

Tmax プログラミングガイド (SQ)

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 プログラミングガイド (SQ)

発行日: 2016年8月5日

ソフトウェアバージョン: Tmax v6.0

ガイドバージョン: v2.1.1

目次

このガイドについて	ix
第1章 紹介	1
1.1. 概要	1
1.2. システム構成	2
1.3. 特徴	3
第2章 使用と管理	5
2.1. 環境設定	5
2.1.1. NODEセクション	5
2.2. システム管理	7
2.2.1. 状態情報の管理	7
第3章 API	11
3.1. 概要	11
3.2. SQ API	12
3.2.1. tmax_sq_put	12
3.2.2. tmax_sq_get	13
3.2.3. tmax_sq_count	14
3.2.4. tmax_sq_purge	15
3.2.5. tmax_sq_keygen	15
3.2.6. tmax_sq_getkeylist	16
3.3. GQ API	17
3.3.1. tmax_gq_put	17
3.3.2. tmax_gq_get	18
3.3.3. tmax_gq_count	19
3.3.4. tmax_gq_purge	20
3.3.5. tmax_gq_keygen	21
3.3.6. tmax_gq_getkeylist	21
3.4. キーおよびセッション照会API	22
3.4.1. tmax_get_sessionid	22
3.4.2. tmax_keylist_count	23
3.4.3. tmax_keylist_getakey	23
3.4.4. tmax_keylist_free	25
第4章 例	27
4.1. クライアント・プログラム	27
4.2. サーバー・プログラム	30
付録 A. ヘッダー・ファイル	33
A.1. tmaxapi.h	33
索引	41

図目次

[図 1.1] Tmaxセッションのフロー	1
[図 1.2] SQシステムの構成	2

このガイドについて

対象読者

本書は、Tmax[®](以下、Tmax)のSQ(Session Queue)を使用して開発を行う開発者を対象としており、SQの概念と使用方法について説明しています。

前提知識

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

本書を理解するためには、以下の事項についての知識が必要です。

- ミドルウェア(Middleware)およびUNIXシステムについて
- Tmaxの基本概念について
- Java、Cプログラミングについて

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セッションの定義とSQのシステム構成および特徴について説明します。

- 第2章: 使用と管理

SQシステムを使用するためにTmax環境ファイルにSQを設定する手順と、管理ツールを使用してSQ状態情報を確認および管理する方法について説明します。

- 第3章: API

実際にアプリケーションを開発するためのAPIの使用方法について説明します。

- 第4章: 例

SQシステムを使用するクライアントとサーバーの例について説明します。

- 付録A: ヘッダー・ファイル

SQ APIのプロトタイプとシステム変数を宣言したヘッダー・ファイルの例を紹介します。

表記上の規則

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

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
Web (Korean): <http://www.tmaxsoft.com>
TechNet: <http://technet.tmaxsoft.com>

USA

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

Japan

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

China

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

Brazil

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

Russia

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

Singapore

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

United Kingdom

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

Canada

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

Australia

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

India

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

Turkey

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

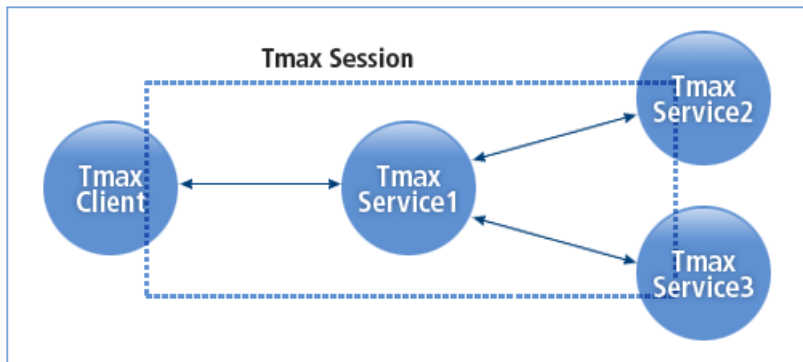
第1章 紹介

本章では、Tmaxセッションの定義とSQのシステム構成および特徴について説明します。

1.1. 概要

Tmaxセッションは、基本的にクライアントがTmaxシステムと接続を確立(tpstart()関数)し、切断(tpend()関数)することを1つのセッションとして定義します。この場合、クライアントで呼び出されたサービスは同じセッションになり、セッションはドメインでのみ維持されます。

【図 1.1】 Tmaxセッションのフロー



UCSサービスはusermain()ルーチンで要求が開始された場合に新しいセッションが開始され、このとき使用されるOUTBOUNDチャンネルはCLHと持続的に接続を確立するので1つのセッション(Long Term Session)として処理されます。再開始するたびに新しいセッションが始まります。

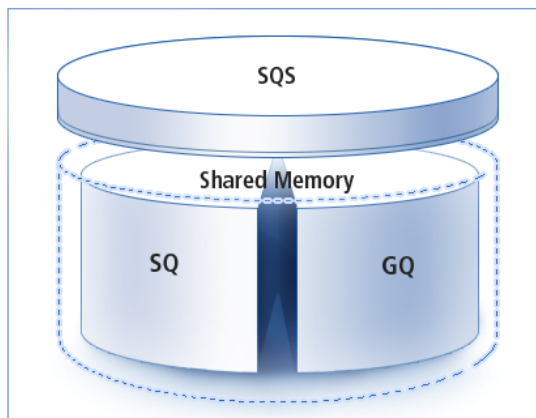
SQ(Session Queue)は同じセッションのクライアントとサービス間で効率的なデータ共有のためのセッション・データ・ストレージを提供します。同じセッションでのみアクセスが可能です。セッションが終了すると、自動でリソースが解除されます。

さらに、セッションと関係なく、グローバルにアクセスが可能なGQ(Global Queue)も提供します。GQのデータは、エンジンが終了するときにリソースを解除します。Tmax SQシステムは共有メモリー・ストレージを使用して、より高速かつ効率的なデータ共有を提供します。

1.2. システム構成

Tmax SQシステムは、システム・ストレージ、SQS、RQ APIで構成されます。

[図 1.2] SQシステムの構成



- システム・ストレージ

SQシステムを使用するために、環境設定ファイルに共有メモリーキー値およびサイズなどを指定する必要があります。また、ストレージのサイズがメモリーのサイズを超えることができないので、大容量データを保存する場合には制限する必要があります。

システム・ストレージ	説明
SQ(Session Queue)	同じセッションに属するクライアントとサービス間で効率的なデータ共有のためのセッション・データ・ストレージです。同じセッションでのみアクセスが可能で、セッションが終了すると自動的にリソースが解放されます
GQ(Global Queue)	セッションに関係なく、グローバルにアクセス可能なセッション・データ・ストレージです。GQのデータはエンジンが終了するときリソースを解放します

- SQS

SQSはCLH(クライアント・マネージャー)の制御を受け、SQデータを保存して読み込むすべてのプロセスを管理および制御します。

- RQ API

Tmaxでは、アプリケーション開発のためにAPIを提供しています。RQ APIについての詳細は、[「第3章 API」](#)を参照してください。

1.3. 特徴

Tmax SQシステムは以下のような特徴を持ちます。

- キー・ベースのキュー

キューで使用するキー方式は、以下のように2つの方式があります。

- システムキー(SYSKEY)の使用方式

システムキーを使用するとき、SQシステムで生成されるユニークなキーを使用できます。SYSKEYは16バイト(SQ_SYSKEY_SIZEで指定)サイズのキーで、「\$SQID」という5つの文字で始まります。残りの11バイトはシステム内部で使われます。したがって、ユーザーは「\$SQID」で始まるキーを任意に使用できません。

