)
        tgerror("tmax_grid_create");

```



```

/* grid lock set */
printf("lock exist\n");
if (tmax_grid_is_exist(lock_key, lock_keylen, TG_NOFLAGS) < 0)
tgerror("tmax_grid_is_exist");

printf("tmax_grid_lock\n");
/* tmax_grid_lock */
if (tmax_grid_lock(lock_key, lock_keylen, timeout, TG_SHARED_LOCK) < 0)
tgerror("tmax_grid_lock");
printf("tmax_grid_lock ok key[%s]!!\n",lock_key);

/*if (tmax_grid_lock(lock_key, lock_keylen, timeout, TG_EXCLUSIVE_LOCK) < 0)
tgerror("tmax_grid_lock");
printf("tmax_grid_lock ok key[%s]!!\n",lock_key);
*/
system("echo tgli |tmadmin");

/* tmax_grid_unlock */
if (tmax_grid_unlock(lock_key, lock_keylen, TG_NOFLAGS) < 0)
tgerror("tmax_grid_unlock");
printf("tmax_grid_unlock ok key[%s]!!\n",lock_key);
sleep(1);
system("echo tgli |tmadmin");

if (tmax_grid_lock(lock_key, lock_keylen, timeout, TG_EXCLUSIVE_LOCK) < 0)
tgerror("tmax_grid_lock");
printf("tmax_grid_lock ok key[%s]!!\n",lock_key);

system("echo tgli |tmadmin");

/* tmax_grid_unlock */
if (tmax_grid_unlock(lock_key, lock_keylen, TG_NOFLAGS) < 0)
tgerror("tmax_grid_unlock");
printf("tmax_grid_unlock ok key[%s]!!\n",lock_key);
sleep(1);
system("echo tgli |tmadmin");

#endif

#if CASE5

if (tmax_grid_create(queue_key, queue_keylen, TG_QUEUE) < 0)
tgerror("tmax_grid_create");

/* tmax_grid_dequeue */
if ((sndbuf = (char *)tpalloc("STRING", NULL, 1024)) == NULL)
tperror("tpalloc sndbuf");

```

```

        sndlen = sprintf(sndbuf, "pid(%d) date()", getpid());
        type = TMAX_GRID_STRING;

        if (tmax_grid_enqueue(queue_key, queue_keylen, 0, sndbuf, sndlen, TG_NOFLAGS) <
0)
        tgerror("tmax_grid_enqueue");

        if ((rcvbuf = (char *)tpalloc("STRING", NULL, 1024)) == NULL)
        tperror("tpalloc rcvbuf");
        type=TMAX_GRID_STRING;
        if (tmax_grid_dequeue(queue_key, queue_keylen, &type, &rcvbuf, &rcvlen,
TG_NOFLAGS) < 0)
        tgerror("tmax_grid_dequeue");
        printf("tmax_grid_dequeue recv data = [%s]\n", rcvbuf);
        tpfree(sndbuf);
        tpfree(rcvbuf);

        /* tmax_grid_dequeue2*/
        sndlen = sprintf(sndbuf2, "get2 pid(%d) ", getpid());
        if (tmax_grid_enqueue(queue_key, queue_keylen, TMAX_GRID_STRING, (void *)sndbuf2,
sndlen, TG_NOFLAGS) < 0)
        tgerror("tmax_grid_enqueue");

        type=TMAX_GRID_STRING;
        if (tmax_grid_dequeue2(queue_key, queue_keylen, &type, &rcvbuf2, &rcvlen,
TG_NOFLAGS) < 0)
        tgerror("tmax_grid_dequeue");
        printf("tmax_grid_dequeue2 recv data = [%s]\n", rcvbuf2);

#endif

#if _CASE6
        if (tmax_grid_create(parent_key, parent_keylen, TG_PERSISTENT) < 0)
        tgerror("tmax_grid_create");
        /* tmax_grid_get_children */

        keylist = tmax_grid_get_children(parent_key, parent_keylen, &count);
        if (keylist == NULL)
        tgerror("tmax_grid_get_children");
        for (nth = 0; nth < count; nth++) {
        if (tmax_grid_get_child_with_index(keylist, nth, &keyinfo) == -1)
        break;
        printf("nth(%d) key(%s) datalen(%d)\n", nth, keyinfo.keylen,
keyinfo.key, keyinfo.datalen);
        }
        tmax_grid_keylist_free(keylist);
        printf("keylist_free = [%s]\n",parent_key);

        system("echo tgi|tmadmin");
        printf("destroy children [%s]\n\n",parent_key);

```

```

        if (tmax_grid_destroy(parent_key, parent_keylen, TG_CHILDREN) < 0)
            tgerror("tmax_grid_destroy");
        sleep(1);
        system("echo tgi|tmadmin");
    #endif

