

Tmax プログラミングガイド (RQ)

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 (RQ)

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	1
1.3. RQシステム・フロー	4
第2章 使用および管理	7
2.1. 環境設定	7
2.1.1. SVRGROUPセクション	7
2.1.2. RQセクション	9
2.2. システム管理	12
2.2.1. 起動および終了	12
2.2.2. 状態情報管理	13
第3章 API	17
3.1. 概要	17
3.2. RQ使用API	18
3.2.1. tpenq	18
3.2.2. tpdeq	21
3.2.3. tpreissue	23
3.3. RQ使用補助API	25
3.3.1. tpextsvcname	25
3.3.2. tpextsvcinfo	26
3.3.3. tpqstat	28
3.3.4. tpqsvcstat	30
3.3.5. tpenq_ctl	31
3.3.6. tpdeq_ctl	36
3.4. その他	41
3.4.1. compute_rq_deqtime	41
第4章 サンプル	43
4.1. 基本プログラム	43
4.1.1. 環境設定	44
4.1.2. クライアント・プログラム	44
4.1.3. サーバー・プログラム	45
4.2. UCS使用プログラム	46
4.2.1. 環境設定	47
4.2.2. クライアント・プログラム	47
4.2.3. サーバー・プログラム	49
4.3. Failキュー復旧プログラム	50
4.3.1. 環境設定	51
4.3.2. サーバー・プログラム	51

付録 A. ヘッダー・ファイル	53
A.1. tmaxapi.h	53
索引	57

図目次

[図 1.1]	RQシステムの構成	1
[図 1.2]	システム・フェイルの場合のRQ	3
[図 1.3]	RQシステム・フロー	4
[図 4.1]	RQプログラムのプロセス	43
[図 4.2]	UCS使用プログラムのプロセス	46
[図 4.3]	Failキュー復旧プログラムのプロセス	50

このガイドについて

対象読者

本書は、Tmax[®](以下Tmax)のRQ(Reliable Queue)の概念と使用方法について記述したガイドであり、Tmax RQを使用して開発する開発者を対象としています。

前提知識

本章はTmaxシステムについての全般的な理解とTmaxシステムが提供する各種の機能および特性を習得するための基本ガイドです。

本書を理解するためには、以下の事項について熟知している必要があります。

- 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。実務における具体的な使用方法や管理および運用に関する事項は各製品の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』と『Tmax アプリケーション開発ガイド』を参照してください。Tmaxで提供するコマンドとC APIについての説明は『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つの章と付録から構成されています。

各章の主な内容は以下のとおりです。

- 第1章: 紹介

RQシステムの定義と構成について説明します。

- 第2章: 使用および管理

RQシステムをアプリケーション開発に適用するために、Tmax環境ファイルにRQを設定するプロセスとプログラム開発プロセスについて説明します。また、RQシステムを起動および終了する方法について説明します。

- 第3章: API

アプリケーション開発のためのAPIの使用方法について説明します。

- 第4章: サンプル

RQシステムを使用する基本例と応用例を紹介します。

- 付録A: ヘッダー・ファイル

RQ APIのプロトタイプとシステム変数を宣言するヘッダー・ファイルの内容を紹介します。

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
参考	参照/注意事項
[図 1.1]	図の名称
[表 1.1]	表の名称
AaBbCc123	コマンド、コマンド実行結果の画面出力、サンプル・コード
{ }	必須パラメータ値
[]	オプション・パラメータ値
	選択パラメータ値

システム要件

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

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RQ(Reliable Queue)の定義およびRQシステムの構成について説明します。

1.1. 概要

クライアントのサービス要求が集中して要求されたサービスを即時処理できず、要求が内部キューにたまっている状態でシステム障害やエラーによりシステムが異常終了する場合、すべてのサービスデータは削除されます。このような場合、信頼性の保証が求められる業務は正常なサービスが行われることができません。このような問題を補完するために、Tmaxシステムは対外機関との連携業務のように信頼性の保証が求められ、業務の特性上トランザクション時間が一般的なオンライン業務に比べて長いサービス进行处理するためにRQ (Reliable Queue)を提供します。

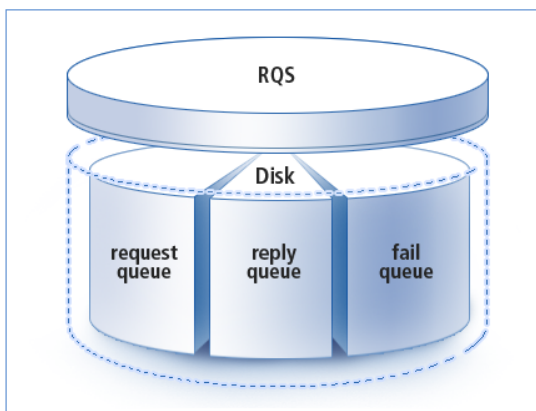
RQは安定的なサービスを保証するためにクライアントの要求をディスクに管理し、クライアント間あるいはサーバー間のPeer-to-peer通信に使用されます。

しかし、RQを使用すると信頼性は保証できますが、ディスクにデータを記録するためスピードは落ちる可能性があります。したがって、信頼性の保証が求められ、長い時間がかかる作業に限ってRQを使用することをお勧めします。

1.2. システム構成

Tmax RQ(Reliable Queue)システムは、クライアントが送信したサービス要求メッセージとサービス実行結果を物理的な媒体(ディスク)または仮想媒体(メモリー)に記録／読み込みし、運用・管理するすべてのAPIとコマンドを含みます。RQを使用することで、万が一の場合にもユーザーのサービス要求とサービス実行結果を失うことがなく、未処理された作業も再開できます。

[図 1.1] RQシステムの構成



RQシステムは以下の要素で構成されます。

- RQファイル・システム
- RQS
- RQ API
- 管理ツール

RQファイル・システム

Tmax RQシステムはRQを管理する実際のデータが保存される場所であり、特別なRawデバイスを使用しません。RQシステムが起動すると、Tmaxはユーザーが指定したディレクトリに16~2047MBのファイルを作成してRQデータ・ファイルとして使用します。

RQデータ・ファイルはTmax環境ファイルに指定したRQ名を利用して「RQ名.data」というファイル名で生成されます。ユーザーはRQデータ・ファイルのサイズや、システムの終了後データが残っているRQファイルを再使用するかどうかなどを指定できます。現在のTmaxバージョンではRQデータ・ファイルのサイズが自動調整されないため、ユーザーが適切な値を直接指定します。

RQはRequest、Reply、Failの3つのキューに分けられます。

- Requestキュー

サービスを実行する前にtpenq()処理された要求データはすべてRequestキューに保存されます。

- Replyキュー

正常にサービスを実行して返されたデータは、サービスの成功や失敗に関係なくすべてReplyキューに保存されます。例外として、サービス名を明示していないデータもここに保存されます。

- Failキュー

サービスの実行に失敗したり実行前に終了されたRQデータはFailキューに保存されます。

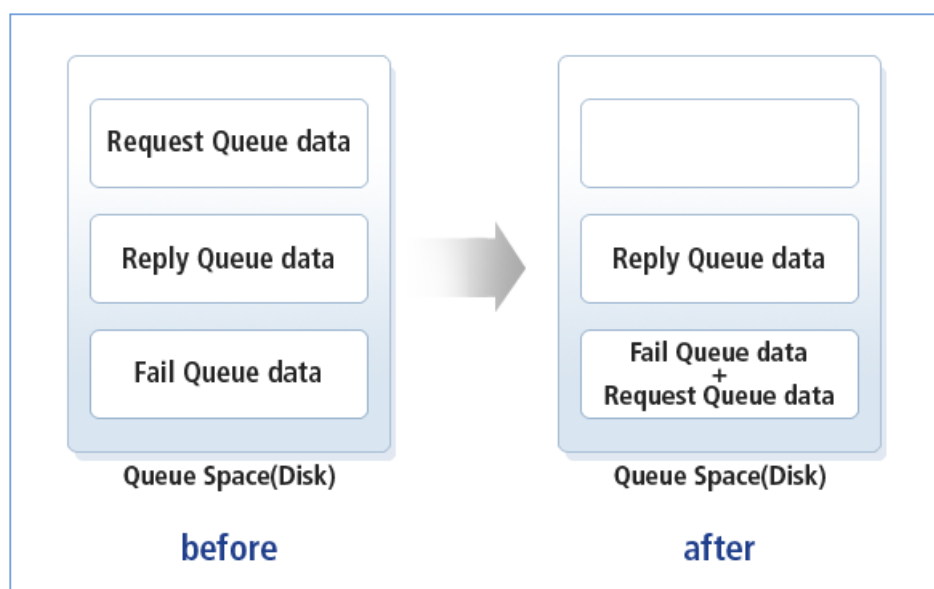
以下のような場合、Failキューにデータが保存されます。

- tpenq()処理されたデータがサーバーから正常なリターン結果を受けられなかった場合
- 当該サービスがない場合
- サービスで正常にリターンされなかった場合
- Requestキューにデータがあるのに、Tmaxシステムが再起動されるときのワームブートする場合

RQに保存されたデータはシステム障害およびネットワーク不安定などによりTmaxシステムが再起動された場合も再使用できます。データ復旧のためには環境ファイルの設定が必要です。環境ファイルの設定についての詳細は「[4.2. UCS使用プログラム](#)」の例を参照してください。

復旧の際に、RequestキューにたまっていたデータとFailキューにたまっていたデータはFailキューに移され、ReplyキューにたまっていたデータはそのままReplyキューに移されます。このデータは再び取り出されてサービスの実行などに使用できます。

[図 1.2] システム・フェイルの場合のRQ



RQS

RQSはRQマネージャーであり、CLH(クライアント・マネージャー)の制御を受け、実際RQファイルにデータを保存して読み込むすべてのプロセスを管理・制御します。RQSは独自のバッファを保持して、実際にディスクに保存されたデータと同期化されたデータを維持し、RQを使用するサービスの実行においてキャッシュ機能を提供します。

