

Tmax プログラミングガイド (ダイナミックライブラリ)

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 (ダイナミックライブラリ)

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. TDLの特徴	1
第2章 使用および管理	5
2.1. ディレクトリー構成	5
2.2. 環境設定	5
2.2.1. TDL環境設定	6
2.2.2. Tmax環境設定	9
2.3. システム管理ツール	10
2.3.1. tdlinit	11
2.3.2. tdlclean	12
2.3.3. tdlnm	13
2.3.4. tdlrm	14
2.3.5. tdlshm	14
2.3.6. tdlsync	17
2.3.7. tdlupdate	18
2.3.8. tdlseqno	19
2.3.9. tdltrace	20
第3章 API	23
3.1. 概要	23
3.2. tdlcall	24
3.3. tdlcall2	25
3.4. tdlcall2v	26
3.5. tdlcall2s	28
3.6. tdlcallva	29
3.7. tdlcallva2	31
3.8. tdlcreate	33
3.9. tdldestroy	35
3.10. tdlerror	37
3.11. tdlgetseqno	38
3.12. tdlstart	38
3.13. tdlend	38
3.14. tdlsuspend	39
3.15. tdlresume	39
3.16. tdlclose	40
3.17. tdlload	40
3.18. tdlload2	41
3.19. tdlinit	42

3.20.	tdldone	42
3.21.	tdlfind	43
3.22.	tdlfind2	43
3.23.	tdlstat	44
3.24.	tdlstat2	44
第4章	サンプル	47
4.1.	基本的な使用例	47
4.2.	C++インターフェースの使用例	48
4.3.	明示的なバージョン保護機能の使用例	50
4.4.	追加機能の例	51
4.4.1.	既存のMAXMODULESの変更	51
4.4.2.	tdlsyncとtdlinitを利用したMAXMODULESの変更	52
索引	53

図目次

[図 1.1]	TDL(Tmax Dynamic Library)の構造	1
[図 1.2]	明示的なバージョン整合性	2
[図 1.3]	暗示的なバージョン整合性	2
[図 1.4]	TDLドメインの構成	4

このガイドについて

対象読者

本書はTmax®(以下Tmax)のTDL(Tmax Dynamic Library)を使用してプログラムを開発する開発者を対象としています。TDLはオペレーティング・システムで提供するダイナミック・リンク・ライブラリー(Dynamic Link Library)を利用して、頻繁に変更される業務モジュールに対して中断のないアップデートを提供します。

前提知識

本書は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。実務での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

1. Tmaxシステム開発に関する基本的な内容については、『Tmax 運用ガイド』、または『Tmax アプリケーション開発ガイド』を参照してください。
 2. Tmax提供のコマンドとC APIについては、『Tmax リファレンスガイド』を参照してください。
-

本書の構成

本書は計4つの章で構成されています。

各章の主要な内容は以下のとおりです。

- 第1章: 紹介

TDL(Tmax Dynamic Library)の基本概念および特徴、構成について説明します。

- 第2章: 使用および管理

TDLを使用するための環境設定方法とTmaxの環境設定方法、システム管理ツールについて説明します。

- 第3章: API

TDLで提供する基本APIと使用方法について説明します。

- 第4章: サンプル

APIを使用したプログラムのサンプルを紹介します。

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
[]	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

