

Tmax TCacheガイド

Tmax v6.0



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

Copyright Notice

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

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

Restricted Rights Legend

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

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

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

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

Trademarks

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

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

Open Source Software Notice

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

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

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

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

文書情報

文書名: Tmax TCacheガイド

発行日: 2016年8月5日

ソフトウェアバージョン: Tmax v6.0

ガイドバージョン: v2.1.1

目次

このガイドについて	ix
第1章 はじめに	1
1.1. 概要	1
1.2. 環境設定	2
第2章 TCacheユーティリティ	5
2.1. pfmtcacheadmin	5
2.2. pfmtcachebackup	8
2.3. pfmtcacherestore	10
第3章 TCacheのAPI	13
3.1. 概要	13
3.2. pfmTCacheGet	13
3.3. pfmTCacheGetAll	16
3.4. pfmTCachePut	17
3.5. pfmTCacheInvalidate	18
3.6. pfmTCacheInvalidateAll	20
3.7. pfmTCacheItem	20
第4章 TCacheの同期化	23
4.1. 概要	23
4.2. マルチ・ドメインの初期化	24
4.3. マルチ・ドメインの初期化(COUSIN)	27
付録 A. エラーコード	31

図目次

[図 1.1]	TCacheの構造	1
[図 4.1]	同期化プロセス	23

このガイドについて

対象読者

本書は、TCacheを使って共有メモリー上でキャッシュ処理を行い、Tmax[®]（以下、Tmax）の性能を向上させようとするユーザーとシステム管理者を対象に記述しています。

前提知識

本書は、UNIX系（LINUXを含む）、Tmax[®]（以下、Tmax）、JEUS[®]（以下、JEUS）などへの知識をベースにして作成されました。したがって、本書の内容を円滑に理解するには、以下の事項を事前に熟知する必要があります。

- UNIX系（LINUXを含む）
- Tmaxに対する基本的な理解
- JEUSに対する基本的な理解

本書の構成

本書は、計4の章と付録で構成されています。

各章の主要内容は以下のとおりです。

- 第1章: はじめに

TCacheの基本構造とTmax上でTCacheを使用するための環境設定方法について簡単に説明します。

- 第2章: TCacheユーティリティ

TCacheを管理するためのユーティリティの使用方法について説明します。

- 第3章: TCacheのAPI

TCacheのAPIの使用方法と例題について説明します。

- 第4章: TCacheの同期化

TCacheの同期を取るマルチ・ドメインの初期化とマルチ・ノードの初期化について説明します。

- 付録A: エラーコード

TCacheのエラーコードの内容と対応法について説明します。

表記上の規則

表記	意味
<AaBbCc123>	プログラム・ソースコードのファイル名、ディレクトリー
<Ctrl>+C	CtrlキーとCキーを同時に押す
[Button]	GUIのボタン、メニュー名
太字	強調
「」、『』（鍵カッコ）	関連文書、あるいはガイド内の他の章および節の表示
「入力項目」	画面UI上の入力項目
<ハイパーリンク>	メール・アカウント、Webサイト
>	メニューの実行順
+----	下位ディレクトリー/ファイル有り
----	下位ディレクトリー/ファイル無し
参考	参照事項
注	注意事項
[図 1.1]	図の名称
[表 1.1]	表の名称
AaBbCc123	Javaコード、XMLドキュメント
[<i>command argument</i>]	オプション・パラメータ
< xyz >	「<」と「>」の間の内容は実際に使用される特定の名称または値で置き換えられる
	構文の中の相互に排他的な選択項目の選択肢を示す
...	パラメータ、値、または他の情報が繰り返される

システム要件

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

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 はじめに

本章では、TCacheの基本構造とTmax上でTCacheを使用するための環境設定方法について簡単に説明します。

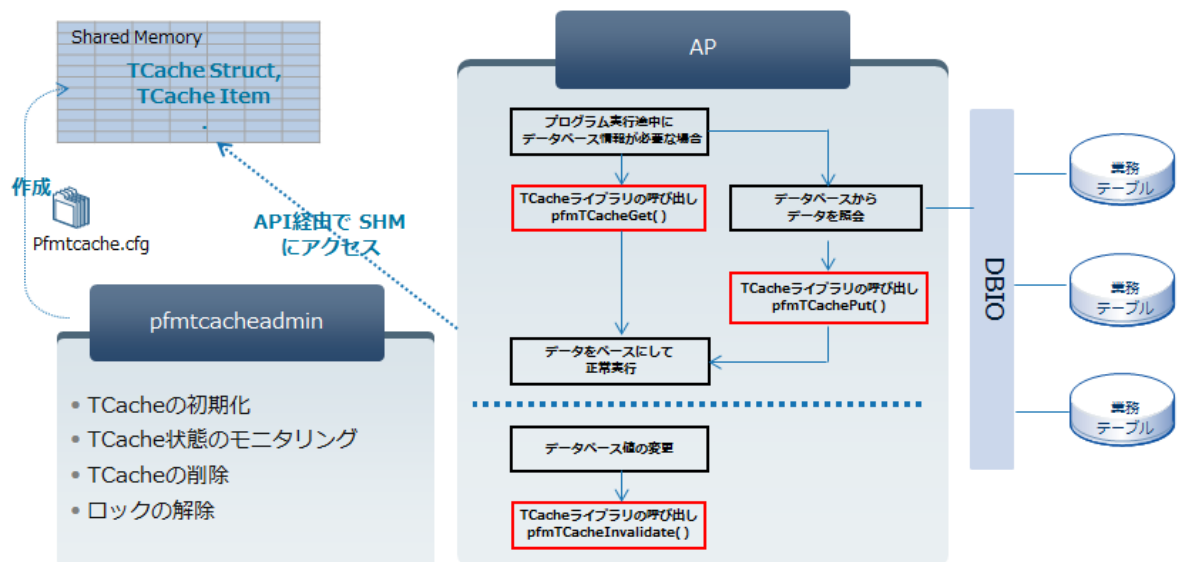
1.1. 概要

Tmaxはキャッシュ処理のためにTCacheを提供しています。TCacheとは、Tmaxの運用中にデータベースへのアクセスが必要な情報を共有メモリーにキャッシュしておくことで、Tmaxの性能を向上させるユーティリティのことです。TCacheを使用することでTmaxの性能が保証できる理由は、データベースへのデータ・リクエストが発生したときに毎回発生するデータベースへのアクセスによるオーバーヘッドを減少させることができるためです。

TCacheは共有メモリーにデータが存在しない場合にのみ、データベースからデータを読み込み、読み込んだデータはTCacheに蓄積しておきます。TCacheに蓄積された後に変更されるデータに対しては、初期化機能を使ってデータベースと同期を取ることができます。

以下は、TCacheの基本構造とプロセスを示した図です。

[図 1.1] TCacheの構造



- pfmtcacheadmin

pfmTCacheadminは基本的な動作を制御します。pfmTCacheadminよりTCacheを初期化したり、削除や作成したりすることができますし、TCacheの状態をモニタリングすることもできます。つまり、初期化した後から現在にいたるまでのキャッシュの状態についてのレポート結果を確認することができます。

- pfmTCCache.cfg

pfmTCCache.cfgはTCCacheの環境設定ファイルです。pfmTCCache.cfgファイルには、共有メモリー・アドレス、権限、キャッシュ対象の名前やメモリーのサイズ、hashテーブルのサイズ、hashキーのサイズ、保存したいデータのサイズなどを指定します。

- libpfmTCCache.so

libpfmTCCache.soはTCCacheライブラリーです。このライブラリーには実際に呼び出す関数がコンパイルされています。TCCacheを使用する場合は、ユーザーのソースをコンパイルする際に、このライブラリーがリンクに含まれている必要があります。