ユーザーはRQSがディスクにデータを保存する方法を指定できます。

RQ API

RQシステムを使用するアプリケーション向けのAPIを提供します。APIについての詳細は「[第3章 API](#)」を参照してください。

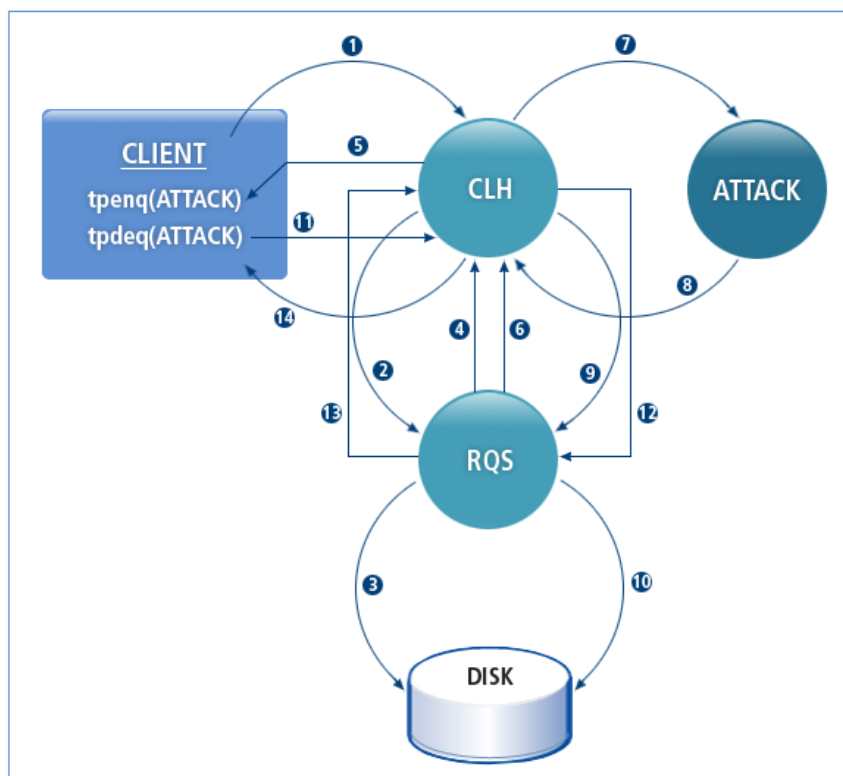
管理ツール

管理者はTmaxが提供するtmaxadminを利用してRQシステムの状態情報を照会または管理することができます。tmaxadminを用いたRQシステム管理についての詳細は、「[2.2. システム管理](#)」を参照してください。

1.3. RQシステム・フロー

基本的なRQシステム・フローは以下のとおりです。クライアントはRQを介してサービスを要求し、その結果値を受信する一般的な使用法を表示しています。

【図 1.3】 RQシステム・フロー



以下はRQシステム・フローの順序についての説明です。

1. クライアントがATTACKというサービスを`tpenq()`で要求します。
2. クライアントの要求を受信したTmaxシステム・ハンドラー(以下CLH)は`tpenq()`に対する処理をRQS(RQ マネージャー)に依頼します。
3. RQSは受信したサービスをRequestキュー(Disk)に記録します。
4. RQSは正常にRequestキューにサービスを入力したことをCLHに送信します。
5. CLHはサービス要求に対する応答が正常に処理されたという結果をクライアントに送信します。

6. RQSはCLHIにサービスの処理を要求します。
7. CLHIは要求を受信したサービス(ATTACK)を実行します。
8. 当該サービス(ATTCK)がルーチンを実行した後tpreturn()を行います。
9. サービス(ATTCK)の結果を受信したCLHIはその結果をRQSに送信します。
- 10 サービス処理結果を受信したRQSは正常に処理された場合はReplyキューに記録し、失敗した場合にはFailキューに記録します。
11. クライアントは当該サービス名(ATTACK)でtpdeq()を要求します。
- 12 CLHIはRQSに処理結果を要求します。
- 13 RQSはディスク内容の最新情報をメモリーに維持することで、追加のディスクI/OなしでCLHIに応答を送信します。
- 14 CLHIは応答をクライアントに送信します。

第2章 使用および管理

RQシステムをアプリケーションの開発に適用するためには、Tmax環境ファイルにRQを設定するプロセスと実際にプログラムを開発する2つのプロセスが必要です。本章ではこれらのプロセスについて説明し、様々な例をあげてアプリケーションの開発におけるRQの応用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、RQシステムを開始するためには基本的な設定が必要です。RQシステムはサーバー・グループ単位で管理されるため、Tmax環境ファイルのSVRGROUPセクションとRQセクションを適切に定義する必要があります。

参考

Tmax環境ファイルの設定に関する詳細は、『Tmax 運用ガイド』を参照してください。

2.1.1. SVRGROUPセクション

RQシステムを使用するためには、RQが属するサーバー・グループを定義する必要があります。そのためにSVRGROUPセクションの基本必須項目以外に、SVGTYPE項目を「RQMGR」に指定します。その他にもCPCなどを設定してシステムの運用効率を高めることができます。

SVRGROUPセクションの環境設定フォーマットは以下のように定義します。

```
*SVRGROUP
SVRGROUP Name      NODENAME = node-name ,
                    CPC = number ,
                    SVGTYPE = [RQMGR|TXRQMGR] ,
                    TMSNAME= "POD"
```

- SVRGROUP Name = string
 - サイズ: 63字以内
 - サーバー・グループの名前を設定します。SVRGROUPセクション内で一意な名前である必要があります。
- NODENAME = string

- サーバー・グループが属するノード名を設定します。NODEセクションで定義したノード名である必要があります。
- CPC = numeric
 - 範囲: 4~128
 - RQSプロセスとCLHプロセス間の並列通信チャンネル数を指定します。RQを使用しない場合は無視されます。RQを頻繁に使用する場合、マルチチャンネルを維持することで処理速度を向上することができます。
- SVGTYPE = [RQMGR|TXRQMGR]
 - サーバー・グループのタイプを定義します。

設定値	説明
RQMGR	RQを使用する場合に設定します
TXRQMGR	<p>トランザクション環境でRQを使用する場合に設定します。</p> <p>以下はトランザクション環境設定に関する注意事項です。</p> <ul style="list-style-type: none"> - RQサーバー・グループのタイプ(SVGTYPE)を「RQMGR」の代わりに「TXRQMGR」に設定します。 - 従来のtpenq/tpdeq APIはトランザクションをサポートしません。 - tpenq_ctlとtpdeq_ctlは1つのトランザクションとして扱うことができません。 - tpenq_ctlをトランザクションとして扱う場合、RQに保存するプロセスまでをトランザクションとして処理します。svc callについてはトランザクションをサポートしません。 - deq_timeを設定した場合、設定時間の後にsvc callをします。「現在時間 + delay_time」に設定します。 <p>delay_timeを10に与えたい場合、以下のように設定します。</p> <pre>time(&t); ctl->deq_time = t + 10;</pre> <ul style="list-style-type: none"> - deq_timeはトランザクションと一緒に使用できません。

- TMSNAME = "POD"

- RQがPODサーバーのように動作します。TMSNAMEフィールドに"POD"を入力した場合、tpeq要求に対してのみ起動を行います。

使用例

```
*SVRGROUP
svg1      NODENAME = tmax1,
          CPC = 4,
          SVGTYPE=RQMGR
```

以下は、トランザクション環境でRQを使用する場合の環境設定の例です。

```
*SVRGROUP
Txrqsvg   NODENAME = tmaxh4,
          CPC = 4 ,
          SVGTYPE = TXRQMGR
```

2.1.2. RQセクション

各サーバー・グループは1つ以上のRQを保持して使用できます。RQセクションにはRQの管理方法を設定します。RQセクションの項目には、QSIZE、FILEPATH、BOOT、FSYNC、BUFFERINGなどがあります。このうち、**SVGNAME**は必ず設定する必要があります。

RQセクションの環境設定フォーマットは以下のように定義します。

```
*RQ
RQ Name   SVGNAME = server-group-name,
          BOOT = COLD | WARM,
          QSIZE = number,
          BUFFERING = Y | N ,
          FSYNC = Y | N ,
          FILEPATH = file-path,
          MAX_MEMQCOUNT = number
```

- RQ Name = string
 - サイズ: 14字以内
 - RQセクションの論理的な名前です。RQセクション内で複数のRQ定義が可能であり、それぞれのRQ名は一意的な値である必要があります。また、1つのサーバー・グループに1つ以上のRQを定義できます。
- SVGNAME = string
 - RQを使用するサーバー・グループ名を設定します

– SVRGROUPセクションで定義したサーバー・グループ名である必要があります。

- BOOT = COLD | WARM

– デフォルト値: COLD

– BOOTフィールドはTmaxシステムが再起動されるときRQに保存されているデータを調整します。

– 以下は設定可能な値です。

設定値	説明
WARM	システム障害復旧の後、キューに溜まっているデータの復旧が可能になります
COLD	Tmaxシステムが再起動されるとき、ディスクに保存されているデータが削除されます

- QSIZE = numeric

– 範囲: 1~2047MB

– デフォルト値: 16MB

– RQのサイズを指定して使用できます。RQは基本的にTMAXDIRのpathディレクトリーに生成され、サービス処理を継続した結果、指定されたキューのサイズを超えた場合には、tpeq()時にTPEQFULLエラーが設定されます。RQのサイズは自動で増加しないため、RQが定義したサイズを越えてエラーが発生することを防ぐためには、適切なサイズを設定する必要があります。RQファイルを作成した後もRQデータが保存されない場合は、Tmaxが終了時にファイルが自動で削除されます。

- BUFFERING = Y | N

– デフォルト値: Y

– RQファイル内容をメモリーにキャッシュするかどうかを指定します。

– 以下は設定可能な値です。

設定値	意味
N	RQ処理が相対的に遅くなりますが、要求されるメモリーは小さくなります

- FSYNC = Y | N

– デフォルト値: Y

- BUFFERINGを「N」に設定しても、オペレーティング・システムはデータを即時ディスクに保存するわけではなく、最長30秒間メモリーに保管します。この待機時間を省いて即時メモリーのデータをディスクに保存させる場合に使用します。
- 以下は設定可能な値です。

設定値	意味
N	システム障害が発生した場合にデータを失う可能性があります、RQ処理速度は速くなります
Y	RQデータがメモリーを使用せずに常にディスクに安全に保存されます。ただし、Nを設定したときより処理速度は遅くなります

- FILEPATH = literal

- デフォルト値: \$(TMAXDIR)/path/RQ名.data
- RQで使用するファイルの場所を指定します。このフィールドはRQデータ・ファイルを作成し、このファイルは全体ファイル名とディレクトリー名で定義します。

参考

TMAXDIRはTmaxをインストールしたホーム・ディレクトリーです。以降、Tmaxホーム・ディレクトリーをTMAXDIRと記述します。

- MAX_MEMQCOUNT = numeric

- デフォルト値: 1024
- RQでTPRQS_NON_PERSISTENTタイプを指定した場合、つまりファイルにデータを記録せずにメモリーにのみデータを保存する場合に保存できるメッセージの最大数です。

使用例

```
*RQ
rqtest  SVGNAME = svg1,
        BOOT = COLD,
        QSIZE = 1024,
        BUFFERING = Y,
        FSYNC = Y,
        FILEPATH = "/user1/tmax/myrq.data"
```