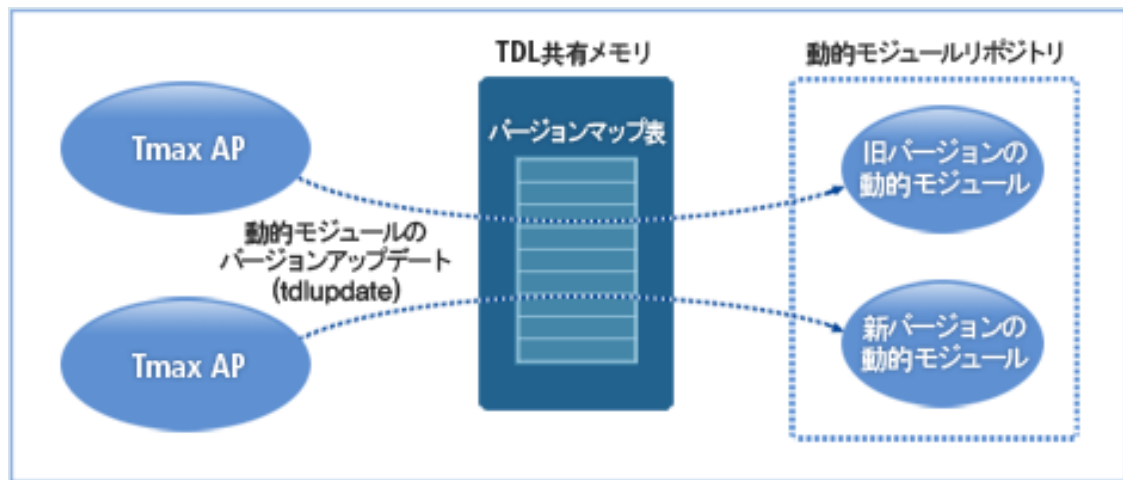
本章では、TDL(Tmax Dynamic Library)の概要および特徴について説明します。

1.1. 概要

TDL(Tmax Dynamic Library)はオペレーティング・システムで提供するダイナミック・リンク・ライブラリー(Dynamic Link Library)を利用し、頻繁に変更される業務モジュールに対して中断のないアップデートを提供します。

このような動的業務モジュールは共有ライブラリー形式でビルドされる必要があります。TDL管理ツールによりアップデート時にバージョンが与えられ、旧バージョンと新バージョンを同時に使用できる環境が構築されます。開発者はtdlcall()関数を利用してモジュールを呼び出します。基本的にtdlcall()は最新バージョンのモジュールを使用し、運用の中断なく旧バージョンから新バージョンに切り替わります。

【図 1.1】 TDL(Tmax Dynamic Library)の構造



1.2. TDLの特徴

TDLは以下のような特徴を持ちます。

- 共有メモリーを利用してバージョン情報を管理します。

TDLは各ライブラリーのバージョン情報を維持する必要があり、Tmaxサーバーやクライアント・プログラムでtdlcall()時にその情報を使用するため、共有メモリーにバージョン情報を保存します。したがって、Tmaxクライアントはローカルでのみ使用でき、必ず\$TDLDIR環境変数を指定する必要があります。

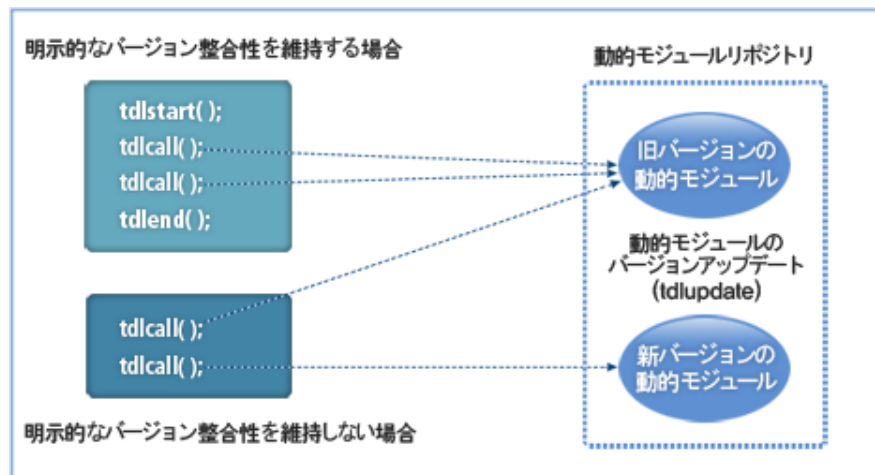
- バージョン整合性を保証します。

TDLは運用中にモジュールをアップデートするため、一時的に旧バージョンと新バージョンを同時に使用することがあります。同一トランザクション内の業務やバッチ業務などは同じバージョンのモジュールを使用する必要があります。そのため、TDLはバージョン整合性を保証します。バージョン整合性には、明示的なバージョン整合性(Explicit Version Consistency)と暗示的なバージョン整合性(Implicit Version Consistency)があります。