- TPFMAGENT

TPFMAGENTは、TCCacheが複数のノードに設定されて同期を取る必要がある際に、初期化機能を行うためのTmaxサーバです。

注

TCCacheの性能を確保するためには、書き取りが頻繁に起きないデータに対してTCCacheを使用することを推奨します。

1.2. 環境設定

pfmtcache.cfgはTCCacheの環境設定ファイルです。共有メモリーの構造情報で構成されています。

pfmtcache.cfgは以下のパスに位置します(環境ファイルの保存場所は、インストール先のディレクトリーがどこなのかによって変更される可能性があります)。

```
$TMAXDIR/config/pfmtcache.cfg
```

このファイルに共有メモリー・アドレス、権限、キャッシュ対象の名前やメモリーのサイズ、ハッシュ・テーブルのサイズ、ハッシュ・キーのサイズ、保存したいデータのサイズを指定します。

```
#####
# 共通領域
#####
SHMKEY      = [shared-memory-key]
IPCPRM      = [shared-memory-permission]
SIZE_LOCAL  = [cache-size]
LIB_PTHREAD = [file-name]
INVALIDATE_TYPE = [invalidate-type]
AGENT_SVC   = [agent_service]

#####
```

```
# 個別領域
#####
CACHE_NAME = [cache-name]
SIZE_MEM    = [cache-memory-size]
SIZE_HASH   = [hash-key-size]
SIZE_KEY     = [key-size]
SIZE_REC    = [record-size]
INV_TIMEOUT= [timeout-sec]
CALLBACK_NAME= [callback-name]
```

項目	説明
SHMKEY	UNIXの共有メモリーに付与したIDを設定します
IPCPREM	共有メモリーへのアクセス権限(user、group、otherへのアクセス権限)を設定します
SIZE_LOCAL	<p>キャッシュ情報をすべて共有メモリーに保存する一方で、その一部をサーバー・プロセスのローカル・メモリーにも記録して保持します。そうすることで、ゲット(Get)時に、共有メモリーより先にローカル・メモリーを検索するようにし、性能低下を防ぎます</p> <p>ローカル・メモリーに記録できるキャッシュの最大容量をKBで設定します</p>
LIB_PTHREAD	未使用(現在シンボリック・リンクのみ使用)
INVALIDATE_TYPE	初期化のタイプを設定します(「 第4章 TCacheの同期化 」参照)
AGENT_SVC	初期化したいサーバーを設定します(「 第4章 TCacheの同期化 」参照)
SHMKEY	UNIXの共有メモリーに付与したIDを設定します
SIZE_MEM	<p>キャッシュ名のために割り当てる共有メモリーのサイズの合計を設定します</p> <p>超過した場合、LRU(Least Recently Used)アルゴリズムを適用します(単位: KB)</p>
SIZE_HASH	ハッシュ・キーの範囲を設定します
SIZE_KEY	<p>照会キーとなる値の長さを設定します</p> <p>キーは10バイト、データは90バイトの構造体(struct {char Key [10]; char Data [90];})をTCacheに登録する場合、「SIZE_KEY」には10を設定します</p> <p>例外的に、複数のメンバーが一つのキーを構成するときには、パディング・ビットの影響により、単純に長さを合計したときとは、値が異なる場合があります</p> <p>そのキーの構造体に対してsizeof演算を行った値をキーに設定すると、正確なキーを設定することができます</p>
SIZE_REC	メモリー割り当ての最小単位であるスロットのサイズを設定します

項目	説明
	<p>キーは10バイト、データは90バイトの構造体(struct {char Key [10]; char Data [90];})をTCacheに登録する場合、「SIZE_REC」には100を設定します</p> <p>例外的に、複数のメンバーが一つのレコードを構成するときには、パディング・ビットの影響により、単純に長さを合計したときとは、値が異なる場合があります。ゲット(Get)したい構造体に対してsizeof演算を行った値をSIZE_RECに設定すると、一つのスロットに1つのレコードを設定することができます。一方、SIZE_RECの可変長を超えてプット(Put)されたレコードの場合は、SIZE_RECの単位で分けられてスロットに保存されます</p> <p>APIで使用するデータを取得する一時バッファのサイズを設定します</p>
INV_TIMEOUT	初期化した後、プット(Put)が禁止される期限を設定します
CALLBACK_NAME	<p>pfmTCacheItemがゲット(Get)に失敗した場合に備えて、照会用のビジネス・モジュール(BM)を設定します</p> <p>CALLBACK_NAMEは、内部の処理構造を反映し、ProFrame Cのビジネス・モジュールに制限します</p> <p>ビジネス・モジュールの物理名をGET_PFM_SVCで設定すれば、libGET_PFM_SVC.soという名前でライブラリー・ファイルが生成されます。この際、CALLBACK_NAMEにはビジネス・モジュールの物理名を設定します</p> <pre>CALLBACK_NAME=GET_PFM_SVC</pre>

以下は、TCacheの環境設定ファイルの例です。

```
# the configuration file of TCACHE
SHMKEY=0x70056          # 共有メモリーのキー
IPCPerm=0777            # 共有メモリーのアクセス権限
SIZE_LOCAL=1024          # L1 cache size in kilo-bytes
LIB_PTHREAD=libpthread.so # the pthread library file name
#INVALIDATE_TYPE=1       # Invalidate type 0:multi-node 1:multi-domain
#AGENT_SVC=SPFMAGENT|2   # multi-node sync. svc SPFMAGENT|2 = SPFMAGENT01,
SPFMAGENT02

# cache for PFM_SVC
CACHE_NAME=PFM_SVC
SIZE_MEM=65536           # the total cache memory size in kilo-bytes
SIZE_HASH=1024           # the number of hash key (MAX=6553)
SIZE_KEY=30              # the number of digits of the index column
SIZE_REC=2048            # the size of a single record in bytes
INV_TIMEOUT=10           # invalidation timeout in sec
#CALLBACK_NAME=GET_PFM_SVC # Collback Func Nmae libGET_PFM_SVC.so
```

第2章 TCacheユーティリティー

本章では、TCacheを管理するためのユーティリティーの使用方法について説明します。

2.1. pfmtcacheadmin

pfmtcacheadminユーティリティーを使用すれば、TCacheを初期化することや削除／作成すること、そしてTCacheの状態をモニタリングすることができます。また、初期化した後から現時点までのキャッシュの統計をレポートとして確認することもできます。

以下は、pfmtcacheadminの基本的な使い方です。

- 使用方法

```
$ pfmtcacheadmin [-s] [-S [sec]] [-c] [-d] [-r] [-R [sec]] [-C [sec]]  
                  [-U ALL] [-u [cacheName]] [-t] [-i[cacheName]]
```

項目	説明
[-s]	TCacheの動作関連のレポートを作成するオプションです。メモリーの使用状況とTCacheの動作状態、キャッシュ・ヒットとキャッシュ・ミスの回数に関する情報を示します TCacheの性能を保証するには、Avg List値を5以下で維持する必要があります なお、Avg List値を減らすには、SIZE_HASH値を増やす必要があります
[-S [sec]]	-sオプションと同じです（周期的に実行）
[-c]	TCacheを起動させます。TCacheがすでに共有メモリーにロードされている場合は、何の作業も行いません
[-d]	TCacheを共有メモリーから削除するオプションです。TCacheを削除できるのは、TCacheを最初に作成した（pfmtcacheadmin -cを実行した）ユーザーだけです
[-r]	TCacheを再生成するオプションです。[-d]や[-c]と同じ動作を行います
[-R [sec]]	TCacheに解除されていないロックが存在するときにTCache全体を再作成します （周期的に実行）

項目	説明
[-C [sec]]	TCacheに解除されていないロックが存在するかどうかを確認します(周期的に実行)
[-U ALL]	すべてのTCacheのロックを解除します
[-u [cacheName]]	TCacheの名前別にロックを解除します
[-t]	TCacheに保持されているキャッシュ・データは保存しおき、TCacheのAPIを一時的に使用できないようにするオプションです 再度APIを使用できるようにするには、[-t]を再実行します
[-i [cacheName]]	特定のTCacheのデータを初期化するためのオプションです ここでいうcacheNameとは、pfmtcache.cfgに定義しておいた「CACHE_NAME」のことです 複数のキャッシュを使用している場合に、特定のキャッシュ領域だけを初期化するために使用します