2.2. システム管理

RQシステムの管理は一般Tmaxシステムの管理方法とほぼ変わりありません。管理者はRQシステムを別途起動および終了したり、tmadminを使用してRQの状態情報を確認したりできます。

2.2.1. 起動および終了

tmboot

RQシステムはtmbootコマンドでTmaxシステムと一緒に起動します。

- 使用方法

```
$ tmboot [ -q SVGNAME ]
```

- オプション

オプション	意味
[-q SVGNAME]	引数として渡されたRQサーバー・グループだけ別途起動します

- 例

- 以下のように[-T]オプションを使用した場合、Tmaxのみ起動され、RQSは起動されません。コマンドの実行結果は以下のとおりです。

```
$ tmboot -T

TMBOOT for node(tmaxs7) is starting:
  TMBOOT: TMM is starting: Thu Aug 23 16:29:16 2012
  TMBOOT: CLL is starting: Thu Aug 23 16:29:16 2012
  TMBOOT: CLH is starting: Thu Aug 23 16:29:16 2012
(I) CLH9991 Current Tmax Configuration: Number of client handler(MINCLH) = 1
      Supported maximum user per node = 16040
      Supported maximum user per handler = 16040 [CLH0141]
  TMBOOT: TLM(tlm) is starting: Thu Aug 23 16:29:16 2012
```

- 以下のように[-q]オプションを使用した場合、RQを管理するサーバー・グループに該当するRQSを起動できます。コマンドの実行結果は以下のとおりです。

```
$ tmboot -q svg1

TMBOOT for node(tmaxs7) is starting:
  TMBOOT: SVR(rqs) is starting: Thu Aug 23 16:33:22 2012
```

tmdown

RQシステムはtmdownコマンドで終了します。

- 使用方法

```
$ tmdown [ -q SVGNAME ]
```

- オプション

オプション	意味
[-q SVGNAME]	引数として渡されたRQサーバー・グループだけ終了させます

- 例

以下のように[-q]オプションを使用した場合、RQを管理するサーバー・グループに該当するRQSを終了できます。コマンドの実行結果は以下のとおりです。

```
$ tmdown -q svg1

TMDOWN for node(tmaxs7) is starting:
TMDOWN: RQS(_rqsvg:34) downed: Thu Aug 23 16:37:56 2012
```

2.2.2. 状態情報管理

Tmaxシステムが起動すると、Tmaxが提供する管理ツールのtmdadminを使用してシステムの状態を管理できます。

tmdadminの実行モードは、次の2種類があります。

- サブマネージャー・モード

システムの運用状態を確認することはできますが、システムの設定は変更できないモードです。以下のように実行します。

```
$ tmdadmin [-s]
```

- マスター・マネージャー・モード

システムの運用状態の確認と設定の変更が可能なモードです。以下のように実行します。

```
$ tmdadmin [-m]
```

参考

tmadminの実行モードについての詳細は、『Tmax 運用ガイド』を参照してください。

rqs

tmadminを実行した後、**rqs**コマンドを使って現在使用できるRQの状態を出力したり、ディスク・キューにたまっているデータを操作したりできます。

● 使用方法

```
$ rqs [ -l ] [ -s RQ名 ] [ -c RQ名 ] [ -f RQ名 ]
```

● オプション

オプション	説明
[-l]	Tmax環境ファイルのRQセクションに定義されたすべてのRQ名を出力します。サブ・マネージャー・モードで実行できます
[-s RQ名]	RQ名に指定されたRQの状態情報を出力します。Requestキュー、Replyキュー、Failキューの状態をすべて出力します。サブ・マネージャー・モードで実行できます
[-c RQ名]	停滞したRQを削除します。マスター・マネージャー・モードでのみ実行できます
[-f RQ名]	RQにたまっているサービスを処理します。マスター・マネージャー・モードでのみ実行できます

● 例

- 以下は、**[-l]**オプションを使用してコマンドを実行した例です。

```
$ tmaxsl (tmadm): rqs -l
list of available RQs:
      rqtest           rq2           rq3
```

- 以下は、**[-s]**オプションを使用してコマンドを実行した例です。

```
$ tmaxsl (tmadm): rqs -s rqtest
-----
Number of queue entries of RQ [rqtest]
-----
      request           0
      reply             20
      fail              0
```

- 以下は、**[-f]**オプションを使用してコマンドを実行した例です。


```
$ tmaxsl (tmaxsl): rqs -f rqtest  
RQ(rqtest) flushed
```


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

使用するAPIは、RQが使用するAPIとRQの使用を補助するAPIに分けれます。すべてのAPIはサーバーとクライアントで使用できます。

● RQ使用API

API	説明
tpenq	RQにデータを保存します
tpdeq	RQからデータをロードします
tpreissue	当該RQのFailキューにたまっている要求データを再びRequestキューに入れます

● RQ使用補助API

API	説明
tpextsvcname	tpdeq() でRQからデータを読み込んだ場合、当該データのサービス名を提供します
tpextsvcinfo	tpdeq() で RQからデータを読み込んだ場合、当該データの詳細情報を提供します
tpqstat	RQに保存されているデータの統計を要求します
tpqsvcstat	RQに保存されているデータのうち指定したサービスの統計を要求する関数で、現在RQにたまっているデータの統計を出します
tpenq_ctl	トランザクションをサポートし、RQデータを保存します
tpdeq_ctl	トランザクションをサポートし、RQからデータをロードします

● その他

API	説明
compute_rq_deqtime	deq_time関連のコールバック関数です

3.2. RQ使用API

3.2.1. tpenq

サーバーとクライアントでRQにデータを保存する関数です。Tmaxシステムは、システムの障害やエラーによるサービス不可能な状態でも、RQに保存されたデータは整合性を保証できます。RQにデータを保存しておき、様々な状況でシステムがダウンして、復旧後に再度実行された場合、まだ処理できていないデータを引き続き処理できます。

tpcall()や**tpacall()**でサービスを要求した場合、該当サービスが実行するデータが累積していると、サービスを要求したデータも待機(waiting)します。この際、システムの障害やエラーによってシステムがダウンした場合、待機中のデータは失われます。こういった問題点を補完し、データの整合性を保証できるように、**tpenq()**はサービスを要求する場合、データをRQに保存します。トランザクション・モードで実行しても、トランザクション・モードから除外されるため、トランザクション・モードで関数を実行中にエラーが発生してもトランザクションは影響を及ぼしません。

● プロトタイプ

```
# include <tmaxapi.h>
int tpenq (char *qname, char *svc, char *data, long len, long flags)
```

● パラメータ

パラメータ	説明
qname	qnameは、データを保存するRQの名前です。configファイルに登録された名前である必要があります
svc	データをRQに保存し、svc名がNULLでない場合、即時サービスを要求します。 svc名がNULLの場合、データはRQに保存され、サービスは実行されません。この場合、後にtpdeq()を使用してサービスを要求する必要があります。svcで命令されたサービスがない場合、またはサービスを実行して処理結果を受信していない状態でシステム障害が発生した場合、このデータは内部的にフェイル・キューに保存されます。データは tpdeq() でサービスを再要求するか、エラー処理をする必要があります
data	NULL値の場合を除き、必ずtpalloc()で割り当てられたバッファーに対するポインターである必要があります。dataのタイプとサブタイプは、svcがサポートするタイプである必要があります
len	送信するデータ長です – dataが指すバッファーが、特別な長さ明示が不要なタイプ (STRING、STRUCT、X_COMMON、X_C_TYPE) の場合、lenは無視され、0が使用されます – dataが指すバッファーが、長さを指定する必要があるタイプ (X_OCTET、CARRAY、MULTI STRUCTURE) の場合、lenは0になれません

パラメータ	説明
	<ul style="list-style-type: none"> dataがNULLの場合、lenは無視され、データなしでサービス要求が送信されます
flags	<p>flagsに使用可能な値は以下のとおりです</p> <ul style="list-style-type: none"> TPRQS <p>svcがNULLでない場合、svcに指定されたサービスを要求し、処理結果をRQに保存します。サービスの処理結果は、tpdeq()を利用して受信します。svcがNULLの場合、データはRQに保存され、サービスは実行しません</p> TPNOREPLY <p>svcがNULLでない場合、svcに指定されたサービスを要求しますが、処理結果はRQに保存しません。svcがNULLの場合、データはRQに保存され、サービスは実行しません</p> TPFUNC <p>サービスごとにRQデータを管理するときに使用されます。TPFUNCフラグが設定されていない場合、最初のtpenq()により保存されたデータが除去されます。データの保存前に除去が必要な場合、tpenq()の呼び出し時にTPFUNCと一緒に設定します</p> 0(zero) <p>サービス処理結果をRQに保存せず、関数の呼び出し元が接続されているTmaxシステムのクライアント・バッファに保存します。サービスはRQを使用して要求しますが、結果はtpcall()のように関数の呼び出し元が接続したクライアントのバッファから取得時に使用します。0(zero)フラグが設定された場合、後で処理結果を受信するためには、tpdeq()の呼び出し時にflagsに0(zero)を設定します</p>

- 戻り値

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

tpenq()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]	引数が有効でない場合です。たとえば、データがtpalloc()で割り当てられていないバッファを指す場合、あるいはflagsが有効でない場合に発生します
[TPENOENT]	存在しないqnameが使用された場合です

エラーコード	説明
[TPEQFULL]	持続的なサービス結果のため指定されたキューのサイズを超えた場合に発生します
[TPGOTSIG]	TPSIGRSTRTが設定されていない状態でシグナルが受信された場合です
[TPEPROTO]	tpenq()が不適切な状況で呼び出された場合です
[TPESYSTEM]	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
[TPEOS]	運用システムにエラーが発生した場合です

- 例

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

void main(int argc, char *argv[])
{
    int ret;
    char *buf;
    long len;

    ret=tpstart((TPSTART_T *)NULL);
    if (ret==-1) { error processing }

    buf = (char *)tpalloc("STRING", NULL, 0);
    if (buf==NULL) { error processing }
    data process....

    ret=tpenq("RQ", "SERVICE", (char *)buf, strlen(buf), TPRQS);
    if (ret==-1) { error processing }
    data process....

    ret=tpdeq("RQ", "SERVICE", (char **)&buf, (long *)&len, TPRQS);
    if (ret==-1) { error processing }
    data process....
    tpfree(buf);
    tpend();
}
```

- 関連関数

tpdeq(), tpqstat()

3.2.2. tpdeq

サーバーとクライアントでRQからデータをロードする関数です。**tpenq()**を使用してサービスの要求結果を受信するか、サービス名をNULLにして保存したデータを読み取ります。tpenq()の呼び出し時、flagsにTPNOREPLYを設定した場合、サービスは結果を受信できません。そのため、トランザクション・モードでtpdeq()を実行中にエラーが発生してもトランザクションは影響を及ぼしません。

● プロトタイプ