– 明示的なバージョン整合性

明示的なバージョン整合性(Explicit Version Consistency)は、`tdlstart()`、`tdlend()`、`tdlsuspend()`、`tdlresume()`などのAPIにより、同じバージョンが使用されるように明示的に指定することです。

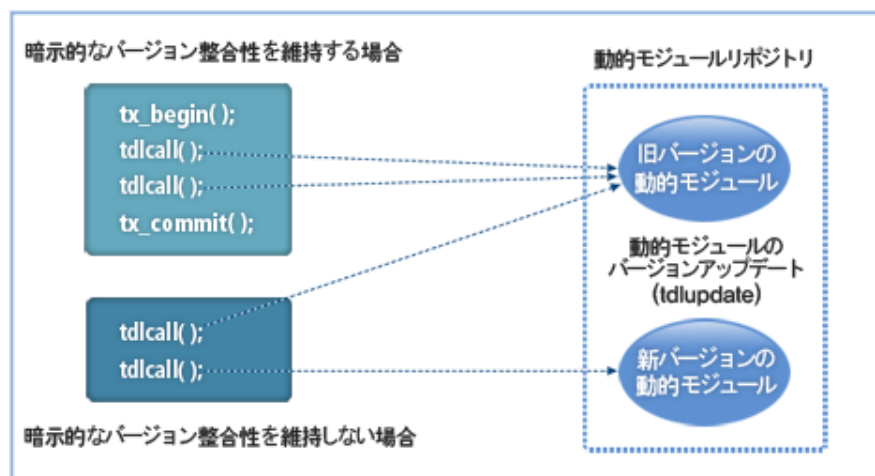
[図 1.2] 明示的なバージョン整合性



– 暗示的なバージョン整合性

暗示的なバージョン整合性(Implicit Version Consistency)は、同一トランザクション内の業務が自動的に同じバージョンを使用するように指定することです。この機能を使用するためにはDOMAINセクションまたはNODEセクションにTDLパラメータを設定する必要があります。

[図 1.3] 暗示的なバージョン整合性



- 再帰呼び出しはモジュールの整合性を保証します。

ユーザーがモジュールを再帰的に呼び出す場合は、関数の呼び出しが完了する前にモジュールがアップデートされることを防止する必要があります。そのため、モジュールは参照カウントを持っていて、該当するモジュールが呼び出されるたびに参照カウントは増加し、呼び出しが完了すると減少します。参照カウントが0以外の場合に、モジュールがアップデートされたり、メモリーから解除されたりすることを防止します。

- 多様な管理ツールをサポートします。

TDLの初期化およびアップデート、共有メモリーのバージョン情報と状態・統計情報の照会、旧バージョンのファイル削除などを行える管理ツールを提供します。管理ツールについての詳細は「[2.3. システム管理ツール](#)」を参照してください。

- エラーロギングおよびバックアップをサポートします。

TDLは管理ツールやAPIの使用時に発生したエラーや情報に関するロギングを残すことができます。エラーロギングを行うには、TDL環境ファイル(\$TMAXDIR/config/tcl.cfg)のLOGDIRパラメータを設定します。

また、共有メモリーに対するファイル・バックアップをサポートし、装備の障害や再起動の際にバックアップファイルからローディングすることができます。バックアップ・ファイルは、TDL環境ファイル(\$TMAXDIR/config/tcl.cfg)のBACKUPパラメータで設定します。