- ユーザーキー(USERKEY)の使用方式

ユーザーが任意に指定できるキーです。一意性の保証はユーザーの責任です。

SQは一般的なキューと違ってキー基盤(Key-based)のキューを提供するので、データの入出力の際に必ずキーを使用する必要があります。SQではキーの一意性(Uniqueness)を保証するので、データを入力(PUT)するとき重複するキーが入力されると、エラー(TPEMATCH)を返します。ただし、アップデート・フラグ(TPSQ_UPDATE)を使用する場合は、データがアップデートされます。

GQの場合は、ユーザーキーを使用する場合、一意性についてはシステムの効率性のためにユーザーに委ねています。ただし、システムキーを使用する場合は一意性を保証します。

- 一時的な分散ストレージ

SQの高速かつ効率的なI/Oをサポートするために共有メモリーストレージを使用します。特にマルチノード環境では、分散保存するようにして拡張性を提供します。主にセッションが維持される間一時的に使用されるため、データはセッションが終了するとき自動的に削除され、エンジン障害が発生した場合の復旧(Recovery)はサポートしません。

GQの場合は、エンジンが終了するとき、データが削除されます。

参考

1つのトランザクションにまとめる場合や、データの損失が致命的である場合は、RQを使用することをお勧めします。

- GQのサポート

GQ(Global Queue)はセッションに関係なく、グローバルにデータを共有できます。マルチノード環境までサポートし、ドメインを超えることはできません。GQもシステムキーとユーザーキーを使用することができ、性能上システムキーを使用することをお勧めします。

- モニタリングおよび管理機能のサポート

SQのモニタリングと管理は、Tmax管理ツール(tmadmin)によってサポートします。

第2章 使用と管理

本章では、SQシステムを使用するためにTmax環境ファイルにSQを設定する手順と、管理ツールを使用してSQ状態情報を確認および管理する方法について説明します。

2.1. 環境設定

アプリケーションの開発に前もって、SQシステムを開始するためには基本的な環境設定が必要です。

2.1.1. NODEセクション

SQシステムはノード単位で管理されるため、Tmax環境ファイルのNODEセクションに設定する必要があります。

SQシステムを使用するためには、NODEセクションにSQKEY、SQSIZE、SQMAX、SQKEYMAX、SQTIMEOUT項目を設定します。SQを使用するためにはSQKEY、SQSIZE、SQMAXは必ず設定する必要があります。

参考

Tmax環境設定の詳細については、『Tmax 運用ガイド』の「第3章 環境ファイルの設定」を参照してください。

NODEセクションの環境設定は、以下のような形式で定義します。

```
* NODE
Node Name      SQKEY = 32768~262143 ,
                SQSIZE = 4 ~ 4000000 ,
                SQMAX= 2~MAX_INT-1 ,
                SQKEYMAX= 1 ~ MAX_INT ,
                SQTIMEOUT=1 ~ MAX_INT
```

- Node Name = string
 - サイズ : 63文字以内
 - ノードの物理的な名前として、UNIXのuname -nコマンドを使って確認された名前を定義します。設定したノード名は、「/etc/hosts」ファイルに登録されている必要があります。1つのドメイン(DOMAIN)は、1つ以上のノードで構成されるので、NODEセクションには少なくとも1つ以上のノード名が定義される必要があります。

- SQKEY = numeric
 - 範囲: 32768~262143
 - SQストレージとして使用される共有メモリー・セグメント(shared memory segment)を示すキー値を定義します。
- SQSIZE = numeric
 - 範囲: 4~4000000(単位 : KB)
 - SQストレージとして使用される共有メモリー・セグメントのサイズを定義します。
- SQMAX = numeric
 - 範囲: 2~MAX_INT-1(単位 : KB)
 - ノードに生成できる最大のSQ数であり、偶数で設定します。奇数で設定した場合は、1つ小さい値になります。
- SQKEYMAX = numeric
 - デフォルト値: 1~MAX_INT
 - 各SQで保存できる最大のキー数です。
- SQTIMEOUT = numeric
 - デフォルト値: 1~MAX_INT (単位: 秒)
 - SQデータの保存タイムアウトです。タイムアウトになると、データが自動削除されます。

使用例

```
*DOMAIN
tmax      ...

*NODE
tmax1     ... ,
          SQKEY = 80000 ,
          SQSIZE = 8192 ,
          SQMAX = 1024 ,
```



```
SQKEYMAX = 64,  
SQTIMEOUT = 30
```

2.2. システム管理

Tmaxシステムが起動すると、Tmaxが提供する管理ツールのtmadminを使用して、システムの状態を管理することができます。

参考

管理ツールの詳細については、『Tmax 運用ガイド』を参照してください。

2.2.1. 状態情報の管理

tmadminのSQに関連したコマンドは、**sqinfo(sqi)**と**sqpurge(sqp)**があります。

コマンド	説明
sqinfo(sqi)	現在使われているSQとGQの使用量およびキーについての情報を照会することができます
sqpurge(sqp)	管理者がキーを強制的に削除することができます

sqinfo(sqi)

SQおよびGQの使用量とキー情報を照会できます。マルチノード環境ではノード別に照会します(GQの場合、セッションIDは「0xffffffff」です)。オプションを与えない場合は、SQおよびGQの使用量などキューのすべての情報を提供します。

- 使用方法

```
$ sqi [-k session_id]
```

- オプション

オプション	説明
[-k session_id]	指定したセッションに保存されているキー・リストの詳細情報を提供します

- 例

- 以下は、コマンドの実行例です。

```
$ tmax1 (tmadm): sqi
```

```

session_id  type  clhno  index      issuer      count  size
-----
0xffffffff GLOBAL    -      -          -          5     785
0x00000327 SERVER    0     113        svr_ucs     0      0
0x00004000 CLIENT    1      0     192.168.1.123  0      0
0x00004327 SERVER    1     113        svr_ucs     0      0

-----
allocated in total = 4096 bytes, actually used = 2208 bytes
-----

Msg from rnode(tmax2):
-----
session_id  type  clhno  index      issuer      count  size
-----
0xffffffff GLOBAL    -      -          -          0      0

-----
allocated in total = 4096 bytes, actually used = 576 bytes
-----

```

以下は、出力された各項目についての説明です。

項目	説明
session_id	現在作成されているセッションのIDです
type	セッションのタイプです - GLOBAL : GQ - CLIENT : クライアントに接続した場合 - SERVER : サーバーに接続した場合
clhno	SQの場合、接続したCLHの番号です
index	SQの場合、セッションの索引です
issuer	SQの場合、セッションの情報をタイプに応じて出力します。 タイプが「CLIENT」の場合はIPを、「SERVER」の場合にはサーバー名を出力します
count	作成したキューの数です
size	キュー・データのサイズです
allocated in total	使用するために割り当てられたメモリーのサイズです。この値には、システム内部で使用するメモリー・サイズが含まれます
actually used	割り当てられたメモリーの中で、実際に使用しているメモリーのサイズです。この値には、システム内部で使用するメモリー・サイズが含まれます