● 実行結果

– [-c]

```
[midas:$TMAXDIR/bin]$ pfmtcacheadmin -c
TCACHE ver. 2.3.5
ERROR : TCACHE is already created.
```

– [-d]

```
[midas:$TMAXDIR/bin]$ pfmtcacheadmin -d
TCACHE ver. 2.3.5
This will delete TCACHE. Are you sure? [y/N] y
TCACHE deleted successfully.
```

– [-r]

```
[midas:$TMAXDIR/bin]$ pfmtcacheadmin -r
TCACHE ver. 2.3.5
This will clear and initialize all TCACHE contents. Are you sure? [y/N] y
The previous TCACHE removed successfully
New TCACHE initialized successfully.
```

– [-t]


```
[midas:$TMAXDIR/bin]$ pfmtcacheadmin -t
TCACHE ver. 2.3.5
TCACHE status changed to NO ACCESS.
```

- [-i *[cacheName]*]

```
[midas:$TMAXDIR/bin]$ pfmtcacheadmin -i PFM_SVC
TCACHE ver. 2.3.5
This will invalidate all contents in PFM_SVC. Are you sure? [y/N] y
```

- [-s]

```
[midas:$TMAXDIR/bin]$ pfmtcacheadmin -s
TCACHE ver. 2.3.5
Reporting TCACHE statistics. [SHMKEY = 0x70056]
version = 2.3.5
# of cache object = 8
status = AVAILABLE
local cache size = 10485760 bytes
Invalidate Type = 0
Tcache Agent Service =

CACHE_NAME   = PFM_SVC
SIZE_MEM     = 6710272 bytes
SIZE_HASH    = 1024 hash lists
SIZE_KEY     = 30 digits
SIZE_REC     = 2048 bytes per single record
INV_TIMEOUT  = 10 sec.
USE_HASH_FREE_LIST = 0
USE_SEM_LOCK = 0
Total # of slots          = 2941 slots
Total # of used slots     = 8 slots
Total # of free slots     = 2933 slots
Total # of GET req. (hit, miss, lock-fail) = 0 (0, 0, 0) times
Total # of PUT req. (repl, out-mem, lock-fail) = 8 (0, 0, 0) times
Total # of INV req. (lock-fail) = 0 (0) times
Total # of null-list.    = 1016 lists(Avg List : 1.00)

CACHE_NAME   = PFM_SVC_EXT
SIZE_MEM     = 6710272 bytes
SIZE_HASH    = 1024 hash lists
SIZE_KEY     = 30 digits
SIZE_REC     = 2048 bytes per single record
INV_TIMEOUT  = 10 sec.
USE_HASH_FREE_LIST = 0
USE_SEM_LOCK = 0
```

```

Total # of slots          = 2941 slots
Total # of used slots     = 7 slots
Total # of free slots     = 2934 slots
Total # of GET req. (hit, miss, lock-fail)      = 0 (0, 0, 0) times
Total # of PUT req. (repl, out-mem, lock-fail)  = 7 (0, 0, 0) times
Total # of INV req. (lock-fail)                = 0 (0) times
Total # of null-list.                          = 1017 lists(Avg List : 1.00)

```

2.2. pfmtcachebackup

pfmtcachebackupユーティリティを使用すれば、共有メモリーにロードされているTCacheのアイテムをバックアップすることができます。

- 使用方法

```
$ pfmtcachebackup -t [ALL | ITEM_NAME | PFM_*]
```

項目	説明
-t ALL	アイテム全体をバックアップします
-t [ITEM_NAME]	特定のアイテムをバックアップします
-t PFM_*	開始文字を基準に、ワイルドカード(「*」のみサポート)文字にマッチするアイテムをバックアップします

- 実行結果

– [-t ALL]

```

[midas:$TMAXDIR/bin]$ pfmtcachebackup -t ALL
===== START TCache UNLOAD[PFM_SVC]
=====
      Successfully Finished
=====
UNLOAD TCache NAME : [PFM_SVC]
START      TIME : [2013-07-04 09:38:38]
END        TIME : [2013-07-04 09:38:38]
-----
WRITE  FILE  NAME : [./20130704/PFM_SVC]
TOTAL   CNT   : [      8]
SUCCESS CNT   : [      8]
FAIL    CNT   : [      0]
=====
===== START TCache UNLOAD[PFM_SVC_EXT]

```

```

=====
      Successfully Finished
=====
UNLOAD TCACHE NAME : [PFM_SVC_EXT]
START      TIME : [2013-07-04 09:38:38]
END        TIME : [2013-07-04 09:38:38]
-----
WRITE  FILE  NAME : [./20130704/PFM_SVC_EXT]
TOTAL      CNT : [          7]
SUCCESS    CNT : [          7]
FAIL       CNT : [          0]
=====

```

– [-t PFM_SVC]

```

[midas:$TMAXDIR/bin]$ pfmtcachebackup -t PFM_SVC
===== START TCACHE UNLOAD[PFM_SVC]
=====
      Successfully Finished
=====
UNLOAD TCACHE NAME : [PFM_SVC]
START      TIME : [2013-07-04 09:39:06]
END        TIME : [2013-07-04 09:39:06]
-----
WRITE  FILE  NAME : [./20130704/PFM_SVC]
TOTAL      CNT : [          8]
SUCCESS    CNT : [          8]
FAIL       CNT : [          0]
=====

```

– [-t PFM_*]

```

[midas:$TMAXDIR/bin]$ pfmtcachebackup -t PFM_*
===== START TCACHE UNLOAD[PFM_SVC]
=====
      Successfully Finished
=====
UNLOAD TCACHE NAME : [PFM_SVC]
START      TIME : [2013-07-04 09:39:29]
END        TIME : [2013-07-04 09:39:29]
-----
WRITE  FILE  NAME : [./20130704/PFM_SVC]
TOTAL      CNT : [          8]
SUCCESS    CNT : [          8]
FAIL       CNT : [          0]
=====

```

```

===== START TCACHE UNLOAD[PFM_SVC_EXT]
=====
      Successfully Finished
=====
UNLOAD TCACHE NAME : [PFM_SVC_EXT]
START      TIME : [2013-07-04 09:39:29]
END        TIME : [2013-07-04 09:39:29]
-----
WRITE  FILE  NAME : [./20130704/PFM_SVC_EXT]
TOTAL      CNT : [      7]
SUCCESS    CNT : [      7]
FAIL       CNT : [      0]
=====

```

2.3. pfmtcacherestore

pfmtcacherestoreを使用すれば、バックアップしたファイルをTCacheのアイテムにロードする機能を提供します。

- 使用方法

```
$ pfmtcacherestore -d [Date : YYYYMMDD] -t [ALL | ITEM_NAME | PFM_*]
```

項目	説明
-d [Date : YYYYMMDD]	ロードしたいバックアップ・ファイルの日付を設定します
-t ALL	バックアップ・ファイルのアイテム全体をロードします
-t [ITEM_NAME]	バックアップ・ファイルのうち、特定のアイテムをロードします
-t PFM_*	バックアップ・ファイルのうち、開始文字を基準に、ワイルドカード(「*」のみサポート)文字にマッチするアイテムをロードします

- 実行結果

– -d [YYYYMMDD] -t ALL

```

[midas:$TMAXDIR/bin]$ pfmtcacherestore -d 20130704 -t ALL
=====>[8]
===== START TCACHE LOAD[PFM_SVC]
=====
      Successfully Finished
=====
LOAD  TCACHE NAME : [PFM_SVC]
START      TIME : [2013-07-04 09:40:10]

```

```

END                TIME : [2013-07-04 09:40:10]
-----
WRITE   FILE  NAME : [./20130704/PFM_SVC]
TOTAL           CNT : [          8]
SUCCESS        CNT : [          8]
FAIL           CNT : [          0]
=====

===== START TCACHE LOAD[PFM_SVC_EXT]
=====

      Successfully Finished
=====

LOAD   TCACHE NAME : [PFM_SVC_EXT]
START           TIME : [2013-07-04 09:40:10]
END            TIME : [2013-07-04 09:40:10]
-----
WRITE   FILE  NAME : [./20130704/PFM_SVC_EXT]
TOTAL           CNT : [          7]
SUCCESS        CNT : [          7]
FAIL           CNT : [          0]
=====