- マルチノード環境をサポートします。

TDLは基本的にTMAXマルチノード環境をサポートします。ただし、複数のノードのうち1つだけがマスター(master)ノードで、その他のノードはスレーブ(slave)ノードに指定する必要があります。初期化およびアップデートツールなどはマスターノードでのみ実行でき、照会ツールなどはスレーブノードでも実行できます。

マルチノード環境を使用するには、すべてのノードがTDL環境を構成する必要があります。TDL環境を使用しないノードがある場合は、TDL環境ファイル(\$TMAXDIR/config/tcl.cfg)のRACFILEパラメータを指定します。

マルチノードを使用する場合は、必ずすべてのノードでracdを起動する必要があります。racdについての説明は『Tmax 運用ガイド』を参照してください。

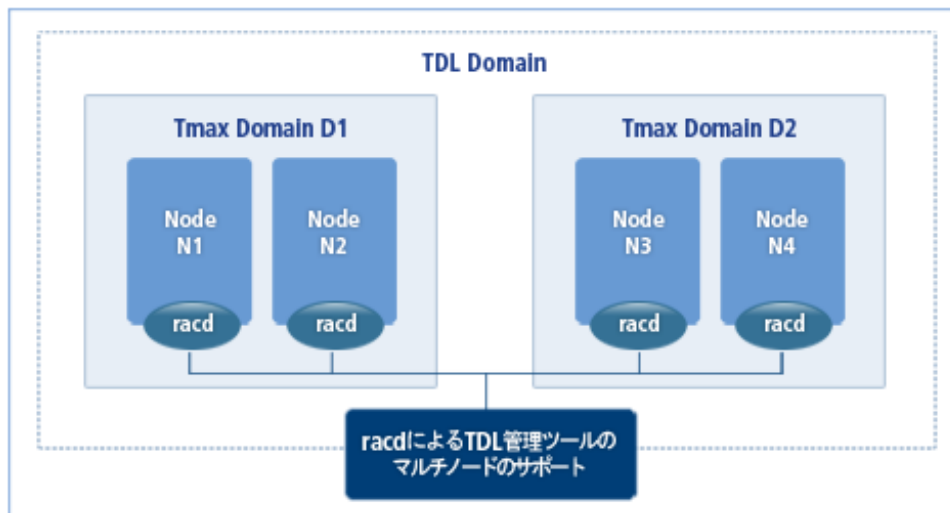
- TDLドメインをサポートします。

TDLでは複数のTMAXドメインを1つのTDLドメインに設定することができます。これは、TDLモジュールがドメインに関係なく共通して使用される環境で有効です。ただし、TDLドメインを使用するためにはracdを使用しなければならないため、TDL環境ファイル(\$TMAXDIR/config/tcl.cfg)のDOMAINIDとRACFILEパラメータを指定する必要があります。

特に、DOMAINIDはTDLドメインが異なる場合、同じく指定してはなりません。DOMAINID値を設定しない場合、デフォルトで0が自動設定されます。TDLドメインを使用する場合にも必ずすべてのノードでracdを起動する必要があります。

TDLドメインとライブラリー・バージョンの整合性は密接な関係があります。明示的なバージョン整合性はTDLドメイン内でのみ保証されます。暗示的なバージョン整合性はTDLドメインが異なる場合、自動的にドメイン別に整合性を維持します。

[図 1.4] TDLドメインの構成



- TDL APIを使用するアプリケーションにランタイム・トレース機能を提供します。

環境変数のTMAX_TRACEを設定してランタイム・トレース機能を利用することができます。詳細については『Tmax 運用ガイド』のTMAX_TRACE項目、あるいは『Tmax リファレンスガイド』の「2.30. tmaxtrace」を参照してください。

第2章 使用および管理

TDLを使用するためにはTDLの環境設定が必要です。また、暗示的なバージョン保護機能を使用するためには、Tmaxの環境設定が必要です。本章では、それぞれの環境設定方法とツールの使用方法について説明します。