```
# include <tmaxapi.h>
int tpdeq (char *qname, char *svc, char **data, long *len, long flags)
```

● パラメータ

パラメータ	説明
qname	データを保存するRQの名前です。configファイルに登録された名前である必要があります
svc	tpenq()を呼び出す場合、svcに渡した名前である必要があります。サービス名でtpenq()を呼び出した場合、サービスが自動要求され、結果がRQに保存されます。サービス結果を受信するには、tpdeq()も同一サービス名を入力する必要があります。 サービス名をNULLでtpenq()を呼び出した場合、tpdeq()も同一サービス名を入力します。サービス名がNULLの場合、サービス名に関係なく、キューに蓄積しているすべてのデータを1つずつロードできます。tpenq()の場合、エラーやシステム障害によってフェイル・キューに保存されたデータをtpdeq()するには、svcに_rq_sub_queue_name[TMAX_FAIL_QUEUE]を与え、deqする必要があります
*data	tpalloc()によって割り当てられたバッファーに対するポインターです。関数が正常に返された場合、*dataは受信されたデータが保存されます
len	tpdeq()が正常に受信したデータ長です。tpdeq()は、必要であれば応答内容が指定されたバッファーに受信されるようバッファーのサイズを増加させます。 lenは*dataのデータ長です。*dataは受信データが大きくて変更されることもあり、それ以外の理由によって変更されることもあります。lenが呼び出し前のバッファーの総サイズより大きい場合、lenが該当バッファーの新規サイズになります。lenが0と返された場合、いかなるデータも受信されず、*dataとlenが指示するバッファーすべて何も変化はありません。 *dataやlenがNULLになるのはエラーです
flags	flagsで使用可能な値は以下のとおりです – TPRQS

パラメータ	説明
	<p>RQからデータを取得時に使用されます。replyキューからサービス結果を取得するために設定されます</p> <p>– TPFUNC</p> <p>サービスごとにRQデータを管理するときに使用されます。TPFUNCフラグが設定されていない場合、最初のtpenq()により保存されたデータが除去されます。データの保存前に除去が必要な場合、tpenq()の呼び出し時にTPFUNCと一緒に設定します</p> <p>– TPBLOCK</p> <p>tpdeq()を呼び出したとき、ブロック・タイムアウト時間の間、メッセージが来るまで待機します</p> <p>– TPNOTIME</p> <p>TPBLOCKと一緒に使用された場合、ブロック・タイムアウト時間に関係なく、応答が来るまで待機します</p> <p>– 0(zero)</p> <p>自身が接続したクライアントのバッファからデータを取得時に使用します</p>

- 戻り値

戻り値	説明
-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

tpdeq()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
[TPEINVAL]	引数が有効でない場合です。たとえば、データがtpalloc()で割り当てられていないバッファを指す場合、あるいはflagsが有効でない場合に発生します
[TPGOTSIG]	TPSIGRSTRTが設定されていない状態でシグナルが受信された場合です
[TPEMATCH]	サービス名が正しくない場合や除去するデータがない場合、あるいは該当条件を満たす除去データが見つからなかった場合に発生します
[TPENOENT]	存在しないqnameが使用された場合です
[TPEPROTO]	tpdeq()が不適切な状況で呼び出された場合です
[TPESYSTEM]	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

エラーコード	説明
[TPEOS]	運用システムにエラーが発生した場合です

- 例

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

void main(int argc, char *argv[])
{
    int ret;
    char *buf;
    long len;
    ret=tpstart((TPSTART_T *)NULL);
    if (ret==-1) { error processing }

    buf = (char *)tpalloc("STRING", NULL, 0);
    if (buf==NULL) { error processing }
    data process....

    ret=tpenq("RQ", "SERVICE", (char *)buf, strlen(buf), TPRQS);
    if (ret==-1) { error processing }
    data process....

    ret=tpdeq("RQ", "SERVICE", (char **)&buf, (long *)&len, TPRQS);
    if (ret==-1) { error processing }
    data process....

    tpfree(buf);
    tpend();
}
```

- 関連関数

tpenq(), tpqstat()

3.2.3. tpissue

サーバーとクライアントで使用され、該当RQのフェイル・キューにたまっている要求データを、再度リクエスト・キューに入れる関数です。**tpenq()**などでRQを使用したサービスの実行中に、ネットワークの不安定やその他サーバー側のエラーによってサービス実行に失敗し、フェイル・キューにたまったクライアントのサービス要求データを再度リクエスト・キューに入れる関数です。保存されたデータを該当サービスに渡し、結果をリプレイ・キューに保存します。

- プロトタイプ

```
#include <tmaxapi.h>
int tpreissue(char *qname, char *filter, long flags)
```

- パラメータ

パラメータ	説明
qname	Tmaxシステムに登録されたRQの名前です
filter	現在サポートしておらず、NULLと設定します
flags	現行バージョンではサポートしていませんが、TPNOFLAGSに設定します

- 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

tpreissue()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]	引数が有効でない場合です。たとえば、qnameがNULLである場合や、サポートしていないフィルターやフラグに上述した値以外の値を設定した場合に発生します
[TPENOENT]	qnameに該当するRQが存在しない場合です
[TPESYSTEM]	Tmaxシステムにエラーが発生した場合です。ネットワーク状態が不安定な場合に発生します

- 関連関数

tpenq(), tpdeq()

3.3. RQ使用補助API

3.3.1. tpextsvcname

サーバーとクライアントで`tpdeq()`を使用してRQからデータを読み込んだ場合、該当データのサービス名を確認する際に使用する関数です。`tpextsvcname()`は、_Failキューに保存されているデータを`tpdeq()`で読み込んだ場合に使用します。

- プロトタイプ

```
# include <tmaxapi.h>
int tpextsvcname (char *data, char *svc)
```

- パラメータ

パラメータ	説明
data	tpdeq()を使用してRQから読み込んだデータが保存されているポインターです。tpalloc()で割り当てられます
svc	該当データのサービス名を取得するためのポインターです

- 戻り値

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

tpextsvcname()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]	引数が有効でない場合です。たとえば、dataにtpalloc()で割り当てられていないバッファが渡された場合に発生します
[TPESYSTEM]	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
[TPEOS]	運用システムにエラーが発生した場合です

- 例

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

```

void main(int argc, char *argv[])
{
    int ret;
    char *buf, *svc_name;
    long len;

    ret=tpstart((TPSTART_T *)NULL);
    if (ret==-1) { error processing }

    buf = (char *)tpalloc("STRING", NULL, 0);
    if (buf==NULL) { error processing }
    data process....

    ret=tpenq("RQ", "SERVICE", (char *)buf, strlen(buf), TPRQS);
    if (ret==-1) { error processing }
    data process....

    ret=tpdeq("RQ", "SERVICE", (char **)&buf, (long *)&len, TPRQS);
    if (ret==-1) { error processing }

    ret=tpextsvcname(buf, svc_name);
    if (ret==-1) { error processing }
    printf("svc name : %s      ",svc_name);
    data process....

    tpfree(buf);
    tpend();
}

```

- 関連関数

tpenq(), tpdeq()

3.3.2. tpextsvcinfo

サーバーとクライアントで**tpdeq()**を使用してRQからデータを読み込んだ場合、該当データについての詳細情報を提供する関数です。

- プロトタイプ

```

# include <tmaxapi.h>
int tpextsvcinfo (char *data, char *svc, , int *type, int *errcode )

```

- パラメータ

パラメータ	説明
data	tpalloc()で割り当てられ、tpdeq()を使用してRQから読み込んだデータが保存されているポインターです
svc	該当データのサービス名を取得するためのポインターです
type	<p>該当データの処理結果を表します。</p> <p>以下は、typeに設定可能な値についての説明です</p> <ul style="list-style-type: none"> – TPREQ(0) <p>tpenq()の実行時、2番目のパラメータとしてNULLを指定した場合、tpenqが正常になった際、typeにTPREQが設定されます</p> – TPFAIL(1) <p>tpenq()の実行時、2番目のパラメータとしてサービス名を指定した場合、サービスでtpreturnの最初のパラメータとしてTPFAILが呼び出された際、typeにTPFAILが設定されます</p> – TPSUCCESS(2) <p>tpenq()の実行時、2番目のパラメータとしてサービス名を指定した場合、サービスでtpreturnの最初のパラメータとしてTPSUCCESSが呼び出された際、typeにTPSUCCESSが設定されます</p> – TPERR(-1) <p>tpenq()が失敗してフェイル・キューに送信された場合、typeにTPERRが設定されます</p>
errcode	エラーが発生した場合、該当するエラーコード値が保存されます

- 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

tpextsvcname()が正常処理されていない場合、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]	引数が有効でない場合です。たとえば、dataまたはsvcがNULLです
[TPEITYPE]	引数が有効でない場合です。たとえば、dataがRQから取得したdataでない場合に発生します

エラーコード	説明
[TPESYSTEM]	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
[TPEOS]	運用システムにエラーが発生した場合です。

- 関連関数

tpenq(), tpdeq(), tpextsvcname()

3.3.3. tpqstat

サーバーとクライアントで使用され、RQに保存されているデータの統計を要求する関数です。RQは、内部的に_fail queue、_request queue、_reply queueの3部で構成されています。フラグ値を使用して、各キューに保存されたデータ統計を求めることができます。

- プロトタイプ

```
# include <tmaxapi.h>
int tpqstat (char *qname, long flags)
```

- パラメータ

パラメータ	説明
qname	Tmax環境ファイルに登録されたRQ名を表します
flags	対象データのタイプです。 flagsで使用可能な値は以下のとおりです <ul style="list-style-type: none"> – 0(TMAX_ANY_QUEUE) : _fail queue、_request queue、_reply queueのデータ統計を求める際に使用します – 1(TMAX_FAIL_QUEUE) : _fail queueのデータ統計を求める際に使用します – 2(TMAX_REQ_QUEUE) : _request queueのデータ統計を求める際に使用します – 3(TMAX_RPLY_QUEUE) : _reply queueのデータ統計を求める際に使用します

- 戻り値

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

tpqstat()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]	引数が有効でない場合です。たとえば、qnameがNULLの場合や、qnameに該当するキューがない場合、flagsが有効でない場合に発生します
[TPEPROTO]	tpqstat()が不適切な状況で呼び出された場合です
[TPESYSTEM]	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
[TPEOS]	運用システムにエラーが発生した場合です