```

– -d [YYYYMMDD] -t [ITEM_NAME]

```

[midas:$TMAXDIR/bin]$ pfmtcacherestore -d 20130704 -t PFM_SVC
=====>[8]
===== START TCACHE LOAD[PFM_SVC]
=====

      Successfully Finished
=====

LOAD   TCACHE NAME : [PFM_SVC]
START           TIME : [2013-07-04 09:41:44]
END            TIME : [2013-07-04 09:41:44]
-----
WRITE   FILE  NAME : [./20130704/PFM_SVC]
TOTAL           CNT : [          8]
SUCCESS        CNT : [          8]
FAIL           CNT : [          0]
=====

```

– -d [YYYYMMDD] -t -t PFM_*

```

[midas:$TMAXDIR/bin]$ pfmtcacherestore -d 20130704 -t PFM_*
=====>[8]
===== START TCACHE LOAD[PFM_SVC]
=====

      Successfully Finished

```

```

=====
LOAD    TCACHE NAME : [PFM_SVC]
START      TIME : [2013-07-04 09:40:33]
END        TIME : [2013-07-04 09:40:33]
-----

WRITE    FILE  NAME : [./20130704/PFM_SVC]
TOTAL      CNT  : [      8]
SUCCESS    CNT  : [      8]
FAIL       CNT  : [      0]
=====

===== START TCACHE LOAD[PFM_SVC_EXT]
=====
      Successfully Finished
=====

LOAD    TCACHE NAME : [PFM_SVC_EXT]
START      TIME : [2013-07-04 09:40:33]
END        TIME : [2013-07-04 09:40:33]
-----

WRITE    FILE  NAME : [./20130704/PFM_SVC_EXT]
TOTAL      CNT  : [      7]
SUCCESS    CNT  : [      7]
FAIL       CNT  : [      0]
=====

```

第3章 TCacheのAPI

本章では、TCacheのAPIの使用方法和例題について説明します。

3.1. 概要

以下は、TCacheが提供するAPIについての説明です。

API	説明
pfmTCacheGet	TCacheの共有メモリーにアイテムの値を設定します
pfmTCacheGetAll	TCacheが管理するcache_nameテーブルに存在するすべてのレコード・データをmemバッファ上sizeの分だけ読み込みます
pfmTCachePut	TCacheが管理するメモリーにデータを保存するためのAPIです
pfmTCacheInvalidate	TCacheが管理するメモリーを初期化するためのAPIです。cache_nameテーブル内のキーに対応するレコード・データを初期化します
pfmTCacheInvalidateAll	TCacheが管理するメモリーを初期化するためのAPIです。cache_nameテーブル内のすべてのレコード・データを初期化します
pfmTCacheItem	ProFrame専用のAPIです。pfmTCacheGetとpfmTCachePutを一つの機能にして実装したAPIです

3.2. pfmTCacheGet

TCacheの共有メモリーのアイテムの値を設定します。TCacheからデータを読み込む際には、READロックを取得する必要があります。No-Waitモードでは、一度トライしてロックの取得に失敗したら、すぐにエラーを戻します。一方、Waitモードだと、一定回数のトライをした後もロック取得に失敗した場合に、エラーを戻します。TCacheの全体的な性能向上のために、基本的にフラグは0 (No-Waitモード) に設定します。

● プロトタイプ

```
long pfmTCacheGet( const char* cacheName, const char* key, void* mem,
                  const unsigned long size, long flags )
```

● パラメーター

パラメーター	説明
cacheName	pfmtcache.cfgで定義した「CACHE_NAME」です

パラメーター	説明
key	入力データの先頭から計り、pfmtcache.cfgファイルに記録された「SIZE_KEY」の長さまでのデータです
mem	保存されたデータを読み込む一時バッファです
size	データを読み込むバッファのサイズを設定します
flags	ブロッキング・モードを決定するためのフラグです – 0:No-Wait(デフォルト値) – 1:Wait

- 戻り値

成功した場合には0を返し、失敗した場合には負数を返します。エラーコードの詳細については、「[付録A. エラーコード](#)」を参照してください。

- 例題

```
rc = pfmTCCacheGet("PFM_SVC", data, get_data, sizeof(get_data), 0);
if(rc != 0 ) {
    printf("PFM_SVC pfmTCCacheGet() fail!!\n");
}else{
    printf("PFM_SVC [%s] Get success!!\n", get_data);
}
```

- 注意事項

データを保存するバッファのmemのサイズがTCCacheに存在するデータよりも小さい場合には、エラーを返さずに、4番目の引数のsizeofの分のデータを読み込み、正常値として処理されるので注意が必要です。

使用例

キャッシュの名前がPFM_SVCであり、「PFM4 ORDERBOOK_SMPF0001_S0010」というキーのデータをゲット(Get)する例です。

キャッシュ・キーに対応するデータが存在しない場合、DBIOのPfmSvcSel000を利用し、DB selectを実行して取得したデータをPFM_SVCキャッシュにプット(Put)します。

```
struct PfmSvcSel000In {
    char projectId[LEN_PFMSVCSEL000IN_PROJECTID + 1];
    char instNo    [LEN_PFMSVCSEL000IN_INSTNO    + 1];
}
```



```

        char txCode    [LEN_PFMSVCSEL000IN_TXCODE    + 1];
};
typedef struct PfmSvcSel000In PfmSvcSel000In;

struct PfmSvcSel000Out {
    char instNo[LEN_PFMSVCSEL000OUT_INSTNO + 1];
    char txCode[LEN_PFMSVCSEL000OUT_TXCODE + 1];
    char applySdate[LEN_PFMSVCSEL000OUT_APPLYSDATE + 1];
    char applyStime[LEN_PFMSVCSEL000OUT_APPLYSTIME + 1];
    char applyEdate[LEN_PFMSVCSEL000OUT_APPLYEDATE + 1];
    char applyEtime[LEN_PFMSVCSEL000OUT_APPLYETIME + 1];
    char stopFlag[LEN_PFMSVCSEL000OUT_STOPFLAG + 1];
    char stopSdate[LEN_PFMSVCSEL000OUT_STOPSDATE + 1];
    char stopStime[LEN_PFMSVCSEL000OUT_STOPSTIME + 1];
    char stopEdate[LEN_PFMSVCSEL000OUT_STOPEDATE + 1];
    char stopEtime[LEN_PFMSVCSEL000OUT_STOPETIME + 1];
    char calleeName[LEN_PFMSVCSEL000OUT_CALLEENAME + 1];
    long staType;
    char txName[LEN_PFMSVCSEL000OUT_TXNAME + 1];
    char svcName[LEN_PFMSVCSEL000OUT_SVCNAME + 1];
    char dummySvcName[LEN_PFMSVCSEL000OUT_DUMMYSVCNAME + 1];
    char inStructName[LEN_PFMSVCSEL000OUT_INSTRUCTNAME + 1];
    char inMapName[LEN_PFMSVCSEL000OUT_INMAPNAME + 1];
    char outStructName[LEN_PFMSVCSEL000OUT_OUTSTRUCTNAME + 1];
    char outMapName[LEN_PFMSVCSEL000OUT_OUTMAPNAME + 1];
    char logType[LEN_PFMSVCSEL000OUT_LOGTYPE + 1];
    long apLogProcess;
    long errorLogProcess;
    char traceLogLevel[LEN_PFMSVCSEL000OUT_TRACELOGLEVEL + 1];
    char debugLogLevel[LEN_PFMSVCSEL000OUT_DEBUGLOGLEVEL + 1];
    char traceLevel[LEN_PFMSVCSEL000OUT_TRACELEVEL + 1];
    long threshold;
    long sqlTraceLevel;
};
typedef struct PfmSvcSel000Out PfmSvcSel000Out;

PfmSvcSel000In      SVCInput;
PfmSvcSel000Out     SVCOutput;
memset(&SVCInput, 0x00, sizeof(PfmSvcSel000In));
memset(&SVCOutput, 0x00, sizeof(PfmSvcSel000Out));
memcpy(SVCInput.projectId, "MDS", 6);
memcpy(SVCInput.tx_code, "SPFM0008A001", 24);
memcpy(SVCInput.inst_no, "PFM", 3);

rc = pfmTCacheGet("PFM_SVC", &SVCInput, &SVCOutput,
    (unsigned int)sizeof(PfmSvcSel000Out), 0);
if( rc != RC_NRM){ /* RC_NRM = 0L : 正常時 */