2.1. ディレクトリー構成

TDLディレクトリーは、次の下位ディレクトリーで構成されます。

ディレクトリー	説明
config	TDL環境ファイル(tdl.cfg)用のディレクトリーです。RACFILEパラメータのデフォルト位置としても使用されます
mod	アップデートするライブラリー用のディレクトリーです。アップデートする共有ライブラリー(*.so または *.sl)をmodディレクトリーにコピーした後、tdlupdateを実行します
run	tdlupdateの後、バージョンが与えられたライブラリー用のディレクトリーです。ここにはシステム内部で使用するファイルも存在するため、任意に削除しないでください

参考

\$TDLDIRはTDLのルート・パスであり、特に指定しない場合は、\$TMAXDIR環境変数に設定されたパスを使用します。別のパスを指定したい場合は、\$TDLDIR環境変数を設定する必要があります。また、ローカル・クライアントでTDLを使用するには、\$TDLDIRを必ず指定します。

TDL 2.xバージョンでは、configではなく、\$TDLDIRにtdl.cfgが存在する必要があります。

2.2. 環境設定

TDLを使用するためには、TDL環境設定ファイルとTmaxのTDL関連項目の環境を設定する必要があります。

環境設定の基本文法は「パラメータ=値」であり、行内に空白があってははいけません。行の先頭文字が「#」の場合、その行はコメントとして処理します。

2.2.1. TDL環境設定

TDLの環境設定ファイルは、\$TMAXDIR/config/tdl.cfgです。以下は、TDL環境ファイルの例と設定項目についての説明です。

```
# shared memory version (1|2|3|4)
VERSION=2

# shared memory key
SHMKEY=0x90000

# shared memory and file creation permission
IPCPERM=0750

# number of dynamic loadable modules, rounded up to powers of 2
MAXMODULES=256

# shared memory backup
BACKUP=log/dlog/tdl.bak

# path of a command file for searching export functions
#COMMAND=tdlcmd
# running mode (single|master|slave)
MODE=single

# monitoring statistics
MONITOR=Y
# language of tdl application [C | COBOL | IBMCOBOL | PL1 | MIXED]
LANG=C

# path of a file including addresses for accessing remote tdl nodes
#RACFILE=tdl.rac
# tdl domain ID
DOMAINID=1

# multiple run, mod dir
DIRMAPPINGLIST=(mod:run),(mod2:run2),(mod2:run2)
# HASHFUNC=[ LOOKUP2 | LOOKUP3 | SUPERFASTHASH | TDLHASH | MURMURHASH ]
# HASHFUNC=MURMURHASH
```

項目	説明
VERSION	TDLの運用のためのバージョンを設定します。1、2、3、4のいずれかを選択します。 (デフォルト値:1) 詳細は「 VERSION項目 」の説明を参照してください
SHMKEY	TDLの運用に使用される共有メモリー・キー値を設定します

項目	説明
IPCPERM	runディレクトリーの動的モジュールおよび共有メモリーの権限を設定します
MAXMODULES	動的モジュールの最大数です。2^nに切り上げて使用します。モジュールを再配置することなく変更できます。詳細は「 第4章 サンプル 」を参照してください
BACKUP	共有メモリーのバックアップ・ファイルを指定します。tdlinit、tdlupdateの実行時に自動的に共有メモリーが指定したファイルにバックアップされます。相対パスである場合、\$TDLDIRまたは\$TMAXDIRからパスが始まります。パス情報の\$TDLDIRが優先されます
COMMAND	ライブラリーからエクスポートする関数を自動で抽出するためのスクリプト・ファイルの場所を指定します。相対パスである場合、\$TDLDIR/configまたは\$TMAXDIR/configからパスが始まります。パス情報の\$TDLDIRが優先されます
MODE	マルチノードまたはドメイン環境で、master、slave、singleのいずれかを指定します (デフォルト値: single)
MONITOR	TDL統計情報を収集するかどうかを設定します(デフォルト値: N) – Y : モジュール別の実行時間のAVG/MIN/MAX値、CPUのAVG/MIN/MAX値が収集され、「tdlshm -S」オプションにより照会できます
LANG	TDLアプリケーションが使用する言語を設定します。C、COBOL、IBMCOBOL、PL1、MIXEDのいずれかを選択します。 何も選択しない場合はCが設定されます。モジュールの拡張子はオペレーティング・システムによって決定されます
RACFILE	マルチノードまたはドメイン環境ですべてのノードのracdアドレスを保持するファイルを指定します。相対パスである場合、\$TDLDIR/configまたは\$TMAXDIR/configからパスが始まります。パス情報の\$TDLDIRが優先されます。 RACFILEの例については「 RACFILEの例 」を参照してください
DOMAINID	TDLドメインのIDです。ドメインごとに一意である必要があります(デフォルト値: 0)
DIRMAPPINGLIST	runディレクトリーを2つ以上使用する場合に設定します。 modディレクトリーとrunディレクトリーのペアを括弧とコロン(:)で区分し、最大10ペアまで設定できます。Tmax v5.0 SP1 Fix#2から適用されます
HASHFUNC	TDLの内部(各モジュールの索引設定)で使用するハッシュの種類を選択して設定できます。Tmax v5.0 SP2から適用されます