● 例

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

void main(int argc, char *argv[])
{
    int ret, i;
    char *buf;

    if (argc!=2) { error processing }
    ret=tpstart((TPSTART_T *)NULL);
    if (ret== -1) {error processing }

    buf = (char *)tpalloc("STRING", NULL, 0);
    if (buf==NULL) { error processing }
    strcpy(buf, argv[1]);

    data process...

    ret=tpenq("RQ", NULL, (char *)buf, strlen(buf), TPRS);
    if (ret== -1) {error processing }
    printf("qstat :");
    for (i=0;i<4;i++) {
        ret=tpqstat("rq", i);
        if (ret== -1) {error processing }
        printf("  %d",ret);          /* qstat :  1  0  0  1 */
    }
}
```

```

printf("\n");
data process...

ret=tpenq("RQ", "SERVICE", (char *)buf, strlen(buf), TPRS);
if (ret==-1) {error processing }
printf("qstat :");

for (i=0;i<4;i++) {
    ret=tpqstat("rq", i);
    if (ret==-1) {error processing }
    printf("  %d",ret);          /* qstat :  2  0  0  2 */

}
printf("\n");
tpfree((char *)buf);
tpend();
}

```

- 関連関数

tpenq(), tpdeq()

3.3.4. tpqsvcstat

サーバーとクライアントで使用され、RQに保存されているデータのうち、指定したサービスの統計を要求する関数です。RQに保存されているデータの統計を要求します。RQは、内部的に_fail queue、_request queue、_reply queueの3部で構成されています。フラグ値を使用して、各キューに保存されたデータ統計を求めることができます。

- プロトタイプ

```

# include <tmaxapi.h>
int tpqsvcstat (char *qname, char * svc, long flags )

```

- パラメータ

パラメータ	説明
qname	Tmax環境ファイルに登録されたRQ名を設定します
svc	統計情報を取得するサービス名です。NULLの場合、 tpqstat() と同一の意味をもちます
flags	
flags	データ統計のタイプを設定します。

パラメータ	説明
	<p>以下は、flagsに設定可能な値についての説明です</p> <ul style="list-style-type: none"> – 0(TMAX_ANY_QUEUE) : _fail queue, _request queue, _reply queueのデータ統計を求める際に使用します – 1(TMAX_FAIL_QUEUE) : _fail queueのデータ統計を求める際に使用します – 2(TMAX_REQ_QUEUE) : _request queueのデータ統計を求める際に使用します – 3(TMAX_RPLY_QUEUE) : _reply queueのデータ統計を求める際に使用します

- 戻り値

戻り値	説明
-1以外の値	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

tpqscvstat()が正常処理されていない場合、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]	引数が有効でない場合です。たとえば、qnameがNULLの場合や、qnameに該当するキューがない場合、タイプが有効でない場合に発生します
[TPEPROTO]	tpqscvstat()が不適切な状況で呼び出された場合です
[TPESYSTEM]	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
[TPEOS]	運用システムにエラーが発生した場合です

- 関連関数

tpenq(), tpdeq(), tpqstat()

3.3.5. tpenq_ctl

サーバーとクライアントでトランザクションをサポートし、RQデータを保存する関数です。Tmaxシステムは、システムの障害やエラーによってサービス不可能な状態でも、RQに保存されたデータは整合性を保証でき

ます。RQにデータを保存しておき、様々な状況でシステムがダウンして、復旧以降に再度実行された場合、まだ処理できていないデータを引き続き処理できるようにします。

tpcall()や**tpacall()**でサービスを要求した場合、該当サービスが実行するデータが累積していると、サービスを要求したデータも待機(waiting)します。この際、システムの障害やエラーによってシステムがダウンした場合、待機中のデータは失われます。こういった問題点を補完し、データの整合性を保証できるように、**tpenq_ctl()**はサービスを要求する場合、データをRQに保存します。

tpenq_ctl()関数は、トランザクション・モードで実行した場合、データをRQに保存するまでトランザクションで処理されます。2回以上の**tpenq_ctl()**を1つのトランザクションに結合した場合、すべてのデータがRQに保存されるまで**tpdeq_ctl()**を使用してロードすることはできず、保存中にエラーが発生するとロールバックされます。

- プロトタイプ

```
#include <usrinc/tmaxapi.h>
int tpenq_ctl(char *qname, char *svc, TMQCTL *ctl, char *data, long len,
              long flags);
```

- パラメータ

パラメータ	説明
qname	データを保存するRQの名前です。環境ファイルに登録された名前である必要があります
svc	データをRQに保存し、svc名がNULLでない場合は即時サービスを要求します。 svc名がNULLの場合、データはRQに保存され、サービスは実行されません。この場合、後に tpdeq_ctl() を使用してサービスを要求する必要があります。svcで命令されたサービスがない場合、あるいはサービスを実行して処理結果を受信していない状態でシステム障害が発生した場合、このデータは内部的にフェイル・キューに保存されます。このデータも、 tpdeq_ctl() でサービスを再要求するか、エラー処理する必要があります
ctl	TMQCTLのdet_timeを使用して一定時間を設定した場合、その時間以降にsvc callします。 deq_timeの設定時は、現在の時間にdelay_timeを設定する必要があります。 delay_timeに3を設定したい場合、「cur_time + 3」のように設定します。TMQCTLの機能のうち、現在はdeq_timeのみサポートします。deq_timeはトランザクションと一緒に使用できません。 flags項目にTPRQS_NON_PERSISTENTを設定する場合、メッセージをファイルに書き込まずにメモリーにのみ書き込みます

パラメータ	説明
data	NULLの場合を除いて、必ずtpalloc()で割り当てられたバッファーに対するポインターである必要があります。dataのタイプとサブタイプは、svcがサポートするタイプである必要があります
len	<p>送信するデータ長です</p> <ul style="list-style-type: none"> dataが指すバッファーが特別な長さ明示が不要なタイプ (STRING、STRUCT、X_COMMON、X_C_TYPE) の場合、lenは無視され、0が使用されます dataが指すバッファーが、長さを指定する必要があるタイプ (X_OCTET、CARRAY、MULTI STRUCTURE) の場合、lenは0になれません dataがNULLの場合、lenは無視され、データなしでサービス要求が送信されます
flags	<p>データ処理タイプを設定します。</p> <p>以下は、flagsに設定可能な値についての説明です</p> <ul style="list-style-type: none"> TPRQS <p>svcがNULLでない場合、svcに指定されたサービスを要求し、処理結果をRQに保存します。サービスの処理結果は、tpdeq_ctl()関数を利用して受信します。svcがNULLの場合、データはRQに保存され、サービスは実行しません</p> TPNOREPLY <p>svcがNULLでない場合、svcに指定されたサービスを要求しますが、処理結果はRQに保存しません。svcがNULLの場合、データはRQに保存され、サービスは実行しません</p> TPFUNC <p>サービスごとにRQデータを管理するときに使用されます。TPFUNCフラグが設定されていない場合、最初のtpenq_ctl()により保存されたデータが除去されます。データの保存前に除去が必要な場合、tpenq_ctl()の呼び出し時にTPFUNCと一緒に設定します</p> 0(zero) <p>サービス処理結果をRQに保存せず、関数の呼び出し元が接続されているTmaxシステムのクライアント・バッファーに保存します。サービスはRQを使用して要求しますが、結果はtpcall()のように関数の呼び出し元が接続したクライアントのバッファーから取得時に使用します。処理結果を受信するには、tpdeq_ctl()の呼び出し時にflagsに0(zero)を設定します</p>

- 戻り値

戻り値	説明
-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

tpenq_ctl()が正常処理されていない場合、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
[TPEINVAL]	引数が有効でない場合です。たとえば、dataがtpalloc()で割り当てられていないバッファを指す場合、あるいはflagsが有効でない場合に発生します
[TPENOENT]	存在しないqnameが使用された場合です
[TPEQFULL]	持続的なサービス結果のため指定されたキューのサイズを超えた場合に発生します
[TPGOTSIG]	TPSIGRSTRTが設定されていない状態でシグナルが受信された場合です
[TPEPROTO]	tpenq_ctl()が不適切な状況で呼び出された場合です
[TPESYSTEM]	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
[TPEOS]	運用システムにエラーが発生した場合です

- 例

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

main(int argc, char *argv[])
{
    char    *sndbuf, *rcvbuf;
    long    rcvlen, sndlen, revent;
    int ret, i, cd;
    TMQCTL  *ctl;
    long t;

    ctl = (TMQCTL*)malloc(sizeof(TMQCTL));
    memset(ctl, 0, sizeof(TMQCTL));
    time(&t);
    ctl->deq_time = (int)t + 3;

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

```

}
if ( (ret = tmaxreadenv( "tmax.env","TMAX" )) == -1 ){
    printf( "tmax read env failed\n" );
    exit(1);
}
if (tpstart((TPSTART_T *)NULL) == -1){
    printf("tpstart failed\n");
    exit(1);
}
if ((sndbuf = (char *)tpalloc("STRING", NULL, 0)) == NULL) {
    printf("sndbuf alloc failed !\n");
    tpend();
    exit(1);
}
if ((rcvbuf = (char *)tpalloc("STRING", NULL, 0)) == NULL) {
    printf("rcvbuf alloc failed !\n");
    tpfree((char *)sndbuf);
    tpend();
    exit(1);
}
ret = tx_begin();
if(ret < 0)
{
    fprintf(stderr, "tx_begin() fail\n");
    tpfree((char *)sndbuf);
    tpfree((char *)rcvbuf);
    exit(1);
}

strcpy(sndbuf, argv[1]);
cd = tpenq_ctl("txrq1", "TOUPPER", ctl, (char *)sndbuf, 0, TPRS );
if (cd < 0) {
    printf("tpenq failed [%s]\n", tpstrerror(tperrno));
    tpfree((char *)sndbuf);
    tpfree((char *)rcvbuf);
    tpend();
    exit(1);
}
cd = tpenq_ctl("txrq1", "TOUPPER", ctl, (char *)sndbuf, 0, TPRS );
if (cd < 0) {
    printf("tpenq failed [%s]\n", tpstrerror(tperrno));
    tpfree((char *)sndbuf);
    tpfree((char *)rcvbuf);
    tpend();
    exit(1);
}
data process....

```

```

ret = tx_commit();
if(ret < 0)
{
    fprintf(stderr, "tx_commit() fail\n");
    tpfree((char *)sndbuf);
    tpfree((char *)rcvbuf);
    tx_rollback();
    exit(1);
}