– 以下は、[-k]オプションを使用したsqiコマンドの使用例です。

```
$ tmax1 (tmadm): sqi -k 0x00004000
-----
ind key(hexa)                                key(string)      klen dlen  stime
-----
  1 61626331                                abcd              4   144 10:50:07
```

以下は、出力された各項目についての説明です。

項目	説明
ind	順序を示します
key(hexa)	キーをHEX値で出力します
key(string)	ユーザーが作成したキー文字列です
klen	キー文字列の長さです
dlen	保存したデータの長さです(単位: バイト)
stime	保存した時間です

sqpurge(sqp)

マスター・モードのtmadminの場合、SQとGQの目的のキーを削除することができます。ただし、ローカル・ノードにあるキーのみ削除できます。

- 使用方法

```
$ sqp
```

- 例

以下は、コマンドの実行例です。

```
$ tmax1 (tmadm): sqp 0x00000000
-----
ind key(hexa)                                key(string)      klen dlen  stime
-----
  1 61626331                                abc1              4   144 10:47:47

select?(1 - 1, 0 for all) 1

1 will be purged.
key = 61626331(abc1), klen(dlen) = 4(144).
ok?(y or n) y

purge ok
```


第3章 API

本章では、実際にアプリケーションを開発するためのAPIの使用方法について説明します。

3.1. 概要

Tmax SQシステムで提供するAPIはSQを使用するためのAPI、GQを使用するためのAPI、キーリストおよびセッションを照会するためのAPIに分けられます。SQ API(「tmax_sq_」で始まる)とGQ API(「tmax_gq_」で始まる)は関数の名前だけ違い、同じ方法で実行されます。

以下は、各APIのリストです。

• SQ API

API	説明
tmax_sq_put	サーバーデータをセッションキューに保存する関数です
tmax_sq_get	データをセッションキューに保存する関数です
tmax_sq_count	現在SQに保存されているデータ数を返す関数です
tmax_sq_purge	SQのデータを削除する関数です
tmax_sq_keygen	システムキーを生成および取得するための関数です
tmax_sq_getkeylist	現在のセッションSQのキーリストを取得するための関数です

• GQ API

API	説明
tmax_gq_put	データをGQに保存する関数です
tmax_gq_get	GQからデータを取得する関数です
tmax_gq_count	GQに保存されているデータ数を返す関数です
tmax_gq_purge	GQのデータを削除する関数です
tmax_gq_keygen	システムキーを生成および取得するための関数です
tmax_gq_getkeylist	GQのキーリストを取得するための関数です

• キーリストおよびセッション照会API

API	説明
tmax_get_sessionid	現在のセッションIDを返す関数です

API	説明
tmax_keylist_count	keylistハンドルからキーリストの数を返します
tmax_keylist_getakey	keylistハンドルからn番目のキー情報を取得します
tmax_keylist_free	keylistハンドルのメモリーやその他のリソースを解除します

参考

各APIのエラーコードについては、『Tmax アプリケーション開発ガイド』を参照してください。

3.2. SQ API

3.2.1. tmax_sq_put

サーバーのデータをSQに保存する関数です。キーとデータ値を渡す必要があります。

● プロトタイプ

```
#include <tmaxapi.h>
int tmax_sq_put(char *key, long keylenl, char *data, long lenl, long flagsl)
```

● パラメータ

パラメータ	説明
key	SQに保存するデータのためのキー値を保存するバッファです
keylenl	キー・バッファのサイズを指定します(単位: バイト)
data	SQに保存するデータを保存しているバッファです。 tpalloc() で割り当てられたバッファのポインターである必要があります
lenl	SQに保存するデータ・バッファのサイズを指定します(単位: バイト)
flagsl	TPSQ_UPDATE、TPSQ_SYSKEY、TPSQ_KEYGENを指定できます
flagsl	<p>以下は、flagslに設定される値についての説明です</p> <ul style="list-style-type: none"> – TPSQ_UPDATE <p>キー値が同じ場合、アップデートします。フラグが設定されていない場合、TPEMATCHエラーが返されます</p> – TPSQ_SYSKEY <p>tmax_sq_keygen()により生成されたシステム・キーを使用する場合に設定します</p>

パラメータ	説明
	– TPSQ_KEYGEN システム・キーを自動で作成して保存します。キーはSQ_SYSKEY_SIZEのサイズで割り当てる必要があります(keylenl = SQ_SYSKEY_SIZE)。正常に実行した場合、keylに生成されたキー値が保存されます

- 戻り値

戻り値	説明
1	関数の呼び出しに成功した場合です
-1	関数の呼び出しに失敗した場合です。tperrnoにエラーコードが設定されます

- エラー

tmax_sq_put()が正常に実行されなかった場合、tperrno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
[TPENOENT]	現行セッションのためのSQが存在しない場合に発生します
[TPEMATCH]	使用されたキー値がすでに存在する場合に発生します
[TPELIMIT]	SQの最大数を超過しているか、またはSQの最大キー数を超過している場合に発生します

3.2.2. tmax_sq_get

データをセッション・キューに保存する関数です。SQでデータを取得するためにキーを指定した場合、該当キーのデータを取得します。基本的に、tmax_sq_get()を実行するとキューからデータが削除されます。データを保存するには、TPSQ_PEEKフラグを設定する必要があります。

- プロトタイプ

```
#include <tmaxapi.h>
int tmax_sq_get(char *key, long keylenl, char **data, long *lenl, long flagsl)
```

- パラメータ

パラメータ	説明
key	SQから取得するデータのためのキー値を保存するバッファです
keylenl	キー・バッファのサイズを指定します(単位: バイト)

パラメータ	説明
data	SQから取得するデータ・バッファのポインターです。tpalloc()で割り当てられたバッファのポインターである必要があります
lenl	SQから取得したデータ・バッファのサイズが保存されます(単位: バイト)
flagsl	TPSQ_PEEKを指定できます。このフラグを指定した場合、データをキューから削除しません

- 戻り値

戻り値	説明
1	関数の呼び出しに成功した場合です
-1	関数の呼び出しに失敗した場合です。tperrnoにエラーコードが設定されます

- エラー

tmax_sq_get()が正常に実行されなかった場合、tperrno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
[TPENOENT]	現行セッションのためのSQが存在しない場合です
[TPEMATCH]	指定したキーのデータが存在しない場合です

3.2.3. tmax_sq_count

現行SQに保存されているデータ数を返す関数です。

- プロトタイプ

```
#include <tmaxapi.h>
int tmax_sq_count(void)
```

- 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
-1	関数の呼び出しに失敗した場合です。tperrnoにエラーコードが設定されます

- エラー

tmax_sq_count()が正常に実行されなかった場合、tperrno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
[TPENOENT]	現行セッションのためのSQが存在しない場合です

3.2.4. tmax_sq_purge

SQのデータを削除する関数です。キーを指定した場合は該当キーのデータを削除し、キーを指定していない場合はすべてのデータを削除します。

- プロトタイプ

```
#include <tmaxapi.h>
int tmax_sq_purge(char *key, long keylenl)
```