VERSION項目

TDLでは、現在4つのバージョンでシステムを運用することができます。このバージョン値は、TDL環境ファイル(tdl.cfg)のVERSIONパラメータにより設定できます。

- VERSION 1

VERSION=1に設定する場合で、ライブラリー当たり1つの関数を含みます。関数名もライブラリー名と一致する必要があります。ただし、関数名は全体で一意である必要があります。

- VERSION 2

VERSION=2に設定する場合で、ライブラリーが複数の関数を含めます。ただし、エクスポートする関数に対しては、以下のような方法をサポートします。ただし、関数名は全体で一意である必要があります。

- エクスポート・ファイルの指定方法

modディレクトリーにエクスポート・ファイル(ライブラリー名.exp)を移して、ユーザーが該当ライブラリーからエクスポートする関数を明示します。自動抽出機能を使用する場合、意図しなかった関数がエクスポートされるおそれがあるため、この方式の使用をお勧めします。

エクスポート・ファイルは1行当たり1つの関数を明示します。行の先頭文字が「#」である場合、その行はコメントとして処理します。

使用例は以下のとおりです。

```
# mylibrary
myfunction1
myfunction2
myfunction3
```

- エクスポート関数の自動抽出方法

エクスポートする関数を自動抽出します。エクスポート・ファイルを指定しない場合、関数は自動抽出されます。内部的にnmが使用され、ユーザーがnmスクリプトを再定義できます。スクリプトは必ず1行に作成します。行の先頭文字が「#」である場合、その行はコメントとして処理します。また、行の最後の文字が「\」である場合は、次の行に続けて処理します。「\$(LIB)」はライブラリー名を示すマクロ変数です。

使用例は以下のとおりです。

```
# for HP32
nm $(LIB) | awk -F'|' '{if ($3 == "extern" && $4 == "code  " )
{print $1}}' | grep -ve "^_"

# for HP64
nm $(LIB) | awk -F'|' '{if ($4 == "FUNC " && $7 == "      .text")
{print $8}}' | grep -ve "^_"
```

```
# for IBM32
nm $(LIB) | awk '{if ($2 == "D") {print $1}}'

# for IBM64
nm -X64 $(LIB) | awk '{if ($2 == "D") {print $1}}'

# for SUN
nm $(LIB) | awk -F"| " '{if ($4 == "FUNC ") {print $8}}' | grep -v "^_"

# for linux
nm $(LIB) | awk '{if ($2 == "T") {print $3}}' | grep -v "^_"

(*上記のスクリプトは例題で、実際のプラットフォームによって修正が必要です)
```

上記のスクリプトはTDL環境ファイル(tdl.cfg)のCOMMANDパラメータを使用するか、または管理ツールで-xオプションを使用して利用できます。またシステムで自動抽出される関数は、tdlnmによりあらかじめ確認できます。