```

```

PFM_TRYNJ( pfmDbioSelect("PFM_SVC_PS0001Output ", &SVCInput, &SVCOutput, NULL,
                        PFMDBIO_NOLOCK));
if( rc != RC_NRM){
    if( rc == PDB_NOTFOUND) /* 1403L :DBで見つからなかった場合に発生 */
        PFM_ERR("SvcParamDataNotFound projectId[%s], tx_code[%s], inst_no[%s]",
                SVCInput.projectId, SVCInput.tx_code, SVCInput.inst_no);
    else
        PFM_ERR("SvcParamTableQueryFail pfmDbioGetErrorNo[%ld],
                pfmDbioGetErrorString[%s]",
                pfmDbioGetErrorNo(), pfmDbioGetErrorString());
}
else{
    rc = pfmTCachePut("PFM_SVC", &SVCOutput, (unsigned int)sizeof(PfmSvcSel000Out),
0);
    if( rc != RC_NRM) /* RC_NRM = 0L : 正常時 */
        PFM_DBG("pfmTCachePutError[%s] rc[%ld]", "PFM_SVC", rc);
    else
        PFM_DBG("pfmTCachePutSuccess [%s]", "PFM_SVC");
}
}
else{
    PFM_DBG("pfmTCacheHitSuccess [%s]", "PFM_SVC");
}
}

```

3.3. pfmTCacheGetAll

TCacheが管理するcache_nameテーブルに存在するすべてのレコード・データをmemバッファ上sizeの分だけ読み込みます。

- プロトタイプ

```

long pfmTCacheGetAll( const char* cacheName, void* mem,
                    unsigned long size, long flags )

```

- パラメーター

パラメーター	説明
cacheName	pfmtcache.cfgで定義した「CACHE_NAME」です
mem	保存されたデータを読み込む一時バッファです
size	データを読み込むバッファのサイズを設定します
flags	– 0:No-Wait(デフォルト値) – 1:Wait

- 出力レイアウト

```
保存されているレコードの数(unsigned int : 8byte), record1のサイズ(unsigned int : 8byte),
record1のデータ, record2のサイズ(unsigned int : 8byte), record2    , ...
```

- 戻り値

成功した場合には0を返し、失敗した場合には負数を返します。エラーコードの詳細については、「[付録A. エラーコード](#)」を参照してください。

- 例題

```
char * get_data;
char * record_count;
get_data = (char*)malloc(10000);

rc = pfmTCCacheGetAll("PFM_SVC", get_data, 10000, 0);
if(rc != 0 ) {
    printf("PFM_SVC pfmTCCacheGetAll() fail!!\n");
}else{
    printf("PFM_SVC GetAll success!!\n");
    memcpy(&record_count, get_data, sizeof(unsigned int));
    printf("PFM_SVC Get Record Count[%ld]\n", record_count);
}
```

- 注意事項

データを読み込むバッファのサイズがTCCacheに保存されたデータよりも小さい場合は、エラーを返さずに、3番目の引数のsizeofの分のデータを読み込み、正常値として処理されるので注意が必要です。pfmTCCacheGetAllItems() APIは、ブロック・モードを使用するので、pfmTCCacheInvalidate()のようにブロック・モードで動作するAPIが頻繁に呼び出される環境では、速度の低下が起こりかねません。

3.4. pfmTCCachePut

TCCacheが管理するメモリーにデータを保存するためのAPIです。cache_nameテーブルでキー値に対応するレコード・データをmemバッファにsizeの分だけ読み込みます。

- プロトタイプ

```
long pfmTCCachePut( const char* cacheName, void* mem,
                   const unsigned long size, long flags )
```

- パラメーター

パラメーター	説明
cacheName	pfmtcache.cfgで定義した「CACHE_NAME」です
mem	保存されたデータを読み込む一時バッファです
size	データを読み込むバッファのサイズを設定します
flags	<ul style="list-style-type: none"> 0: No-Wait (デフォルト値) 1: Wait

- 戻り値

成功した場合には0を返し、失敗した場合には負数を返します。エラーコードの詳細については、[「付録 A. エラーコード」](#)を参照してください。

- 例題

```
rc = pfmTCCachePut("PFM_SVC", set_data, sizeof(set_data), 0);
if(rc != 0 ) {
    printf("PFM_SVC pfmTCCachePut() fail!!\n");
}else{
    printf("PFM_SVC [%s] Put success!!\n", set_data);
}
```

- 注意事項

保存されるデータの最初のバイトからpfmtcache.cfgで「SIZE_KEY」に設定した値の分までの情報をTCCacheのキーとして利用するので、キーに関する情報を、保存しようとするデータの一番前に位置するようにします。

使用例

[「3.2. pfmTCCacheGet」](#)の例を参照してください。

3.5. pfmTCCacheInvalidate

TCCacheが管理するメモリーを初期化するためのAPIです。cache_nameテーブル内のキーに対応するレコード・データを初期化します。

- プロトタイプ

```
long pfmTCCacheInvalidate( const char* cacheName, const char* key, long flags )
```

- パラメーター

パラメーター	説明
* cacheName	pfmtcache.cfgで定義した「CACHE_NAME」です
* key	入力データの先頭から計り、pfmtcache.cfgファイルに記録された「SIZE_KEY」の長さまでのデータです
size	データを読み込むバッファのサイズを設定します
flags	初期化される際に、内部的に使用するフラグです。基本的に「0」に設定します

- 戻り値

成功した場合には0を返し、失敗した場合には負数を返します。エラーコードの詳細については、[「付録A. エラーコード」](#)を参照してください。

- 例題

```
rc = pfmTCCacheInvalidate("PFM_SVC", inv_data, sizeof(inv_data), 0);
if(rc != 0 ) {
    printf("PFM_SVC pfmTCCacheInvalidate() fail!!\n");
}else{
    printf("PFM_SVC [%s] Invalidate success!!\n", inv_data);
}
```

使用例

「tb_mti700」というDBIOを更新した後、同期を取るためにTCCacheを初期化する例です。

```
tb_mti700_pu999In    mti700_pu999In;
bzero(&mti700_pu999In, sizeof(tb_mti700_pu999In));
mti700_pu999In.isu_id = 20231;

strncpy(mti700_pu999In.isu_cd, "ks00000000", sizeof(mti700_pu999In.isu_cd));
rc = pfmDbioAmend("tb_mti700_pu999", &mti700_pu999In, NULL, PFMDBIO_NOLOCK);
if(rc != RC_NRM) {
    PFM_ERR("TRE0001020", "アップデートに失敗しました。");
}
else {
    PFM_DBG("====TCCache invalidate====");
    PFM_TRYNJ(pfmTCCacheInvalidate("TB_MTI700", &mti700_ps999In, 0));
    if(rc != RC_NRM) {
        if(rc == RC_NFD) {
            /* TCCacheにキャッシュ・データが存在しない場合 */

```

```

        PFM_DBG("pfmTCCacheInvalidate NOT Found");
    }
    else {
        /* TCacheのinvalidateエラーの処理 */
        PFM_ERR("TRE0001021", "pfmTCCacheInvalidate ERROR! rc[%ld]", rc);
        return RC_ERR;
    }
} else {
    PFM_DBG("pfmTCCacheInvalidate SUCCESS");
}
}
}

```

3.6. pfmTCCacheInvalidateAll

TCacheが管理するメモリーを初期化するためのAPIです。cache_nameテーブル内のすべてのレコード・データを初期化します。

- プロトタイプ

```
long pfmTCCacheInvalidateAll( const char* cacheName, long flags )
```