- パラメータ

パラメータ	説明
key	SQから削除するデータのためのキー値を保存するバッファです。NULLの場合、現行セッションSQのすべてのデータを削除します
keylenl	キー・バッファのサイズを指定します(単位: バイト)

- 戻り値

戻り値	説明
削除されたデータ数	関数の呼び出しに成功した場合です
-1	関数の呼び出しに失敗した場合です。tperrnoにエラーコードが設定されます

- エラー

tmax_sq_purge()が正常に実行されなかった場合、tperrno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
[TPENOENT]	現行セッションのためのSQが存在しない場合です

3.2.5. tmax_sq_keygen

システム・キーを作成および取得する関数です。システム・キーのサイズは、SQ_SYSKEY_SIZE(16バイト)です。バッファ・サイズはシステム・キーのサイズより大きい値か、同一の値を割り当てる必要があります。

- プロトタイプ

```
#include <tmaxapi.h>
int tmax_sq_keygen(char *key, long keylen1)
```

- パラメータ

パラメータ	説明
key	生成されたシステム・キー値が保存されるバッファです。バッファのサイズは、SQ_SYSKEY_SIZE(16バイト)より大きい値を割り当てる必要があります
keylen1	キー・バッファのサイズを指定します。必ずSQ_SYSKEY_SIZE(16バイト)より大きい値を指定します

- 戻り値

戻り値	説明
1	関数の呼び出しに成功した場合です
-1	関数の呼び出しに失敗した場合です。tperrnoにエラーコードが設定されます

- エラー

tmax_sq_keygen()が正常に実行されなかった場合、tperrno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
[TPENOENT]	現行セッションのためのSQが存在しない場合です

3.2.6. tmax_sq_getkeylist

現行セッションSQのキー・リストを取得する関数です。キー・リストは、SQ_KEYLIST_Tタイプのハンドルが返され、各キーについての情報は、tmax_keylist_count()、tmax_keylist_getakey()、tmax_keylist_free()関数を使用して確認できます。

キー・リストはバイト単位で、ASCIIコード値を比較して低い値順にソートされます。キー値を指定した場合、該当キー値より大きいキー・リスト、もしくは同一のキー・リストがソートされて照会されます。キー値をNULLと指定した場合、全体キー・リストが照会されます。

- プロトタイプ

```
#include <tmaxapi.h>
SQ_KEYLIST_T tmax_sq_getkeylist(char *key, long keylen1)
```

- パラメータ

パラメータ	説明
key	キー値を保存するバッファです
keylenl	キー・バッファのサイズを指定します(単位: バイト)

- 戻り値

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

- エラー

tmax_sq_getkeylist()が正常に実行されなかった場合、tperrno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
[TPENOENT]	現行セッションのためのSQが存在しない場合です
[TPEMATCH]	指定したキーより大きいキー、もしくは同一のキーが存在しない場合です

3.3. GQ API

3.3.1. tmax_gq_put

データをGQに保存する関数で、キーとデータ値を渡します。**tmax_gq_keygen()**関数を使用して生成されたシステム・キーを使用するには、**TPSQ_SYSKEY**フラグを使用します。システム・キーを作成して保存するには、**TPSQ_KEYGEN**フラグを使用します。

- プロトタイプ

```
#include <tmaxapi.h>
int tmax_gq_put(char *key, long keylenl, char *data, long lenl, long flagsl)
```

- パラメータ

パラメータ	説明
key	GQに保存するデータのためのキー値を保存するバッファです
keylenl	キー・バッファのサイズを指定します(単位: バイト)
data	GQに保存するデータを保存しているバッファです。 tpalloc() で割り当てられたバッファのポインターである必要があります
lenl	GQに保存するデータ・バッファのサイズを指定します(単位: バイト)

パラメータ	説明
flagsl	<p>TPSQ_UPDATE、TPSQ_SYSKEY、TPSQ_KEYGENが指定できます。</p> <p>以下は、flagslに設定可能な値についての説明です</p> <ul style="list-style-type: none"> TPSQ_UPDATE <p>キー値が同じ場合、アップデートします。このフラグが設定されていない場合、TPEMATCHエラーが返されます</p> TPSQ_SYSKEY <p>システム・キーを使用する場合に設定します</p> TPSQ_KEYGEN <p>システム・キーを自動作成して保存します。このフラグを使用する場合、keyはSQ_SYSKEY_SIZEサイズを割り当てる必要があります(keylenl = SQ_SYSKEY_SIZE)。正常に実行した場合、keyに生成されたキー値が保存されます</p>

- 戻り値

戻り値	説明
1	関数の呼び出しに成功した場合です
-1	関数の呼び出しに失敗した場合です。tperrnoにエラーコードが設定されます

- エラー

tmax_gq_put()が正常に実行されなかった場合、tperrno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
[TPEMATCH]	使用されたキー値がすでに存在する場合です
[TPELIMIT]	GQの最大キー数を超えた場合です

3.3.2. tmax_gq_get

GQからデータを取得する関数です。キーを指定した場合、該当キーのデータを取得します。基本的に、tmax_gq_get()を実行するとキューからデータが削除されます。データを保存するには、TPSQ_PEEKフラグを設定する必要があります。

- プロトタイプ

```
#include <tmaxapi.h>
int tmax_gq_get(char *key, long keylen1, char **data, long *len1, long flags1)
```

- パラメータ

パラメータ	説明
key	GQから取得するデータのためのキー値を保存するバッファです
keylen1	キー・バッファのサイズを指定します(単位: バイト)
data	GQから取得するデータ・バッファのポインターです。tpalloc()で割り当てられたバッファのポインターである必要があります
len1	GQから取得したデータ・バッファのサイズが保存されます(単位: バイト)
flags1	TPSQ_PEEKが指定できます。このフラグを指定した場合、データをキューから削除しません

- 戻り値

戻り値	説明
1	関数の呼び出しに成功した場合です
-1	関数の呼び出しに失敗した場合です。tperrnoにエラーコードが設定されます

- エラー

tmax_gq_get()が正常に実行されなかった場合、tperrno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
[TPEMATCH]	指定したキーのデータが存在しない場合です

3.3.3. tmax_gq_count

GQに保存されたデータ数を返す関数です。

- プロトタイプ

```
#include <tmaxapi.h>
int tmax_gq_count (void)
```

- 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
-1	関数の呼び出しに失敗した場合です。tperrnoにエラーコードが設定されます

- エラー

tmax_gq_count()が正常に実行されなかった場合、tperrno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
[TPENOENT]	現行セッションのためのSQが存在しない場合です

3.3.4. tmax_gq_purge

GQのデータを削除する関数です。キーを指定した場合は該当キーのデータを削除し、キーを指定していない場合はすべてのデータを削除します。

- プロトタイプ

```
#include <tmaxapi.h>
int tmax_gq_purge(char *key, long keylen1)
```

- パラメータ

パラメータ	説明
key	GQから削除するデータのためのキー値を保存するバッファです。NULLの場合、GQのすべてのデータを削除します
keylen1	キー・バッファのサイズを指定します(単位: バイト)