    #if _CASE7
        printf("tg_bulk_handler_t\n");
        lptra=(tg_bulk_handler_t *)malloc(sizeof(tg_bulk_handler_t));

        printf("tg_bulk_handler init\n");
        if(tmax_grid_init_bulk_handle(lptra)<0)
            tgerror("tmax_grid_init_bulk_handle");

        if ((sndbuf = (char *)tpalloc("STRING", NULL, 1024)) == NULL)
            tperror("tpalloc sndbuf");

        sndlen = sprintf(sndbuf, "(%d) date()", getpid());

        printf("grig_set_bulk_handle\n");
        if(tmax_grid_set_bulk_handle(lptra,key,strlen(key), -1,0,sndbuf,sndlen)<0)
            tgerror("tmax_grid_set_bulk_handle");

        sndlen = sprintf(sndbuf2, "13(%d) ", getpid());
        type=TMAX_GRID_STRING;

        printf("grig_set_bulk_handle\n");
        if(tmax_grid_set_bulk_handle(lptra,key1,strlen(key1),type,0,(void
*)sndbuf2,sndlen)<0)
            tgerror("tmax_grid_set_bulk_handle");

        printf(" bulk count = %d \n",lptra->count);

        printf("grig_set_bulk\n");
        if(tmax_grid_bulk_set(lptra,0)<0)
            tgerror("tmax_grid_bulk_set");

        if(tmax_grid_bulk_get(lptra,0)<0)
            tgerror("tmax_grid_bulk_get");
        printf("get bulk data = %s , key = %s \n",lptra->tail->data, lptra->tail->key);
        system("\necho tgi | tmadmin");

    #endif

    #if _CASE8

        browser=(tg_key_browser_t *)malloc(sizeof(tg_key_browser_t));
        tmax_grid_init_key_browser(browser,key,strlen(key));
        rcvlen=tmax_grid_get_next(browser,0);
    
```

```

        printf("browser keylen = %d \n",rcvlen);
        printf("start_key = %s , start_key_len =%d\n",
browser->start_key,browser->start_key_len);
        printf("next_key = %s , next_key_len =%d\n",
browser->next_key,browser->next_key_len);
        count=tmax_grid_count(0);
        printf("tmax_grid conut = %d \n",count);
        system("\necho tgi | tadmin");
#endif

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

```

4.3. Java言語のクライアント

WebtRemoteServiceからGQ2に該当するAPIを呼び出してみる例題です。

gq2Setの例

```

import tmax.webt.WebtBuffer;
import tmax.webt.WebtConnection;
import tmax.webt.WebtException;
import tmax.webt.WebtRemoteService;
import tmax.webt.WebtServiceFailException;

public class TestGQ2 {
    public static void main(String[] argv) {
        WebtConnection connection = new WebtConnection("192.168.1.83", 8888, false);
        connection.setHeaderType("extendedV4");
        connection.connect();

        WebtRemoteService service = new WebtRemoteService("GQ2", connection);
        WebtBuffer buffer = service.createStringBuffer();
        buffer.setString("TG_Test");
        try {
            int oi = service.gq2Count();
            System.out.println("oi = " + oi);
            service.gq2Set("/Webt12345", buffer, null);
        } catch (WebtServiceFailException se) {
            se.printStackTrace();
        } catch (WebtException e) {
            e.printStackTrace();
        } finally {
            connection.close();
        }
    }
}

```

```
}  
}
```

gq2Getの例

```
import tmax.webt.WebtBuffer;  
import tmax.webt.WebtConnection;  
import tmax.webt.WebtException;  
import tmax.webt.WebtRemoteService;  
import tmax.webt.WebtServiceFailException;  
  
public class TestGQ2 {  
    public static void main(String[] argv) {  
        WebtConnection connection = new WebtConnection("192.168.1.83", 8888, false);  
        connection.setHeaderType("extendedV4");  
        connection.connect();  
  
        WebtRemoteService service = new WebtRemoteService("GQ2", connection);  
        try {  
            int oi = service.gq2Count();  
            System.out.println("oi = " + oi);  
            WebtBuffer buffer = service.gq2Get("/Webt12345", null);  
            System.out.println(buffer);  
        } catch (WebtServiceFailException se) {  
            se.printStackTrace();  
        } catch (WebtException e) {  
            e.printStackTrace();  
        } finally {  
            connection.close();  
        }  
    }  
}
```

gq2Lockの例

```
import java.io.IOException;  
  
import tmax.webt.GQ2KeyList;  
import tmax.webt.WebtBuffer;  
import tmax.webt.WebtEventConnection;  
import tmax.webt.WebtException;  
import tmax.webt.WebtGQ2Attribute;  
import tmax.webt.WebtGQ2Handler;
```

```

import tmax.webt.WebtRemoteService;
import tmax.webt.WebtServiceFailException;

public class TestGQ2 {
    public static void main(String[] argv) {

        WebtEventConnection connection = new WebtEventConnection("192.168.1.83", 8122,
false);
        connection.setHeaderType("extendedV4");
        connection.connect();