- VERSION 3

VERSION=3に設定する場合で、ライブラリー名と関数名をキーとして使用します。したがって、バージョン1、2とは違って、関数名はライブラリー内でのみ一意であればいいです。ただし、バージョン3を使用する場合は、tdlcall2()、tdlcall2v()、tdlcall2s()を使用する必要があります。

- VERSION 4

VERSION=4に設定する場合で、ライブラリー名とクラス名の名前空間を使用します。C++で使用する方式であり、tdlcallではなく、別途のインターフェースを使用します。詳細は「[第4章 サンプル](#)」を参照してください。

RACFILEの例

以下のように設定されたファイルを「.rac」拡張子で保存し、RACFILE項目に該当ファイル名を設定します。

```
hostname1 192.168.1.1 3333
hostname2 192.168.1.2 3333
hostname3 192.168.1.3 3333
```

2.2.2. Tmax環境設定

暗示的なバージョン保護機能を使用するには、DOMAINセクションまたはNODEセクションにTDL項目を設定する必要があります。

```
*DOMAIN
tmax . . . ,
        TDL = Y

*NODE
tmax1 . . . ,
        TDL = Y
```

● DOMAINセクション

項目	説明
TDL	TDLの暗示的なバージョン保護機能を使用するかどうか(Y N)を設定します(デフォルト値: N)

● NODEセクション

項目	説明
TDL	TDLの暗示的なバージョン保護機能を使用するかどうか(Y N)を設定します。設定しない場合、DOMAINセクションのTDL値を引き継ぎます

2.3. システム管理ツール

TDLでは、以下のような管理ツールを提供します。管理ツールは次のパスに存在します。

```
$TMAXDIR/bin
```

コマンド	説明
tdlinit	TDL共有メモリーおよび動的モジュールを初期化します
tdlclean	runディレクトリーの旧バージョンのライブラリー・ファイルや不要なファイルを整理します
tdlnm	指定したライブラリーに対して自動エクスポートされる関数の一覧を照会します
tdlrm	TDLを使用しなくなった場合に、共有メモリーを完全に削除します
tdlshm	TDL共有メモリー情報を照会し、統計モニタリングとモジュールを有効にするかどうかを設定します
tdlsync	TDL共有メモリーとバックアップ・ファイルの同期化を実行します
tdlupdate	指定した動的モジュールをアップデートします
tdlseqno	動的モジュールのシーケンス番号を照会します
tdltrace	TDL環境設定情報と統計情報を照会します

2.3.1. tdlinit

TDL共有メモリーおよび動的モジュールを初期化するコマンドです。tdlinitコマンドは、インストール時に1回のみ実行し、Tmaxを起動前に実行する必要があります。マルチノード環境の場合、マスター・ノードでのみ実行可能です。

● 使用方法

```
$ tdlinit [-p TDLルート・ディレクトリーのパス] [-x エクスポート関数抽出スクリプト・ファイルのパス]
          [-f] [-b] [-B バックアップ・ファイルのパス] [-i] [-v | -V] [-h]
```

項目	説明
[-p TDLルート・ディレクトリーのパス]	TDLルート・ディレクトリーは、 <code>\${TDLDIR}</code> または <code>\${TMAXDIR}</code> を使用します。ルート・パスを別途指定して実行する場合、[-p]オプションを使用してパスを指定できます
[-x エクスポート関数抽出スクリプト・ファイルのパス]	エクスポート関数の抽出のためのスクリプト・ファイルのパスを指定します
[-f]	共有メモリーがすでに存在する場合、強制的に初期化します
[-b]	バックアップ・ファイルから共有メモリーをリカバリーします
[-B バックアップ・ファイルのパス]	指定されたバックアップ・ファイルから共有メモリーをリカバリーします
[-i]	バックアップのリカバリー後、runディレクトリーをチェックします。[-b]または[-B]オプションと一緒に使用します
[-v -V]	バージョン情報を出力します
[-h]	使用方法を出力します