- 戻り値

戻り値	説明
削除されたデータ数	関数の呼び出しに成功した場合です
-1	関数の呼び出しに成功した場合です。tperrnoにエラーコードが設定されます

- エラー

tmax_gq_purge()が正常に実行されなかった場合、tperrno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
[TPEINVAL]	入力したキーが正しくない場合です
[TPENOENT]	現行セッションのためのSQが存在しない場合です

3.3.5. tmax_gq_keygen

システム・キーを作成および取得する関数です。システム・キーのサイズは、SQ_SYSKEY_SIZE(16バイト)です。バッファ・サイズはシステム・キーのサイズより大きい値か、同一の値を割り当てる必要があります。

- プロトタイプ

```
#include <tmaxapi.h>
int tmax_gq_keygen(char *key, long keylenl)
```

- パラメータ

パラメータ	説明
key	生成されたシステム・キー値が保存されるバッファです。バッファのサイズは、SQ_SYSKEY_SIZE(16バイト)より大きい値を割り当てる必要があります
keylenl	キー・バッファのサイズを指定します。必ずSQ_SYSKEY_SIZE(16バイト)より大きい値を指定します

- 戻り値

戻り値	説明
1	関数の呼び出しに成功した場合です
-1	関数の呼び出しに失敗した場合です。tperrnoにエラーコードが設定されます

- エラー

tmax_gq_keygen()が正常に実行されなかった場合、tperrno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
[TPEINVAL]	入力したキーが正しくない場合です

3.3.6. tmax_gq_getkeylist

GQのキー・リストを取得する関数です。キー・リストは、SQ_KEYLIST_Tタイプのハンドルが返されます。各キーについての情報は、tmax_keylist_count()、tmax_keylist_count()、tmax_keylist_free()関数を使用して確認できます。キー・リストはバイト単位で、ASCIIコード値を比較して低い値順にソートされます。

キー値を指定した場合、該当キー値より大きいキー・リスト、もしくは同一のキー・リストがソートされて照会されます。キー値をNULLと指定した場合、全体キー・リストが照会されます。

- プロトタイプ

```
#include <tmaxapi.h>
SQ_KEYLIST_T tmax_gq_getkeylist(char *key, long keylen1)
```

- パラメータ

パラメータ	説明
key	キー値を保存するバッファです
keylen1	キー・バッファのサイズを指定します(単位: バイト)

- 戻り値

戻り値	説明
SQ_KEYLIST_T	関数の呼び出しに成功した場合です
NULL	関数の呼び出しに失敗した場合です。tperno1にエラーコードが設定されます

- エラー

tmax_gq_getkeylist()が正常に実行されなかった場合、tperno1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
[TPEMATCH]	指定したキーより大きいキー、もしくは同一のキーが存在しない場合です

3.4. キーおよびセッション照会API

3.4.1. tmax_get_sessionid

現行セッションのIDを返す関数です。

- プロトタイプ

```
#include <tmaxapi.h>
int tmax_get_sessionid(void)
```

- 戻り値

戻り値	説明
-1ではない値	関数の呼び出しに成功した場合です
-1	関数の呼び出しに失敗した場合です。tperno1にエラーコードが設定されます

- エラー

tmax_get_sessionid()が正常に実行されなかった場合、tperno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
[TPEPROTO]	セッションが終了している場合です

3.4.2. tmax_keylist_count

keylistハンドルからキー・リストの数を返す関数です。

- プロトタイプ

```
#include <tmaxapi.h>
int tmax_keylist_count(SQ_KEYLIST_T keylist)
```

- パラメータ

パラメータ	説明
keylist	tmax_sq_getkeylist()またはtmax_gri_getkeylist()から渡されたハンドルです

- 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
-1	関数の呼び出しに失敗した場合です。tpernoにエラーコードが設定されます

- エラー

tmax_keylist_count()が正常に実行されなかった場合、tperno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
[TPEBADDESC]	keylistハンドルが正しくない場合です

3.4.3. tmax_keylist_getakey

keylistハンドルからn番目のキー情報を取得する関数です。

- プロトタイプ

```
#include <tmaxapi.h>
int tmax_keylist_getakey(SQ_KEYLIST_T keylist, int nth, SQ_KEYINFO_T *keyinfo)
```

- パラメータ

パラメータ	説明
keylist	tmax_sq_getkeylist() または tmax_gq_getkeylist() から渡されたハンドルです
nth	keylistハンドルからn番目のキーです。nは0からtmax_keylist_count() -1の間の値である必要があります
keyinfo	n番目のキー情報が保存される構造体です。詳細は、下の内容を参照してください

以下は、keyinfoについての説明です。

```
Structure keyinfo {
    long keylen
    long datalen
    time_t starttime
    char *key
};
```

メンバー	説明
long keylen	キーのサイズです(単位: バイト)
long datalen	データのサイズです(単位: バイト)
time_t starttime	データが保存またはアップデートされた時間です
char *key	キー値を保存しているバッファです

- 戻り値

戻り値	説明
1	関数の呼び出しに成功した場合です
-1	関数の呼び出しに失敗した場合です。tpernoにエラーコードが設定されます

- エラー

tmax_keylist_getakey()が正常に実行されなかった場合、tperno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
[TPEBADDESC]	keylistハンドルが正しくない場合です
[TPELIMIT]	nthの範囲を超えた場合です

3.4.4. tmax_keylist_free

keylistハンドルのメモリーやその他のリソースを解除する関数です。

- プロトタイプ

```
#include <tmaxapi.h>
int tmax_keylist_free(SQ_KEYLIST_T keylist)
```

- パラメータ

パラメータ	説明
keylist	tmax_sq_getkeylist()またはtmax_gq_getkeylist()から渡されたハンドルです

- 戻り値

戻り値	説明
-1ではない値	関数の呼び出しに成功した場合です
-1	関数の呼び出しに失敗した場合です。tperrnoにエラーコードが設定されます

- エラー

tmax_keylist_free()が正常に実行されなかった場合、tperrno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
[TPEBADDESC]	keylistハンドルが正しくない場合です

第4章 例

本章では、SQシステムを使用するクライアントとサーバーの例について説明します

4.1. クライアント・プログラム

以下は、クライアント・プログラムの例です。

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

// 入力するキーの数
#define KEYCOUNT      3

int get_keylist(char [KEYCOUNT][SQ_SYSKEY_SIZE+1]);

int main(int argc, char *argv[])
{
    char *sqbuf, *sndbuf, sqbuf_org[1024];
    char put_key[KEYCOUNT][SQ_SYSKEY_SIZE+1];
    char get_key[KEYCOUNT][SQ_SYSKEY_SIZE+1];
    long sqbuf_len = 0, rlen, flags;
    int ret, i, len=0, sys_flag=0;
    char key[SQ_SYSKEY_SIZE+1];

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

    // Tmaxシステムに接続
    if (tpstart((TPSTART_T *)NULL) == -1){
        printf("tpstart failed [%s]\n", tpstrerror(tperrno));
        return -1;
    }

    // セッションキューに入力するデータの割り当て (必ずtpallocで割り当てます)
    if ((sqbuf = (char *)tpalloc("STRING", NULL, 0)) == NULL) {
        printf("sqbuf alloc failed [%s]\n", tpstrerror(tperrno));
    }
}
```