- パラメーター

パラメーター	説明
cacheName	pfmtcache.cfgで定義した「CACHE_NAME」です
flags	初期化される際に、内部的に使用するフラグです。基本的に「0」に設定します

- 戻り値

成功した場合には0を返し、失敗した場合には負数を返します。エラーコードの詳細については、[「付録A. エラーコード」](#)を参照してください。

- 例題

```

rc = pfmTCCacheInvalidateAll("PFM_SVC", 0);
if(rc != 0 ) {
    printf("PFM_SVC pfmTCCacheInvalidateAll() fail!!\n");
}else{
    printf("PFM_SVC Invalidate All success!!\n");
}

```

3.7. pfmTCCacheItem

ProFrame専用のAPIです。pfmTCCacheGetとpfmTCCachePutを一つの機能にして実装したAPIです。

pfmTCacheItemと関連付けられている\$TCACHECONFはCALLBACK_NAMEのパラメーター値です。設定されたCALLBACK_NAMEがGET_SVC_BM(「CALLBACK_NAME = GET_PFM_SVC」)である場合を例に挙げます。pfmTCacheGetを行ってデータが見つからなかった場合、GET_PFM_SVC BMからデータを取得し、pfmTCachePutを行うロジックを行います。この際、DICallの対象となるライブラリーとしてはlibGET_PFM_SVC.soが呼び出されます。

内部的な処理構造により、IN／OUTを持つProFrame Cのビジネス・モジュールに制限します。そのビジネスモジュールのInput構造体のサイズはSIZE_KEYと対応され、Output構造体のサイズはSIZE_RECと対応されます。CALLBACK_NAMEはビジネス・モジュールの物理名である「CALLBACK_NAME = GET_PFM_SVC」に設定します。

● プロトタイプ

```
long pfmTCacheItem( const char* cacheName, const char* key, void* mem,
                   const unsigned long size, long flags );
```

● パラメーター

パラメーター	説明
cacheName	pfmtcache.cfgで定義した「CACHE_NAME」です
key	入力データの先頭から計り、pfmtcache.cfgファイルに記録された「SIZE_KEY」の長さまでのデータです
mem	保存されたデータを読み込む一時バッファです
size	データを読み込むバッファのサイズを設定します
flags	ブロッキング・モードを決定するためのフラグです <ul style="list-style-type: none"> – 0: No-Wait (デフォルト値) – 1: Wait

● 戻り値

成功した場合には0を返し、失敗した場合には負数を返します。

● 例題

– pfmtcache.cfgの設定値

```
# cache for PFM_SVC
CACHE_NAME=PFM_SVC
SIZE_MEM=6553           # the total cache memory size in kilo-bytes
SIZE_HASH=1024          # the number of hash key (MAX=6553)
SIZE_KEY=30             # the number of digits of the index column
SIZE_REC=2048           # the size of a single record in bytes
```

```
INV_TIMEOUT=10           # invalidation timeout in sec
CALLBACK_NAME=GET_SVC_PARAM # libGET_SVC_PARAM.so
```

– Source Code

```
rc = pfmTCacheItem("PFM_SVC", data, get_data, sizeof(get_data), 0);
if(rc != 0 ) {
    printf("PFM_SVC pfmTCacheItem() fail!!\n");
}else{
    printf("PFM_SVC [%s] Get success!!\n", get_data);
}
```


第4章 TCacheの同期化

本章では、TCacheの同期を取るマルチ・ドメインの初期化 (Multi Domain Invalidation) とマルチ・ノードの初期化 (Multi Node Invalidation) について説明します。

4.1. 概要

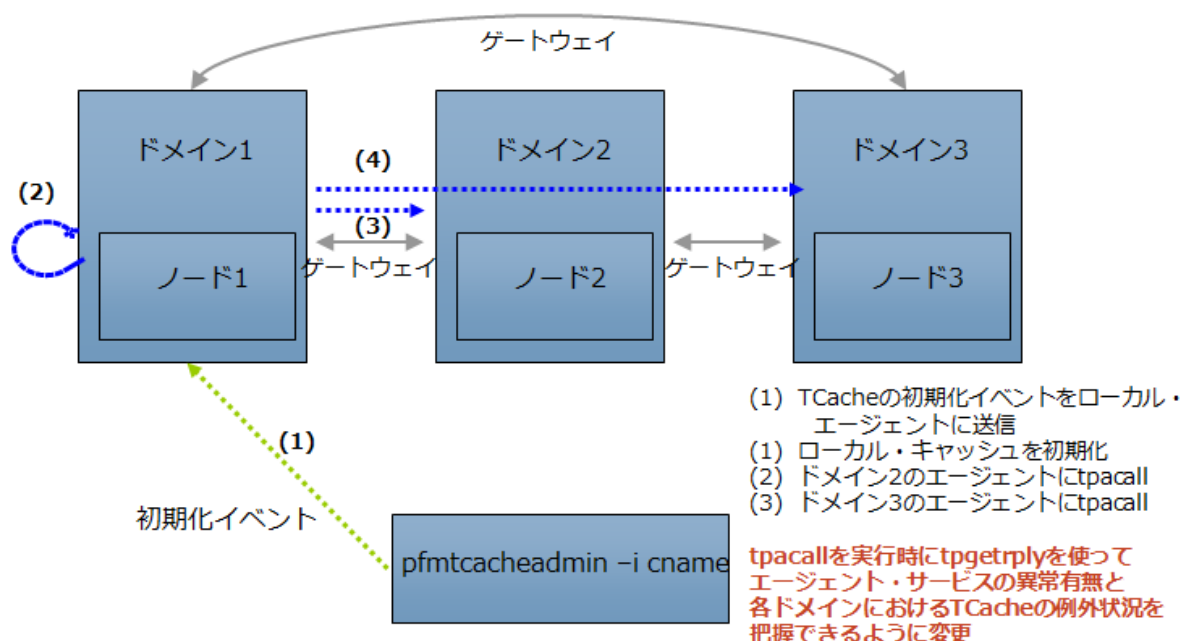
TCacheを使用中のデータベースのデータを変更する場合、TCacheのデータとデータベースのテーブルの間で同期を取る必要があります。TCacheを使用するデータに対して更新 (update) や削除 (delete) を実行するときは、プログラムで初期化を行う必要があります。

一方、データベースに直接アクセスして実行する場合には、以下のコマンドを使用して、強制的に初期化を行う必要があります。この場合、キャッシュ・オブジェクトがすべて削除されるので注意が必要です。

```
pfmtcacheadmin -i [CacheName]
```

以下の図は、TCacheの同期処理のプロセスを示しています。

【図 4.1】同期化プロセス



複数のノード上でTCacheの同期を取る場合も、やはり、それぞれのノードに対して初期化を行う必要があります。こうした場合には、TPFMAGENTというTmaxサーバーを利用して初期化を行い、同期を取ってください。

い。詳細設定については、「[4.2. マルチ・ドメインの初期化](#)」と「[4.3. マルチ・ドメインの初期化\(COUSIN\)](#)」を参照してください。

4.2. マルチ・ドメインの初期化

各ノードの間でtpacallができるようにドメイン・ゲートウェイが設定されている環境で、マルチ・ドメインの初期化を設定する場合です。初期化プロセスは、[\[図 4.1\]](#)と同じです。技術的な詳細については、Tmaxガイドを参照してください。