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

tpend();
}

```

- 関連関数

tpdeq_ctl(), tpqstat()

3.3.6. tpdeq_ctl

サーバーとクライアントでトランザクションをサポートし、RQからデータをロードする関数です。**tpenq_ctl()**を使用してサービスを要求した結果、あるいはサービス名をNULLにして保存したデータを読み取る関数です。tpenq_ctl()を呼び出したとき、flagsにTPNOREPLYを設定したサービスは結果を受信できません。tpdnq_ctl()関数は、トランザクション・モードで実行した場合、データをRQからロードする間トランザクションで処理されます。2回以上のtpdeq_ctl()を1つのトランザクションに結合した場合、すべてのデータがRQからロード中にエラーが発生するとロールバックされます。

- プロトタイプ

```

#include <usrinc/tmaxapi.h>
int tpdeq_ctl(char *qname, char *svc, TMQCTL *ctl, char **data,
              long *len, long flags);

```

- パラメータ

パラメータ	説明
qname	データを保存するRQの名前です。環境ファイルに登録された名前である必要があります
svc	tpenq_ctl()を呼び出す場合、svcに渡した名前である必要があります。 サービス名でtpenq_ctl()を呼び出した場合、サービスが自動要求され、結果がRQに保存されます。サービス結果を受信するには、tpdeq_ctl()も同一サービス名を入力

パラメータ	説明
	<p>します。サービス名をNULLでtpenq_ctl()を呼び出した場合、tpdeq_ctl()も同一サービス名を入力します。サービス名がNULLの場合、サービス名に関係なく、キューに蓄積しているすべてのデータを1つずつdeqできます。</p> <p>tpenq_ctl()関数がエラーやシステム障害によってフェイル・キューに保存されたデータをtpdeq_ctl()するには、svclに_rq_sub_queue_name [TMAX_FAIL_QUEUE]を与え、deqする必要があります</p>
ctl	<p>– * flags</p> <ul style="list-style-type: none"> • TPRQS_AUTOACK : deqするとき、別途にackを与えなくてもRQSから削除されるようにします • TPRQS_NOAUTOACK : deqするとき、RQSから削除されないようにします • TPRQS_ACK_SUCCESS : ctlに該当するq要素をackを与えて削除します • TPRQS_ACK_FAIL : deqするとき、RQSから削除されないようにします <p>– * exp_time</p> <p>deqした後、与えられた時間が過ぎても応答がない場合は、q要素が再びdeqできる状態に戻ります(単位: 秒)</p>
data	<p>以前にtpalloc()によって割り当てられたバッファに対するポインターである必要があります。tpdeq_ctl()が正常に実行された場合、受信されたデータが保存されます</p>
len	<p>tpdeq_ctl()が正常に受信したデータ長です。必要な場合、応答内容が指定されたバッファに受信されるようにバッファのサイズを増加させます。</p> <p>*dataは、受信データが大きくて変更されることもあり、それ以外の理由によって変更されることもあります。lenが呼び出し前のバッファの総サイズより大きい場合、lenが該当バッファの新規サイズになります。lenが0と返された場合、いかなるデータも受信されず、*dataとlenが指示するバッファすべて何も変化はありません。*dataやlenがNULLになるのはエラーです</p>
flags	<p>データ処理タイプを設定します。</p> <p>以下は、flagsで使用可能な値についての説明です</p> <p>– TPRQS</p> <p>RQからデータを取得時に使用されます。replyキューからサービス結果を取得するために設定されます</p> <p>– TPFUNC</p>

パラメータ	説明
	<p>サービスごとにRQデータを管理するときに使用されます。TPFUNCフラグが設定されていない場合、最初のtpenq()により保存されたデータが除去されます。データの保存前に除去が必要な場合、tpenq()の呼び出し時にTPFUNCと一緒に設定します</p> <p>– TPBLOCK</p> <p>tpdeq_ctl()を呼び出したとき、ブロック・タイムアウト時間の間、メッセージが来るまで待機します</p> <p>– TPNOTIME</p> <p>TPBLOCKと一緒に使用された場合、ブロック・タイムアウト時間に関係なく、応答が来るまで待機します</p> <p>– 0(zero)</p> <p>自身が接続したクライアントのバッファからデータを取得時に使用します</p>

- 戻り値

戻り値	説明
-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

tpdeq_ctl()が正常処理されていない場合、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
[TPEINVAL]	引数が有効でない場合です。たとえば、データがtpalloc()で割り当てられていないバッファを指す場合、あるいはflagsが有効でない場合に発生します
[TPGOTSIG]	TPSIGRSTRTが設定されていない状態でシグナルが受信された場合です
[TPEMATCH]	サービス名が正しくない場合です。あるいは、該当条件を満たすデータが見つからない場合です
[TPENOENT]	存在しないqnameが使用された場合です
[TPEPROTO]	tpdeq_ctl()が不適切な状況で呼び出された場合です
[TPESYSTEM]	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
[TPEOS]	運用システムにエラーが発生した場合です

- 例


```

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

main(int argc, char *argv[])
{
    char      *sndbuf, *rcvbuf;
    long      rcvlen, sndlen, revent;
    int ret, i, cd;
    TMQCTL    *ctl;

    ctl = (TMQCTL*)malloc(sizeof(TMQCTL));
    memset(ctl, 0, sizeof(TMQCTL));
    if (argc != 2) {
        printf("Usage: toupper_rq string\n");
        exit(1);
    }

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

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

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

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

    ret = tx_begin();
    if(ret < 0)
    {

```

```

        fprintf(stderr, "tx_begin() fail\n");
        tpfree((char *)sndbuf);
        tpfree((char *)rcvbuf);
        exit(1);
    }

    strcpy(sndbuf, argv[1]);
    cd = tpdeq_ctl("txrq1", "TOUPPER", ctl, &rcvbuf, &rcvlen, TPRS );
    if (cd < 0)
    {
        printf("tpdeq failed [%s]\n", tpstrerror(tperrno) );
        tpfree((char *)sndbuf);
        tpfree((char *)rcvbuf);
        tpend();
        exit(1);
    }

    cd = tpdeq_ctl("txrq1", "TOUPPER", ctl, &rcvbuf, &rcvlen, TPRS );
    if (cd < 0)
    {
        printf("tpdeq failed [%s]\n", tpstrerror(tperrno) );
        tpfree((char *)sndbuf);
        tpfree((char *)rcvbuf);
        tpend();
        exit(1);
    }

    data process....
    ret = tx_commit();
    if(ret < 0)
    {
        fprintf(stderr, "tx_commit() fail\n");
        tpfree((char *)sndbuf);
        tpfree((char *)rcvbuf);
        tx_rollback();
        exit(1);
    }
    tpfree((char *)sndbuf);
    tpfree((char *)rcvbuf);
    tpend();
}

```

- 関連関数

tpenq_ctl(), tpqstat()

3.4. その他

3.4.1. compute_rq_deqtime

AnyLinkゲートウェイがdeq_timeを決める場合は、RQがダウンしたときに、遅延なく処理するためには、deq_timeをリセットしなければなりません。しかし、RQのダウンや起動をAnyLinkゲートウェイは知らないため、リセットできないという問題があります。AnyLinkゲートウェイがダウンする場合も、deq_timeが初期化されるので、順序を保証できない状況が発生することがあります。

そのため、AnyLinkゲートウェイではなく、RQがdeq_timeを適用できるようにします。RQはdeq_timeを自ら決められないため、deq_timeを適用するには、コールバック関数を呼び出してAnyLinkライブラリーを呼び出します。

RQSが起動する環境変数に以下のように設定した後使用できます。

```
RQ_DEQTIME_CALLBACK=libanme.so
```

この設定があれば、該当するライブラリーをdlopenし、コールバック関数を呼び出してdeq_timeを適用します。

● プロトタイプ

```
int compute_rq_deqtime(TMQCTL *ctl, char *dp, int datasize);
```

● パラメータ

パラメータ	説明
ctl	– * deq_time : 関数を作成するとき、構造体内のdeq_timeに記録するように作成する必要があります(単位: 秒)
dp	チェックするメッセージのポインターです
datasize	メッセージの長さです

● 戻り値

戻り値	説明
1	deq_timeの計算に成功した場合です
0	コールバック関数で渡されたデータにAnyLinkヘッダーのマジック・ナンバーが存在しないか、一致しない場合です(AnyLinkエンジン以外のサービスでRQを呼び出した場合を仮定)
-1	処理中にエラーが発生した場合です

- 関連関数

`tpenq_ctl()`, `tpenq()`

第4章 サンプル

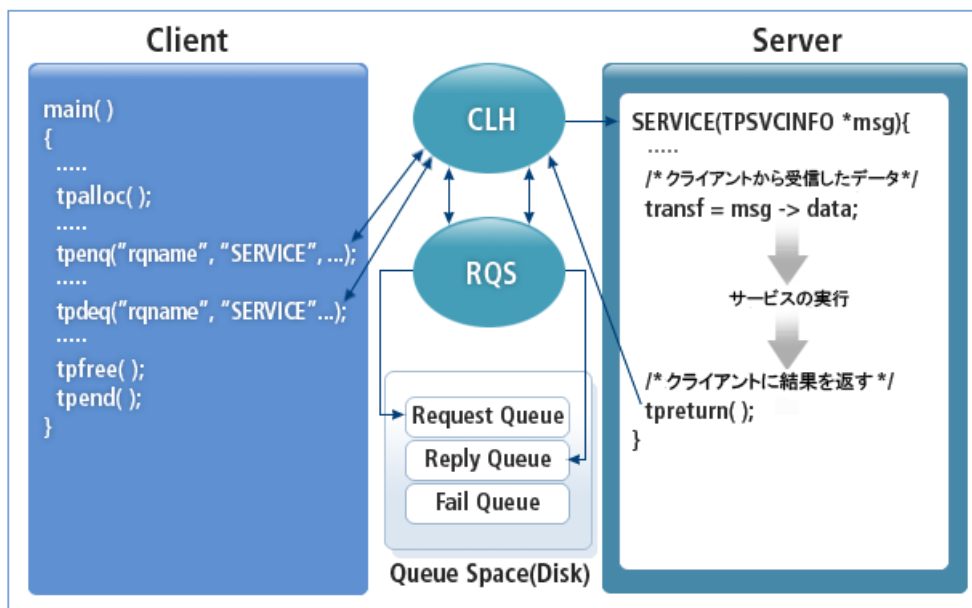
本章では、実際にTmax RQシステムのサンプルを通して使用方法を説明します。

4.1. 基本プログラム

RQプログラムの動作は基本的に非同期型通信と似ています。RQシステムを使用する最も基本的な形式のプログラムで、クライアントは1つのRQを使用してtpenq()でサービスを呼び出し、tpdeq()で結果値を取得します。

以下は、クライアントがtpenq()とtpdeq()を使用してサービス結果を取得するRQプログラムのプロセスです。

【図 4.1】 RQプログラムのプロセス



クライアントはtpenq()でサービスを呼び出し、tpdeq()で結果値を取得します。tpenq()のフラグにTPNOFLAGSを与えた場合のプロセスはこれとほぼ同じです。tpenq()にTPNOFLAGS以外のフラグを与えた場合は、サービスの結果値がクライアントではなくサーバー側のRQデータ・ファイルに保存されます。つまり、クライアントはサービス実行の完了をtpsleap()などで確認できません。開発者がサービス完了時点を知りたい場合は、TPNOFLAGSを設定して使用するか、または他のAPIを使用して実装できます。TPNOFLAGS以外のフラグを使用する場合、サーバーから渡されるurcode値は無視されます。

サービス名を明示し、フラグはTPRQSを使用しました。クライアントはtpqstat()を使用してreplyキューにたまっているデータを読み込みます。サービス名を明示してRQを使用する場合、サービスのスケジューリングは各サービス別に実行され、tpenq()で保存された順序とは異なる可能性があります。

参考

エラーが発生するプロセスについては「[3.2.2. tpdeq](#)」を参照してください。関数についての詳細は『Tmax リファレンスガイド』を参照してください。

4.1.1. 環境設定

以下はTmax環境設定の例です。