```

        tpend();
        return -1;
    }

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

    if (argc != 3)
    {
        printf("Usage : %s Key Data\n", argv[0]);
        printf("          Key      : \"SYSTEM\" or Specific Key Value\n");
        exit(1);
    }

    strcpy(key, argv[1]);
    strcpy(sqbuf_org, argv[2]);

    // tmax_sq_putの際に使用するフラグの初期化
    flags = 0L;

    // システムキーを使用する場合
    if(!strcmp(key, "SYSTEM"))
    {
        flags |= TPSQ_SYSKEY;
        flags |= TPSQ_KEYGEN;
        len = SQ_SYSKEY_SIZE;
        sys_flag = 1;
    }
    else
    {
        strcpy(put_key[0], key);
        len = strlen(key);
    }

    // キー値が重複する場合、既存のデータをアップデートします。このフラグが設定されないと

    // キー値の重複エラー
    flags |= TPSQ_UPDATE;

    for(i=0; i<KEYCOUNT; i++)
    {
        if( !sys_flag )
        {
            memset(put_key[i], 0x00, SQ_SYSKEY_SIZE+1);

```

```

        sprintf(put_key[i], "%s%d", key, i);
    }
    memset(sqbuf, 0x00, 1024);
    sprintf(sqbuf, "%s%d", sqbuf_org, i);

    // セッションキューにデータを入力します。
    ret = tmax_sq_put(put_key[i], len+1, sqbuf, strlen(sqbuf), flags);

    if(ret < 0)
    {
        printf("tmax_sq_put error, key = %s [%s]\n", put_key[i],

                tpstrerror(tperrno));
        tpfree(sqbuf);
        tpend();
        exit(1);
    }
    // キー、データ値の出力
    printf("PUT : [%d] Key = [%s] , Data = [%s]\n", i, put_key[i],
sqbuf);

    }
    printf("\n");

    // Tmaxサービスの呼び出し
    ret = tpcall("GET_SQ", (char *)sndbuf, 0, (char **)&sndbuf, &rlen,
        TPNOFLAGS);
    if(ret < 0)
    {
        printf("tpcall failed [%s]\n", tpstrerror(tperrno));

        tpfree((char *)sndbuf);
        tpfree((char *)sqbuf);
        tpend();
        exit(1);
    }
    printf("recv data: %s\n", sndbuf);

    tpfree((char *)sqbuf);
    tpfree((char *)sndbuf);
    tpend();
    return 0;
}

```

4.2. サーバー・プログラム

以下は、サーバー・プログラムの例です。

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

#define MAXKEYCNT 10

GET_SQ(TPSVCINFO *msg)
{
    char get_key[MAXKEYCNT][SQ_SYSKEY_SIZE+1];
    int count;

    count = get_keylist(get_key);

    sprintf(msg->data, "GET_SQ is done, count=[%d]", count);

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

int get_keylist(char get_key[MAXKEYCNT][SQ_SYSKEY_SIZE])
{
    char *sqbuf = NULL;
    SQ_KEYLIST_T sqh;
    SQ_KEYINFO_T keyinfo;
    long sqbuf_len = 0;
    int count, i, j, ret;

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

    // SQに保存されているキーリストを取得します。
    sqh = tmax_sq_getkeylist(NULL, 0);
    if(sqh != NULL)
    {
        // キー数
        count = tmax_keylist_count(sqh);

        for(i=0; i<count; i++)
        {
            memset((char *)&keyinfo, 0x00, sizeof(keyinfo));
            // i 順番に該当するキー値を取得して保存
            tmax_keylist_getakey(sqh, i, &keyinfo);
            memset(get_key[i], 0x00, SQ_SYSKEY_SIZE);
        }
    }
}
```



```

        for(j=0; j< keyinfo.keylen; j++)
            get_key[i][j] = keyinfo.key[j];
        get_key[i][j] = 0;

        // SQでキー値に該当するデータを保存
        ret = tmax_sq_get(get_key[i], strlen(get_key[i]),
            (char **)&sqbuf, &sqbuf_len, 0);
        if(ret < 0)
        {
            printf("tmax_sq_get failed [%s]\n", t
                pstrerror(tperrno));
            tpfree(sqbuf);
            return -1;
        }
        printf("GET : [%d] Key = [%s] , Data = [%s]\n", i,
            get_key[i], sqbuf);
    }
    printf("\n");

    // sqhを解除
    tmax_keylist_free(sqh);
}

return i;
}

```


付録 A. ヘッダー・ファイル

より詳しい情報を必要とするユーザーのために、SQ APIのプロトタイプとシステム変数を宣言したヘッダー・ファイルの例を紹介します。Tmaxが提供するヘッダー・ファイルは、以下のパスに存在しています。

```
TMAXDIR/usrinc
```

A.1. tmaxapi.h

```
/* ----- usrinc/tmaxapi.h ----- */
/*
                                     */
/*      Copyright (c) 2000 - 2004 Tmax Soft Co., Ltd      */
/*      All Rights Reserved                                     */
/*
                                     */
/* ----- */

#ifndef _TMAXAPI_H
#define _TMAXAPI_H

#ifndef _CE_MODULE
#include <sys/types.h>
#endif
#include <usrinc/atmi.h>
#ifdef _WIN32
#ifdef _CE_MODULE
#include <winsock.h>
#else
#include <winsock2.h>
#endif /* _CE_MODULE */
#include <usrinc/svct.h>
#include <usrinc/sdl.h>
#else
#ifndef ORA_PROC
#include <sys/socket.h>
#endif
#endif

/* client logout type */
#define CLIENT_CLOSE_NORMAL      0
#define CLIENT_CLOSE_ABNORMAL   1
#define CLIENT_PRUNED           2
```

```

/* RQ Sub-queue type */
#define TMAX_ANY_QUEUE          0
#define TMAX_FAIL_QUEUE        1
#define TMAX_REQ_QUEUE         2
#define TMAX_RPLY_QUEUE        3
#define TMAX_MAX_QUEUE         4

extern char _rq_sub_queue_name[TMAX_MAX_QUEUE][XATMI_SERVICE_NAME_LENGTH];

/* RQ related macros */
#define RQ_NAME_LENGTH          16

/* unsolicited msg type */
#define UNSOL_TPPOST            1
#define UNSOL_TPBROADCAST      2
#define UNSOL_TPNOTIFY         3
#define UNSOL_TPSENDTOCLI      4
#define UNSOL_ANY              5

/* RM type */
#define ORACLE_TYPE             1
#define SYBASE_TYPE            2
#define INFORMIX_TYPE          3
#define DB2_TYPE               4

#define NONXA_MODE              0
#define XA_MODE                 1

/* Check SVCINFO cmds */
#define ISSVC_FORWARDED         0x00000001
#define ISSVC_NOREPLY           0x00000002

/* TPEVCTL ctl_flags */
#define TPEV_SVC                 0x00000001
#define TPEV_PROC                0x00000002

/* tpgethostaddr flags */
#define GET_SVR_CON              0x00000000
#define GET_CUR_IP              0x00000001

/* tmax_sq_get/tmax_sq_put flags */
#define TPSQ_PEEK                0x00001000
#define TPSQ_UPDATE              0x00002000
#define TPSQ_SYSKEY              0x00004000
#define TPSQ_KEYGEN              0x00008000