Tmaxサーバーの設定(ノード1)

```
#####
#
#      Sample Configuration File for Tmax System      #
#      =====#
#
#      Copyright(c) 2015 TmaxSoft Inc. All rights reserved      #
#
#####

*DOMAIN
tmax1      SHMKEY =79000, MINCLH=1, MAXCLH=3,
           TPORTNO=9999, BLOCKTIME=30

*NODE
fw2test      TMAXDIR = "/home/tmax/tmax",
           APPDIR  = "/home/tmax/tmax/appbin",
           PATHDIR = "/home/tmax/tmax/path",
           TLOGDIR = "/home/tmax/tmax/log/tlog",
           ULOGDIR = "/home/tmax/tmax/log/ulog",
           SLOGDIR = "/home/tmax/tmax/log/slog"

*SVRGROUP
NFRGRP01      NODENAME = "fw2test"

*SERVER
##### TCache Domain SERVER#####
TPFMAGENT      SVGNAME = NFRGRP01

#### TCache alias name ####
TCACHEFR02      SVGNAME = NFRGRP01, MIN=1, MAX=5, TARGET = TPFMAGENT,
           CLOPT = "-r -l -x SPFMAGENT:SPFMAGENT02 -o
$(SVR)_$(YEAR)$(MONTH)$(DAY).log -e $(SVR)_$(YEAR)$(MONTH)$(DAY).log"
```

```

*SERVICE
##### TCache Service #####
SPFMAGENT          SVRNAME = TPFMAGENT,    SVCTIME=30
SPFMAGENT02        SVRNAME = TCACHEFR02
SPFMAGENT01        SVRNAME = TGW02
# TCache ConfigのSPFMAGENT|2(SPFMAGENT01, SPFMAGENT02)はMatchされます。
# 内部のLocalについてはSPFMAGENT02がCallされ、RemoteについてはSPFMAGENT01がCallされて
# Invalidateされます。
# 技術的な詳細については、Tmaxガイドを参照してください。

##### TCache Gateway #####
*GATEWAY
TGW02              NODENAME = fw2test, GWTYPE=TMAXNONTX, PORTNO=9302,
                   RGWADDR = "192.168.1.51", RGWPORTNO=9301, RESTART=Y,
                   MAXRSTART=128, CLOPT="-i", CPC=8

```

TCacheの環境設定(ノード1)

```

# the configuration file of TCACHE

SHMKEY=0x70005      # the key of shared memory
IPCPERM=0777        # permission of the shared memory
SIZE_LOCAL=1024     # L1 cache size in kilo-bytes
LIB_PTHREAD=libpthread.so  # the pthread library file name

INVALIDATE_TYPE=1    # Invalidate type 0:multi-node 1:multi-domain

# INVALIDATE_TYPE : INVALIDATE_TYPEが1であれば、Domain Gateway方式を使うとの意味です。

AGENT_SVC=SPFMAGENT|2 # multi-node sync. svc SPFMAGENT|2 = SPFMAGENT01, SPFMAGENT02

# AGENT_SVC : 2つのNodeがあれば、SPFMAGENT|2のように設定し、SPFMAGENT01、SPFMAGENT02を
# tpacallします。

# cache for PFM_SVC
CACHE_NAME=PFM_SVC
SIZE_MEM=65536      # the total cache memory size in kilo-bytes
SIZE_HASH=1024      # the number of hash key (MAX=65536)
SIZE_KEY=30         # the number of digits of the index column
SIZE_REC=2048       # the size of a single record in bytes
INV_TIMEOUT=10      # invalidation timeout in sec
.
.
.

```

Tmaxサーバーの設定(ノード2)

```
#####
#                                                                    #
#      Sample Configuration File for Tmax System                    #
#      =====                                                    #
#                                                                    #
#      Copyright(c) 2015 TmaxSoft Inc. All rights reserved         #
#                                                                    #
#####

*DOMAIN
tmax1          SHMKEY =78898, MINCLH=1, MAXCLH=3,
               TPORTNO=9030, BLOCKTIME=30

*NODE
tmaxh6         TMAXDIR = "/data5/tmax/tmax",
               APPDIR  = "/data5/tmax/tmax/appbin",
               PATHDIR = "/data5/tmax/tmax/path",
               TLOGDIR = "/data5/tmax/tmax/log/tlog",
               ULOGDIR = "/data5/tmax/tmax/log/ulog",
               SLOGDIR = "/data5/tmax/tmax/log/slog"

*SVRGROUP
NFRGRP02       NODENAME = "tmaxh6"

*SERVER
##### TCache Domain SERVER#####
TPFMAGENT      SVGNAME = NFRGRP02

#### TCache alias name ####
TCACHEFR01     SVGNAME = NFRGRP02, MIN=1, MAX=5, TARGET = TPFMAGENT,
               CLOPT = "-r -l -x SPFMAGENT:SPFMAGENT01 -o
               $(SVR)_$(CYEAR)$ (CMONTH)$ (CDAY).log -e
$(SVR)_$(CYEAR)$ (CMONTH)$ (CDAY).log"

*SERVICE
##### TCache Service #####
SPFMAGENT      SVRNAME = TPFMAGENT,   SVCTIME=30
SPFMAGENT01    SVRNAME = TCACHEFR01
SPFMAGENT02    SVRNAME = TGW01
# TCache ConfigのSPFMAGENT|2(SPFMAGENT01, SPFMAGENT02)はMatchされます。
# 内部のLocalについてはSPFMAGENT01がCallされ、RemoteについてはSPFMAGENT02がCallされて
# Invalidateされます。
# 技術的な詳細については、Tmaxガイドを参照してください。
```

```
##### TCache Gateway #####
*GATEWAY
TGW01          NODENAME = tmaxh6, GWTYPE=TMAXNONTX, PORTNO=9301,
                RGWADDR = "192.168.32.86", RGWPORTNO=9302, RESTART=Y,
                MAXRSTART=128, CLOPT="-i", CPC=8
```

TCacheの環境設定(ノード2)

```
# the configuration file of TCACHE

SHMKEY=0x78016      # the key of shared memory
IPCPERM=0777        # permission of the shared memory
SIZE_LOCAL=1024     # L1 cache size in kilo-bytes
LIB_PTHREAD=libpthread.so  # the pthread library file name
INVALIDATE_TYPE=1    # Invalidate type 0:multi-node 1:multi-domain
# INVALIDATE_TYPE : INVALIDATE_TYPEが1であれば、Domain Gateway方式を使うとの意味です。
AGENT_SVC=SPFMAGENT|2 # multi-node sync. svc SPFMAGENT|2 = SPFMAGENT01, SPFMAGENT02)
# AGENT_SVC : 2つのNodeがあれば、SPFMAGENT|2のように設定し、SPFMAGENT01、SPFMAGENT02を
tpacallします。

# cache for PFM_SVC
CACHE_NAME=PFM_SVC
SIZE_MEM=1024       # the total cache memory size in kilo-bytes
SIZE_HASH=1024      # the number of hash key (MAX=65536)
SIZE_KEY=30         # the number of digits of the index column
SIZE_REC=1024       # the size of a single record in bytes
INV_TIMEOUT=10      # invalidation timeout in sec
.
.
.
```

4.3. マルチ・ドメインの初期化(COUSIN)

ドメイン・ゲートウェイの設定が不可能な環境で、マルチ・ドメインの初期化を設定する場合です。まず、初期化を行いたいノードからの呼び出し先をTmax環境設定ファイルに設定します。その後、tpgetsvglistを読み込み、それぞれのノードのサーバーに対してtpacallsvgを使ってtpacallを行います。初期化プロセスは、[\[図 4.1\]](#)と同じです。技術的な詳細についてはTmaxガイドを参照してください。

Tmaxサーバーの設定(ノード1とノード2で同じ)

```
#####
# DOMAIN SECTION                                     #
#####
```

```

*DOMAIN

SHPDOM05      SHMKEY          = 97990,
               TPORTNO        = 8890,
               MAXUSER        = 3000,
               MINCLH         = 1,
               MAXCLH         = 2,
               RACPORT        = 9493,
               BLOCKTIME      = 1800,
               MAXSVR         = 2500,
               MAXSPR         = 6500,
               MAXSVC         = 20000,
               MAXSACALL      = 1024,
               MAXCACALL      = 1024,
               MAXSVG         = 100,
               MAXTOTALSVG    = 600,
               MAXGW          = 1024,
               MAXCPC         = 400,
               MAXCOUSIN      = 60,
               MAXCOUSINSVG   = 120,
               GWCHKINT       = 30,
               GWCONNECT_TIMEOUT = 30,
               NCLHCHKTIME    = 180,
               NLIVEINQ       = 30,
               IPCPERM        = 0664,
               MAXNODE        = 4

#####
# NODE SECTION                                     #
#####
*NODE
BCS01          HOSTNAME = "tmaxi4",
               TMAXDIR  = "/data2/tmax/tmax",
               APPDIR   = "/data2/tmax/tmax/appbin"

BCS02          HOSTNAME = "tmaxi7",
               TMAXDIR  = "/data2/tmax/package/tmax",
               APPDIR   = "/data2/tmax/tmax/appbin"

#####
# Server Group SECTION                             #
#####
*SVRGROUP

OPFMGRP11      NODENAME = "BCS01"
OPFMGRP21      NODENAME = "BCS02"