```
*DOMAIN
tmax      SHMKEY=88000,
          TPORTNO = 8888

*NODE
tmax1     TMAXDIR="/user1/tmax",
          APPDIR="/user1/tmax/appbin"

*SVRGROUP
svg1      NODENAME = tmax1,
          CPC = 4,
          SVGTYPE=RQMGR

*RQ
rqtest    SVGNAME = svg1

*SERVER
svr       SVGNAME = svg1

*SERVICE
SERVICE  SVRNAME = svr
```

参考

各セクションの設定項目についての説明は、『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。

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

```

int main(int argc, char *argv[])
{
    char *buf;
    long rlen;
    int rqcount, i;

    if (argc != 2){
        error procesing
    }

    if (tpstart((TPSTART_T *)NULL) == -1){
        error processing
    }

    buf = tpalloc("CARRAY", NULL, 0);
    if (buf == NULL){
        error processing
    }

    strcpy(buf, argv[1]);

    if (tpenq("rqtest", "SERVICE", buf, strlen(buf), TPRQS) == -1){
        error processing
    }

    ...

    rqcount = tpqstat("rqtest", TMAX_RPLY_QUEUE);
    for(i = 0; i < rqcount; i++){
        ...
        if(tpdeq("rqtest", "SERVICE", &buf, &rlen, TPRQS) == -1){
            error processing
        }
        ...
    }

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

```

4.1.3. 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>

SERVICE(TPSVCINFO *msg)
{
    int i;
    ...
    tpreturn(TPSUCCESS, 0, (char *)msg->data, 0, TPNOFLAGS);
}
```

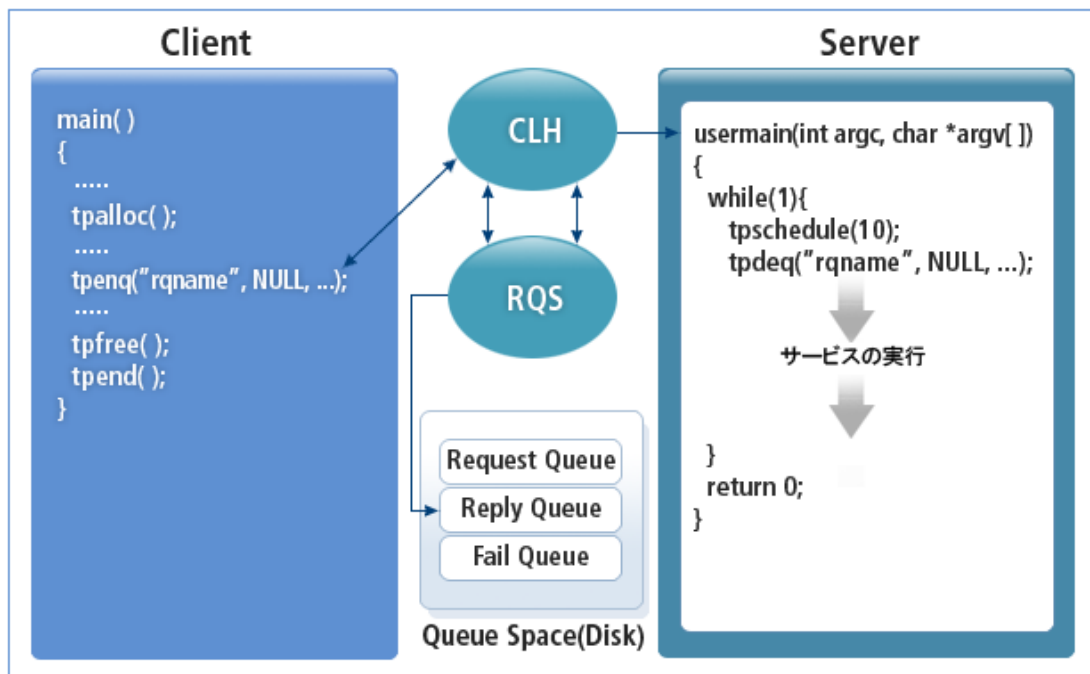
4.2. UCS使用プログラム

tpenq() – tpdeq()という簡単な構造のRQプログラムもTmaxの多様なプロセス管理タイプを利用して開発するとその使用範囲がとてまくなりまします。本節ではTmaxの独自の方法であるUCS方式を使用した様々なプログラミング技法を説明します。

クライアントが要求する一連の作業が順に実行される必要があり、各サーバー・プログラムの実行時間が長い場合、クライアントは全体サービス期間の間ブロック状態で待たなければなりません。しかしRQを使用するとクライアントは希望するサービス内容をRQに保存して次の作業を実行することができ、サーバーはサービス要求内容をRQから順に読み込んで実行することができます。

以下はUCSを使用するプログラムのプロセスです。

[図 4.2] UCS使用プログラムのプロセス



クライアントで希望するサービス名とデータを構造体に保存し、複数回tpenq()を実行します。このとき、サービス名はNULLを与えます。該当データはReplyキューにたまり、UCSサーバーではキューからデータを読み込んで該当サービスにデータを送信してサービスを呼び出します。

4.2.1. 環境設定

以下はTmax環境設定の例です。

```
*DOMAIN
tmax      SHMKEY=88000,
          TPORTNO = 8888

*NODE
tmax1     TMAXDIR="/user1/tmax",
          APPDIR="/user1/tmax/appbin"

*SVRGROUP
svg1      NODENAME = tmax1,
          CPC = 4,
          SVGTYPE=RQMGR

*RQ
rqenq     SVGNAME = svg1
rqdeq     SVGNAME = svg1

*SERVER
svr1      SVGNAME = svg1
svr2      SVGNAME = svg1,
          SVRTYPE = UCS

*SERVICE
SERVICE1 SVRNAME = svr1
SERVICE2 SVRNAME = svr1
SERVICE3 SVRNAME = svr1
```

参考

各セクションの設定項目についての説明は、『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。

4.2.2. 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 <sys/time.h>
#include <usrinc/atmi.h>
#include <usrinc/tmaxapi.h>
#include "../sdl/demo.s"

int main(int argc, char *argv[])
{
    struct *buf1, *buf2, *buf3
    long rlen;
    int rqcount, i;
    struct timeval tval;

    if (argc != 4){
        error processing
    }

    if (tpstart((TPSTART_T *)NULL) == -1){
        error processing
    }

    buf1 = (struct rqsvc *)tpalloc("STRUCT", "rqsvc", 0);
    buf2 = (struct rqsvc *)tpalloc("STRUCT", "rqsvc", 0);
    buf3 = (struct rqsvc *)tpalloc("STRUCT", "rqsvc", 0);
    if (buf1 == NULL || buf2==NULL || buf3==NULL){
        error processing
    }

    strcpy(buf1->svc, "SERVICE1");
    strcpy(buf1->data, argv[1]);
    strcpy(buf2->svc, "SERVICE2");
    strcpy(buf2->data, argv[2]);
    strcpy(buf3->svc, "SERVICE3");
    strcpy(buf3->data, argv[3]);

    /*サービス名をNULLで与え、Replyキューにデータを送信します。*/
    if (tpenq("rqenq", NULL, buf1, 0, TPRQS) == -1){
        error processing
    }
    if (tpenq("rqenq", NULL, buf2, 0, TPRQS) == -1){
        error processing
    }
    if (tpenq("rqenq", NULL, buf3, 0, TPRQS) == -1){
        error processing
    }

    tpfree((char *)buf1);

```

```

    tpfree((char *)buf2);
    tpfree((char *)buf3);
    tpend();
}

```

4.2.3. 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/ucs.h>
#include <usrinc/tmaxapi.h>
#include "../sdl/demo_sdl.h"

int usermain(int argc, char *argv[])
{
    struct rqsvc *buf;
    char *sndbuf;
    long rlen;
    int count, i, ret;

    buf = (struct rqsvc *)tpalloc("STRUCT", "rqsvc", 0);
    if (buf == NULL){
        error processing
    }

    sndbuf = tpalloc("STRING", NULL, 0);
    if (sndbuf == NULL){
        error processing
    }

    while(1){
        tpschedule(10);

        /*Replyキューのデータ数を取得します。*/
        count = tpqstat("rqenq", TMAX_RPLY_QUEUE);
        printf("count = %d\n", count);

        for (i = 0; i < count ; i++){
            /*Replyキューにあるデータのうちサービス名がNULLのデータを送信します。*/
            if (tpdeq("rqenq", NULL, (char **)&buf, &rlen, TPRQS) == -1){
                error processing
            }

            buf=(struct rqsvc *)buf;