```

```

#define SQ_KEYLIST_T      void*
#define SQ_KEYINFO_T      sq_keyinfo_t
#define SQ_SYSKEY_SIZE    16

struct sq_keyinfo_s {
    long keylen;
    long datalen;
    time_t starttime;
    char *key;
};

typedef struct sq_keyinfo_s sq_keyinfo_t;

struct tpevctl {
    long ctl_flags;
    long post_flags;
    char svc[XATMI_SERVICE_NAME_LENGTH];
    char qname[RQ_NAME_LENGTH];
};

typedef struct tpevctl TPEVCTL;
typedef void __EXPORT Unsolfunc(char *, long, long);
#define TPUNSOLERR        ((Unsolfunc *) -1)

struct tptranid {
    int  info[3];
    int  flags;
};

typedef struct tptranid TPTRANID;

/* Multicast call related structures */
struct svglist {
    int    ns_entry;    /* number of entries of s_list */
    int    nf_entry;    /* number of entries of f_list */
    int *s_list;        /* list of server group numbers */
    int *f_list;        /* list of server group numbers */
};

/* My svrinfo */
#ifndef TMAX_NAME_SIZE
#define TMAX_NAME_SIZE    16
#endif

#ifndef MAX_DBSESSIONID_SIZE
#define MAX_DBSESSIONID_SIZE    128
#endif

```

```

typedef struct {
    int nodeno;      /* node index */
    int svgi;        /* server group index; unique in the node */
    int svri;        /* server index; unique in the node */
    int spri;        /* server process index; unique in the node */
    int spr_seqno;    /* server process seqno ; unique in the server */
    int min, max;     /* min/max server process number */
    int clhi;        /* for RDP only, corresponding CLH id */
    char nodename[TMAX_NAME_SIZE];
    char svgname[TMAX_NAME_SIZE];
    char svrname[TMAX_NAME_SIZE];
    char reserved_char[TMAX_NAME_SIZE];
    /* for more detail use tmaxadmin API */
} TMAXSVRINFO;

#define MSGIDLEN      32
#define CORRIDLEN     32

typedef struct {      /* control parameters to queue primitives */
    int flags;         /* indicates which of the values are set */
    int deq_time;      /* absolute/relative time for dequeuing */
    int priority;      /* enqueue priority */
    int diagnostic;    /* indicates reason for failure */
    char msgid[MSGIDLEN]; /* id of message before which to queue */
    char corrid[CORRIDLEN]; /* correlation id used to identify message */
    char replyqueue[TMAX_NAME_SIZE]; /* queue name for reply message */
    char failurequeue[TMAX_NAME_SIZE]; /* queue name for failure message */
    CLIENTID cltid;    /* client identifier for originating client */
    int urcode;        /* application user-return code */
    int appkey;        /* application authentication client key */
    int delivery_qos;
    int reply_qos;
    int exp_time;
} TMQCTL;

#ifdef _WIN32
typedef int (__EXPORT *WinTmaxCallback)(WPARAM, LPARAM);
#endif

/* Macro functions */
#define tpalivechk()    tmax_chk_conn(0)

#ifdef __cplusplus
extern "C" {
#endif

```

```

/* ----- unsolicited messaging API ----- */
long __EXPORT tpsubscribe(char *eventexpr, char *filter, TPEVCTL *ctl, long flags);
long __EXPORT tpsubscribe2(char *eventexpr, char *svcname, long flags);
int __EXPORT tpunsubscribe(long sd, long flags);
int __EXPORT tppost(char *eventname, char *data, long len, long flags);
int __EXPORT tpbroadcast(char *lnid, char *usrname, char *cltname, char *data,
                        long len, long flags);
Unsolfunc *__EXPORT tpsetunsol(Unsolfunc *func);
int __EXPORT tpsetunsol_flag(int flag);
int __EXPORT tpgetunsol(int type, char **data, long *len, long flags);
int __EXPORT tpclearunsol(void);
int __EXPORT tpchkunsol(void);

/* ----- RQS API ----- */
int __EXPORT tpenq(char *qname, char *svc, char *data, long len, long flags);
int __EXPORT tpdeq(char *qname, char *svc, char **data, long *len, long flags);
int __EXPORT tpenq_ctl(char *qname, char *svc, TMQCTL *ctl, char *data, long len,
                      long flags);
int __EXPORT tpdeq_ctl(char *qname, char *svc, TMQCTL *ctl, char **data, long *len,
                      long flags);
int __EXPORT tpqstat(char *qname, long type);
int __EXPORT tpqsvcstat(char *qname, char *svc, long type);
int __EXPORT tpextsvcname(char *data, char *svc);
int __EXPORT tpextsvcinfo(char *data, char *svc, int *type, int *errcode);
int __EXPORT tpreissue(char *qname, char *filter, long flags);
char *__EXPORT tpsubqname(int type);

/* ----- server API ----- */
int __EXPORT tpgetminsvr(void);
int __EXPORT tpgetmaxsvr(void);
int __EXPORT tpgetmaxuser(void);
int __EXPORT tpgetsvrseqno(void);
int __EXPORT tpgetmysvrid(void);
int __EXPORT tpgetmysvrno(void);
int __EXPORT tpgetmaxuser(void);
int __EXPORT tpsendtocli(int clid, char *data, long len, long flags);
int __EXPORT tpgetclid(void);
int __EXPORT tpgetpeer_ipaddr(struct sockaddr *name, int *namelen);
int __EXPORT tpchkclid(int clid);
int __EXPORT tmax_clh_maxuser(void);
int __EXPORT tmax_my_svrinfo(TMAXSVRINFO*);
int __EXPORT tmax_cind2clid(int cind);
char *__EXPORT tpgetmynode(int *nodeno);
char *__EXPORT tpgetmysvg(void);
int __EXPORT tpgetmysvgno(void);