```

```

OPFMGRP31      NODENAME = "BCS01", COUSIN = "OPFMGRP32", LOAD = 0
OPFMGRP32      NODENAME = "BCS02", LOAD = 0

#####
# SERVER SECTION                                     #
#####
*SERVER
# TCache Multi Node ..
TPFMAGENT      SVGNAME = OPFMGRP31, MIN = 1, MAX = 1,
                CLOPT="-o $(SVR)_$(CYEAR)$ (CMONTH)$ (CDAY).log -e
$(SVR)_$(CYEAR)$ (CMONTH)$ (CDAY).log"

*SERVICE

SPFMAGENT      SVRNAME=TPFMAGENT

# ----- #
# Non Framework ...
# ----- #

```

TCacheの環境設定(ノード1)

```

# the configuration file of TCACHE

SHMKEY=0x70099      # the key of shared memory
IPCPERM=0777        # permission of the shared memory
SIZE_LOCAL=1024      # L1 cache size in kilo-bytes
LIB_PTHREAD=libpthread.so  # the pthread library file name
INVALIDATE_TYPE=0    # multi-node invalidate type.
# INVALIDATE_TYPE : INVALIDATE_TYPEが0であれば、Cousin方式を使用するとの意味です。
AGENT_SVC = SPFMAGENT # multi-node sync. svc
# AGENT_SVC : SPFMAGENTサーバーをtpacallします。

# cache for PFM_SVC
CACHE_NAME=PFM_SVC
SIZE_MEM=65536      # the total cache memory size in kilo-bytes
SIZE_HASH=1024      # the number of hash key (MAX=65536)
SIZE_KEY=30         # the number of digits of the index column
SIZE_REC=2048       # the size of a single record in bytes
INV_TIMEOUT=10      # invalidation timeout in sec
.
.
.

```

TCacheの環境設定(ノード2)

```
# the configuration file of TCache

SHMKEY=0x70999      # the key of shared memory
IPCPerm=0777        # permission of the shared memory
SIZE_LOCAL=1024      # L1 cache size in kilo-bytes
LIB_PTHREAD=libpthread.so  # the pthread library file name
INVALIDATE_TYPE=0    # multi-node invalidate type.
# INVALIDATE_TYPE : INVALIDATE_TYPEが0であれば、Cousin方式を使用するとの意味です。
AGENT_SVC = SPFMAGENT # multi-node sync. svc
# AGENT_SVC : SPFMAGENTサーバーをtpacallします。

# cache for PFM_SVC
CACHE_NAME=PFM_SVC
SIZE_MEM=65536       # the total cache memory size in kilo-bytes
SIZE_HASH=1024       # the number of hash key (MAX=65536)
SIZE_KEY=30          # the number of digits of the index column
SIZE_REC=2048        # the size of a single record in bytes
INV_TIMEOUT=10       # invalidation timeout in sec
.
.
.
```


付録 A. エラーコード

本章では、TCacheのエラーコードの内容と対応法について説明します。

- ERR_VER(-1)

説明	TCacheメモリーのバージョンとTCacheバイナリのバージョンが一致しない場合に発生するエラーです
対応方法	TCacheメモリーを再作成します

- ERR_UNAVAILABLE(-2)

説明	TCacheメモリーを作成、または削除する途中に発生するエラーです
対応方法	若干の時間が経過すると自動で解決されます。同じ状況が長時間維持される場合は、TCacheメモリーに問題が生じたのであり、メモリーの再作成が必要です

- ERR_NOACCESS(-3)

説明	「pfmtcacheadmin -t」を使用して、TCacheメモリーを使用不可にした状態で発生するエラーです
対応方法	<p>再度pfmtcacheadmin -tを使用してメモリーを使用可能な状態にします</p> <p>以下のコマンドを使用してTCacheメモリーの状態を確認することができます</p> <pre>\$ pfmtcacheadmin -s grep status</pre> <p>– status = AVAILABLE (正常)</p> <p>– status = NO ACCESS (使用不可)</p>

- ERR_NOTFOUND(-101)

説明	TCacheに当該データが存在しない場合に発生するエラーです
対応方法	キー値にデータが存在しません

- ERR_SHM(-102)

説明	共有メモリーをハンドリングする中で問題が発生した場合です
----	------------------------------

	(create、attach、remove、detachなど)
対応方法	TCacheメモリーを再作成します

● ERR_LOCK(-103)

説明	ロックの取得に失敗した場合(異常の原因で失敗)に発生するエラーです
対応方法	TCacheメモリーを再作成します

● ERR_WRONG_INPUT(-104)

説明	正しくないCacheNameを使用するか、SIZE_KEYよりもサイズが小さいデータを使用するときに発生するエラーです
対応方法	入力値を確認して修正します

● ERR_OUTOFMEM(-105)

説明	pfmTCachePutを実行するときにTCacheメモリー不足により発生するエラーです
対応方法	若干の時間が経過すると自動で解決されます。もし、特定のCacheNameで引き続き発生する場合には、キャッシュ設定のSIZE_MEM値を増やします

● ERR_NOCONFIG(-106)

説明	TCACHECONFの環境変数が設定されていないか、TCacheの環境設定ファイルに問題がある場合に発生するエラーです
対応方法	<ul style="list-style-type: none"> – 環境変数と環境設定ファイルの存在を確認します – ファイルの状態が正常なのかどうかを確認します – 読み込みの権限を確認します – ファイルがviで開かれるかを確認します

● ERR_PARSE_CONF(-107)

説明	TCacheの環境設定を読み込み途中、または処理途中に失敗した場合に発生するエラーです
対応方法	<ul style="list-style-type: none"> – 環境設定ファイルに必須項目が正しく設定されているかを確認します – キャッシュの数が128個以上であるかを確認します

● ERR_INV(-108)

説明	現在、このアイテムを初期化途中である場合に発生するエラーです
対応方法	TCacheの環境設定に設定したINV_TIMEOUT値だけの時間がたつと解除されます。 INV_TIMEOUTの単位は秒です

● ERR_SYS(-109)

説明	OS、またはTP関連のエラーが発生した場合です
対応方法	<ul style="list-style-type: none"> – システムの状態を確認します – pfmTCacheInvalidateを実行するときにERR_SYSが発生した場合は、TPの設定を確認します (SPFMAGENTが正しく設定されているかを確認)

● ERR_INTERNAL(-111)

説明	pfmTCacheReplaceを実行するときに発生する内部的なエラーです
対応方法	メモリーの再作成後も同じ状況が繰り返されるかどうかを確認した後、サポートを要請します

● ERR_BUSY(-112)

説明	ロックの取得に失敗した場合 (正常的な原因でLockWaitが発生) に発生するエラーです
対応方法	若干の時間が経過すると自動で解決されます。同じ状況が長時間維持される場合は、TCacheメモリーに問題が生じたのであり、メモリーの再作成が必要です

● ERR_OFFSET(-114)

説明	pfmTCacheReplaceを実行するときにoffset_data値が実際のデータよりも大きい場合に発生するエラーです
対応方法	APでデータ・サイズを確認します

● ERR_ARG(-115)

説明	TCache APIを実行するときにInvalid argumentを使用した場合に発生するエラーです
対応方法	APコードを確認します。cacheName、key、dataポインターなどがNULLかどうか、そしてsizeが正数なのかどうかを確認します