        WebtRemoteService service = new WebtRemoteService("GQ2", connection);
        WebtBuffer buffer = service.createStringBuffer();
        buffer.setString("TG_Test222");
        WebtGQ2Attribute attr = new WebtGQ2Attribute(WebtGQ2Attribute.TG_LOCK);

        WebtGQ2Handler handler = null;
        String key = new String("/WebtLock");
        try {
            int timeout = 6000000;
            service.gq2Lock(key, timeout, attr);
            System.out.println("LOCK");

            /* business coding */

            service.gq2Unlock(key, attr);
            System.out.println("UNLOCK");
        } catch (WebtServiceFailException se) {
            se.printStackTrace();
        } catch (WebtException e) {
            e.printStackTrace();
        } finally {
            connection.close();
        }
    }
}

```

gq2SetWatcherの例

```

package test;

import java.io.IOException;

import tmax.webt.Gq;

```

```

import tmax.webt.WebtAttribute;
import tmax.webt.WebtBuffer;
import tmax.webt.WebtEventConnection;
import tmax.webt.WebtEventHandler;
import tmax.webt.WebtException;
import tmax.webt.WebtGQ2Attribute;
import tmax.webt.WebtGQ2Handler;
import tmax.webt.WebtIOException;
import tmax.webt.WebtRemoteService;

public class GQ2EventSample implements {

    public void handleEvent(String key, int eventType) {

        System.out.println("event received. key = " + key + ", eventType = " +
eventType);

    }

    public static void main(String[] argv) throws IOException {

        WebtEventConnection connection = new WebtEventConnection("192.168.33.216",
12080, false);
        connection.setHeaderType("extendedV4");
        connection.connect();

        WebtRemoteService service = new WebtRemoteService("GQ2", connection);
        WebtBuffer buffer = service.createStringBuffer();
        buffer.setString("EVENT_TEST");

        WebtGQ2Attribute attr =
new WebtGQ2Attribute(WebtGQ2Attribute.TG_WATCHER_PERSISTENT+
WebtGQ2Attribute.TG_EVENT_SET_SELF);
        GQ2EventSample eventSample = new GQ2EventSample();
        String key = new String("/Webt12345");
        service.gq2SetWatcher(key, attr, eventSample);

        System.in.read();

    }

}

```


索引

C

C Client/Server API

- tgstrerror, 18
- tmax_grid_bulk_get, 48
- tmax_grid_bulk_set, 47
- tmax_grid_count, 29
- tmax_grid_create, 18
- tmax_grid_create2, 20
- tmax_grid_dequeue, 37
- tmax_grid_dequeue2, 38
- tmax_grid_destroy, 22
- tmax_grid_destroy_bulk_handle, 46
- tmax_grid_enqueue, 35
- tmax_grid_get, 25
- tmax_grid_get_child_with_index, 41
- tmax_grid_get_children, 40
- tmax_grid_get_next, 43
- tmax_grid_get2, 26
- tmax_grid_init_bulk_handle, 44
- tmax_grid_init_key_browser, 42
- tmax_grid_is_exist, 28
- tmax_grid_keylist_free, 42
- tmax_grid_lock, 30
- tmax_grid_set, 23
- tmax_grid_set_bulk_handle, 45
- tmax_grid_set_watcher, 32
- tmax_grid_unlock, 31
- tmax_grid_wait_watcher, 34

G

- gq2Count, 52
- gq2Create, 50
- gq2Dequeue, 55
- gq2Destroy, 51
- gq2Enqueue, 54
- gq2Get, 52

- gq2GetChildren, 55
- gq2IsExist, 52
- gq2Lock, 53
- gq2Set, 51
- gq2SetWatcher, 54
- gq2Unlock, 53

J

Java Client/Server API

- gq2Count, 52
- gq2Create, 50
- gq2Dequeue, 55
- gq2Destroy, 51
- gq2Enqueue, 54
- gq2Get, 52
- gq2GetChildren, 55
- gq2IsExist, 52
- gq2Lock, 53
- gq2Set, 51
- gq2SetWatcher, 54
- gq2Unlock, 53

T

- tgstrerror, 18
- tmax_grid_bulk_get, 48
- tmax_grid_bulk_set, 47
- tmax_grid_count, 29
- tmax_grid_create, 18
- tmax_grid_create2, 20
- tmax_grid_dequeue, 37
- tmax_grid_dequeue2, 38
- tmax_grid_destroy, 22
- tmax_grid_destroy_bulk_handle, 46
- tmax_grid_enqueue, 35
- tmax_grid_get, 25
- tmax_grid_get2, 26
- tmax_grid_get_child_with_index, 41
- tmax_grid_get_children, 40
- tmax_grid_get_next, 43
- tmax_grid_init_bulk_handle, 44
- tmax_grid_init_key_browser, 42
- tmax_grid_is_exist, 28

tmax_grid_keylist_free, 42
tmax_grid_lock, 30
tmax_grid_set, 23
tmax_grid_set_bulk_handle, 45
tmax_grid_set_watcher, 32
tmax_grid_unlock, 31
tmax_grid_wait_watcher, 34