```

```

int __EXPORT tmax_is_restarted(void);
int __EXPORT tpsuspendtx(TPTRANID *tranid, long flags);
int __EXPORT tpresetmetx(TPTRANID *tranid, long flags);

/* ----- etc API ----- */
int __EXPORT tp_sleep(int sec);
int __EXPORT tp_usleep(int usec);
int __EXPORT tpset_timeout(int sec);
int __EXPORT tpget_timeout(void);
int __EXPORT tmaxreadenv(char *file, char *label);
char *__EXPORT tpgetenv(char* str);
int __EXPORT tpputenv(char* str);
int __EXPORT tpgetsockname(struct sockaddr *name, int *namelen);
int __EXPORT tpgetpeername(struct sockaddr *name, int *namelen);
int __EXPORT tpgetactivesvr(char *nodename, char **outbufp);
int __EXPORT tperrordetail(int i);
int __EXPORT tpreset(void);
int __EXPORT tptobackup(void);
struct svglist *__EXPORT
    tpmcall(char *qname, char *svc, char *data, long len, long flags);
struct svglist *__EXPORT tpgetsvglist(char *svc, long flags);
int __EXPORT tpsvgcall(int svgno, char *qname,
    char *svc, char *data, long len, long flags);
int __EXPORT tpflush(void);
char *__EXPORT tmaxlastsvc(char *idata, char *odata, long flags);
int __EXPORT tpgetorgnode(int clid);
int __EXPORT tpgetorgclh(int clid);
char *__EXPORT tpgetnodename(int nodeno);
int __EXPORT tpgetnodeno(char *nodename);
int __EXPORT tpgetasize(char *data);
int __EXPORT tpgettype(char *data);
char *__EXPORT tpgetsubtype(char *data);
int __EXPORT tpgetcliaddr(int clid, int *ip, int *port, long flags);
int __EXPORT tmax_chk_conn(int timeout);
int __EXPORT tpgethostaddr(int *ip, int *port, long flags);
char *__EXPORT tpgetdbsessionid(int flags);
int __EXPORT tpcallsvg(int svgno, char *svc, char *idata, long ilen,
    char **odata, long *olen, long flags);
int __EXPORT tpacallsvg(int svgno, char *svc, char *data, long len, long flags);

#ifdef _WIN32
int __EXPORT WinTmaxAcall(TPSTART_T *sinfo, HANDLE wHandle, unsigned int msgType,

    char *svc, char *sndbuf, int len, int flags);
int __EXPORT WinTmaxAcall2(TPSTART_T *sinfo, WinTmaxCallback fn,
    char *svc, char *sndbuf, int len, int flags);
#endif

```



```

#if !defined(_TMAX_KERNEL) && !defined(_TMAX_RCA_H)
/* ----- User supplied routines ----- */
int __EXPORT tpsvrinit(int argc, char *argv[]);
int __EXPORT tpsvrdone(void);
void __EXPORT tpsvctimeout(TPSVCINFO *msg);
int __EXPORT _tmax_event_handler(char *progname, int pid, int tid, char *msg,
                                int flags);
int __EXPORT tpsetdbsessionid(char dbsessionid[MAX_DBSESSIONID_SIZE], int flags);
int __EXPORT tpprechk(void);
#endif

/*
   Internal functions: ONLY BE CALLED FROM AUTOMATICALLY
   GENERATED STUB FILES. DO NOT DIRECTLY CALL THESE FUNCTIONS.
*/
int __EXPORT get_clhfd(void);
int __EXPORT tmax_chk_svcinfo(int cmd);
int __EXPORT _tmax_main(int argc, char *argv[]);
int __EXPORT _tmax_cob_main(int argc, char *argv[]);
#if defined(_WIN32)
int __EXPORT _tmax_regfn(void *initFn, void *doneFn, void *timeoutFn,
                        void *userMainFn);
int __EXPORT _tmax_regtab(int svcTabSz, _svc_t *svcTab, int funcTabSz,
                        void *funcTab);
int __EXPORT _tmax_regSDL(int _sdl_table_size2, struct _sdl_struct_s *_sdl_table2,

                        int _sdl_field_table_size2, struct _sdl_field_s *_sdl_field_table2);
int __EXPORT _tmax_regevtHnd(void *evtHndFn);
#endif
int __EXPORT _double_encode(char *in, char *out);
int __EXPORT _double_decode(char *in, char *out);
/* --- power builder interface API --- */
int __EXPORT _make_struct_from_pbindata(char *subtype, char *tpidata, int ilen,
                                       char *indata);
int __EXPORT _make_field_from_pbindata(char **tpidata, char *indata);
int __EXPORT _make_pdata_from_struct(char *subtype, char *odata, int olen,
                                       char *tpodata);
int __EXPORT _make_pdata_from_field(char *form, char *odata, char *tpodata);
int __EXPORT _make_pbfodata_from_field(char *fform, char *fodata, char *tpodata);
int __EXPORT _get_value_from_pbsdata(char *cur, char *vald);
int __EXPORT _get_name_value_from_pbfdata(char *cur, int *n2);
int __EXPORT _get_name_from_form(char *cur);
int __EXPORT _insert_value_to_pdata(int type, char *out, char *in,
                                     int asize, int asize2);
int __EXPORT tmax_get_db_usrname(char *svgname, char *usrname, int type);
int __EXPORT tmax_get_db_passwd(char *svgname, char *passwd, int type);

```

```

int __EXPORT tmax_get_db_tnsname(char *svgname, char *tnsname, int type);

int __EXPORT tmax_is_xa();

/* 4.0 Sp2 Fix1 added */
int __EXPORT tmax_get_svccnt();
int __EXPORT tmax_get_svclist(char *buf, int bufsize);

/* ----- SessionQ API ----- */
int __EXPORT tmax_sq_put(char *key, long keylenl, char *data, long lenl,
                        long flagsl);
int __EXPORT tmax_sq_get(char *key, long keylenl, char **data, long *lenl,
                        long flagsl);
int __EXPORT tmax_sq_count(void);
int __EXPORT tmax_sq_purge(char *key, long keylenl);
int __EXPORT tmax_sq_keygen(char *key, long keylenl);
SQ_KEYLIST_T __EXPORT tmax_sq_getkeylist(char *key, long keylenl);

int __EXPORT tmax_gq_put(char *key, long keylenl, char *data, long lenl,
                        long flagsl);
int __EXPORT tmax_gq_get(char *key, long keylenl, char **data, long *lenl,
                        long flagsl);
int __EXPORT tmax_gq_count();
int __EXPORT tmax_gq_purge(char *key, long keylenl);
int __EXPORT tmax_gq_keygen(char *key, long keylenl);
SQ_KEYLIST_T __EXPORT tmax_gq_getkeylist(char *key, long keylenl);

int __EXPORT tmax_get_sessionid(void);
int __EXPORT tmax_keylist_count(SQ_KEYLIST_T keylist);
int __EXPORT tmax_keylist_getakey(SQ_KEYLIST_T keylist, int nth,
                                SQ_KEYINFO_T *keyinfo);
int __EXPORT tmax_keylist_free(SQ_KEYLIST_T keylist);

#ifdef (__cplusplus)
}
#endif

#endif /* end of _TMAXAPI_H */

```

索引

G

GQ(Global Queue), 2

N

NODEセクション

Node Name, 5

SQKEY, 6

SQKEYMAX, 6

SQMAX, 6

SQSIZE, 6

SQTIMEOUT, 6

R

RQ API, 2

S

SQ, 1

SQ(Session Queue), 2

sqinfo(sqi), 7

sqinfo(sqi)オブション

[-k], 7

sqpurge(sq), 9

SQS, 2

SQ環境設定

NODEセクション, 5

T

tmax_get_sessionid, 22

tmax_gq_count, 19

tmax_gq_get, 18

tmax_gq_getkeylist, 21

tmax_gq_keygen, 21

tmax_gq_purge, 20

tmax_gq_put, 17

tmax_keylist_count, 23

tmax_keylist_free, 25

tmax_keylist_getakey, 23

tmax_sq_count, 14

tmax_sq_get, 13

tmax_sq_getkeylist, 16

tmax_sq_keygen, 15

tmax_sq_purge, 15

tmax_sq_put, 12

Tmaxセッション, 1

さ

システムキー(SYSKEY)の使用方式, 3

システム・ストレージ, 2

や

ユーザーキー(USERKEY)の使用方式, 3