```

```

strcpy(sndbuf, buf->data);

/*順次にサービスを実行します。*/
if (tpcall(buf->svc, sndbuf, 0, (char **)&buf, (long *)&rlen,
          TPNOFLAGS) == -1){
    error processing
}
...
}
}
return 1;
}

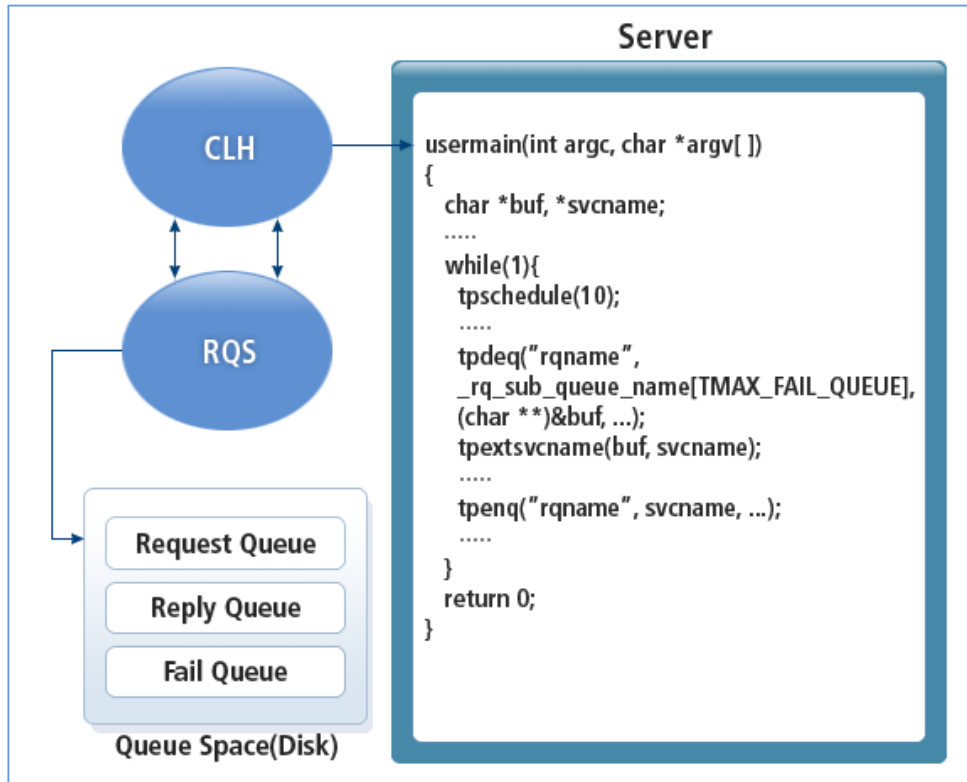
```

4.3. Failキュー復旧プログラム

様々な原因でFailキューに蓄積されたデータは状況によって適切な方式で処理しなければなりません。最も簡単な方法は管理ツールを使用してキューの内容を削除することですが、ユーザーが希望する方式で処理したい場合にはそのためのプログラムを作成する必要があります。

以下はUCS方式のプログラムで、Failキューを復旧するプログラムのプロセスです。

[図 4.3] Failキュー復旧プログラムのプロセス



10秒間隔でFailキューのデータを読み込んでtpextsvcname()で該当するサービスを探した後、特定のサービスのデータは破棄し、残りは再び該当サービスのRequestキューに送信します。tpreissue()の場合は、データを読み込んで該当するサービスに送信するプロセスを一度に実行しますが、上記のように状況によって別の行動を取る場合には使用できません。

4.3.1. 環境設定

以下はTmax環境設定の例です。

```
*DOMAIN
tmax      SHMKEY=88000,
          TPORTNO = 8888

*NODE
tmax1     TMAXDIR="/user1/tmax",
          APPDIR="/user1/tmax/appbin"

*SVRGROUP
svg1      NODENAME = tmax1,
          CPC = 4,
          SVGTYPE=RQMGR

*RQ
rqtest    SVGNAME = svg1, BOOT = WARM

*SERVER
svr1      SVGNAME = svg1
svr2      SVGNAME = svg1
svr3      SVGNAME = svg1,
          SVRTYPE = UCS

*SERVICE
SERVICE1 SVRNAME = svr1
SERVICE2 SVRNAME = svr2
STOP      SVRNAME = svr2
```

参考

各セクションの設定項目についての説明は、『*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。

4.3.2. 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>

int usermain(int argc, char *argv[])
{
    char *buf, svcname[16];
    long rlen;
    int failcount, i, ret;

    buf = tpalloc("STRING", NULL, 0);
    if (buf == NULL){
        error processing
    }

    while(1){
        tpschedule(10);
        /*Failキューの数を取得します。*/
        failcount = tpqstat("rqtest", TMAX_FAIL_QUEUE);
        for (i = 0; i < failcount ; i++){
            /*Failキューのデータを受信します。*/
            if (tpdeq("rqtest", _rq_sub_queue_name[TMAX_FAIL_QUEUE], &buf, &rlen,

                TPRQS) == -1){
                error processing
            }
            /*サービス名を取得します。*/
            if (tpextsvcname(buf, svcname) == -1){
                error processing
            }
            /* 読み込んだデータが"STOP"サービスの場合はスキップします。*/
            if (strcmp(svcname, "STOP")) {
                /*当該サービス名でtpenqします。*/
                ret=tpenq("rqtest", svcname, buf, rlen, TPRQS);
                if (ret<0) {
                    error processing
                }
            }
        }
    }

    return 1;
}

```

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RQ APIのプロトタイプとシステム変数を宣言したヘッダー・ファイルの内容を紹介します。Tmaxが提供するヘッダー・ファイルは次の場所に存在します。

```
TMAXDIR/usrinc
```

A.1. tmaxapi.h

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

#ifndef _TMAXAPI_H
#define _TMAXAPI_H

#include <sys/types.h>
#include <usrinc/atmi.h>
#ifdef _WIN32
#include <winsock2.h>
#include <usrinc/svct.h>
#include <usrinc/sdl.h>
#else
#include <sys/socket.h>
#define __cdecl
#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

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

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

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 __cdecl Unsolfunc(char *, long, long);
#define TPUNSOLERR ((Unsolfunc *) -1)

/* 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 */
};

#ifdef __cplusplus
extern "C" {
#endif

/* ----- unsolicited messaging API ----- */
long __cdecl tpsubscribe(char *eventexpr, char *filter, TPEVCTL *ctl,
                        long flags);

```



```

int __cdecl tpunsubscribe(long sd, long flags);
int __cdecl tppost(char *eventname, char *data, long len,
                  long flags);
int __cdecl tpbroadcast(char *lnid, char *usrname, char *cltname, char *data,
                       long len, long flags);
Unsolfunc *__cdecl tpsetunsol(Unsolfunc *func);
int __cdecl tpsetunsol_flag(int flag);
int __cdecl tpgetunsol(int type, char **data, long *len, long flags);
int __cdecl tpclearunsol(void);

/* ----- RQS API ----- */
int __cdecl tpenq(char *qname, char *svc, char *data, long len, long flags);
int __cdecl tpdeq(char *qname, char *svc, char **data, long *len, long flags);
int __cdecl tpqstat(char *qname, long type);
int __cdecl tpqsvcstat(char *qname, char *svc, long type);
int __cdecl tpextsvcname(char *data, char *svc);
int __cdecl tpextsvcinfol(char *data, char *svc, int *type, int *errcode);
int __cdecl tpreissue(char *qname, char *filter, long flags);
char *__cdecl tpsubqname(int type);

/* ----- server API ----- */
int __cdecl tpgetminsvr(void);
int __cdecl tpgetmaxsvr(void);
int __cdecl tpgetmaxuser(void);
int __cdecl tpgetsvrseqno(void);
int __cdecl tpgetmysvrid(void);
int __cdecl tpgetmaxuser(void);
int __cdecl tpsendtocli(int clid, char *data, long len, long flags);
int __cdecl tpgetclid(void);
int __cdecl tpgetpeer_ipaddr(struct sockaddr *name, int *namelen);
int __cdecl tpchkclid(int clid);
int __cdecl tmax_clh_maxuser(void);
int __cdecl tmax_chk_svcinfo(int cmd);

/* ----- etc API ----- */
int __cdecl tp_sleep(int sec);
int __cdecl tp_usleep(int usec);
int __cdecl tpset_timeout(int sec);
int __cdecl tmaxreadenv(char *file, char *label);
char *__cdecl tpgetenv(char* str);
int __cdecl tpputenv(char* str);
int __cdecl tpgetsockname(struct sockaddr *name, int *namelen);
int __cdecl tpgetpeername(struct sockaddr *name, int *namelen);
int __cdecl tpgetactivesvr(char *nodename, char **outbufp);
int __cdecl tperrordetail(int i);
int __cdecl tpreset(void);
int __cdecl tptobackup();

```

```

struct svglist *__cdecl tpmcall(char *qname, char *svc, char *data, long len,
                                long flags);
struct svglist *__cdecl tpgetsvglist(char *svc, long flags);
int __cdecl tpsvgcall(int svgno, char *qname, char *svc, char *data, long len,
                      long flags);
int __cdecl tpflush();
char *__cdecl tmaxlastsvc(char *idata, char *odata, long flags);

#ifdef _TMAX_KERNEL
/* ----- User supplied routines ----- */
int __cdecl tpsvrinit(int argc, char *argv[]);
int __cdecl tpsvrdone();
void __cdecl tpsvctimeout(TPSVCINFO *msg);
#endif

/*
   Internal functions: ONLY BE CALLED FROM AUTOMATICALLY
   GENERATED STUB FILES. DO NOT DIRECTLY CALL THESE FUNCTIONS.
*/
int __cdecl get_clhfd(void);
#ifdef _WIN32
int __cdecl _tmax_regfn(void *initFn, void *doneFn, void *timeoutFn,
                       void *userMainFn);
int __cdecl _tmax_regtab(int svcTabSz, _svc_t *svcTab, int funcTabSz,
                       void *funcTab);
int __cdecl _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 __cdecl _tmax_main(int argc, char *argv[]);
int __cdecl _double_encode(char *in, char *out);
int __cdecl _double_decode(char *in, char *out);
#endif

#ifdef __cplusplus
}
#endif

#endif /* end of _TMAXAPI_H */

```

索引

C

compute_rq_deqtime, 41

F

Failキュー, 2

R

Replyキュー, 2

Requestキュー, 2

RQ(Reliable Queue)システム, 1

RQS, 3

rqs, 14

rqsコマンド・オプション

[-c], 14

[-f], 14

[-l], 14

[-s], 14

RQシステム・フロー, 4

RQセクション

BOOT, 10

BUFFERING, 10

FILEPATH, 11

FSYNC, 10

MAX_MEMQCOUNT, 11

QSIZE, 10

RQ Name, 9

SVGNAME, 9

RQファイル・システム, 2

S

SVRGROUPセクション

CPC, 8

NODENAME, 7

SVGTYPE, 8

SVRGROUP Name, 7

TMSNAME, 8

T

tmadmin

サブマネージャー・モード, 13

マスター・マネージャー・モード, 13

tmboot, 12

tmbootオプション

[-q], 12

tmdown, 13

tmdownオプション

[-q], 13

tpdeq, 21

tpdeq_ctl, 36

tpenq, 18

tpenq_ctl, 31

tpextsvcinfol, 26

tpextsvcname, 25

tpqstat, 28

tpqsvcstat, 30

tpreissue, 23

か

環境設定

RQセクション, 9

RQトランザクション, 8

SVRGROUPセクション, 7

た

データ復旧作業, 3