● 例

- 以下は、TDL環境ファイル(tdl.cfg)を参照し、共有メモリーおよびモジュールを初期化する例です。

```
$ tdlinit
```

- 以下は、装備不具合の場合や再度開始する場合、バックアップ・ファイルから共有メモリーをリカバリーする例です。

```
$ tdlinit -b
```

● 参考

VERSION=4に設定した場合には、エクスポート関数を抽出するためのスクリプト・ファイルが必要です。

```

/* Example of exp file */
/* dlib.exp */

#! dlib.so
TmaxSoft::Airplain
Car

```

2.3.2. tdlclean

runディレクトリーの旧バージョンのライブラリー・ファイルや不要なファイルを整理するコマンドです。特に、[-m]オプションあるいは[-M]オプションを使用すると、共有メモリーで指定した動的モジュールが完全に削除されます。

• 使用方法

```

$ tdlclean [-p TDLルート・ディレクトリー・パス] [-m ライブラリー名] [-M 関数名] [-b]
           [-d yyyymmddhhmi] [-D "n hour|day" [-N 個数] [-v | -V] [-h]

```

項目	説明
[-p TDLルート・ディレクトリー・パス]	TDLルート・ディレクトリーは、 \$(TDLDIR) または \$(TMAXDIR) を使用します。 ルート・パスを別途指定して実行する場合、[-p]オプションを使用してパスを指定できます
[-m ライブラリー名]	TDL共有メモリーで指定したライブラリーとrunディレクトリーのファイルを削除します
[-M 関数名]	共有メモリーで指定した関数のみ削除し、runディレクトリーの関連ファイルは削除しません。ただし、VERSION=1と設定された場合、[-m]オプションと同一動作します
[-b]	TDL環境ファイル(tdl.cfg)にBACKUPパラメータが指定されていても、共有メモリー・ファイルのバックアップを実行しません。 [-m]または[-M]オプションと一緒に使用します
[-d yyyymmddhhmi]	指定時間(yyyymmddhhmi)以前の旧バージョンのライブラリー・ファイルをすべて削除します。 [-m]または[-M]オプションと一緒にには使用できません
[-D "n hour day"]	nは、時間(hour)または日(day)を設定します。指定時間または指定日以前の旧バージョンのライブラリー・ファイルをすべて削除します。 [-m]または[-M]オプションと一緒にには使用できません

項目	説明
[-N 個数]	指定した個数(number)分の旧バージョンのライブラリー・ファイルを残し、それ以外の旧バージョンのファイルをすべて削除します。 [-m]または[-M]オプションと一緒にには使用できません
[-v -V]	バージョン情報を出力します
[-h]	使用方法を出力します

- 例

- 以下は、旧バージョンのライブラリー・ファイルを削除する例です。

```
$ tdlclean
```

- 以下は、2009年2月1日00時00分以前の旧バージョンのライブラリー・ファイルを削除する例です。

```
$ tdlclean -d 200902010000
```

- 以下は、5日以前の旧バージョンのライブラリー・ファイルを削除する例です。

```
$ tdlclean -D "5 day"
```

- 以下は、mylibraryをTDL共有メモリーから完全に削除し、旧バージョンのファイルも削除する例です。

```
$ tdlclean -m mylibrary
```

2.3.3. tdlnm

VERSION=2以上で指定したライブラリーに対して自動エクスポートされる関数の一覧を照会するコマンドです。

- 使用方法

```
$ tdlnm [-p TDLルート・ディレクトリーのパス] [-x exportエクスポート関数抽出スクリプト・ファイルのパス]
        [-m ライブラリー名] [-v | -V] [-h]
```

項目	説明
[-p TDLルート・ディレクトリーのパス]	TDLルート・ディレクトリーは、 `\${TDLDIR}` または `\${TMAXDIR}` を使用します。 ルート・パスを別途指定して実行する場合、[-p]オプションを使用してパスを指定できます

項目	説明
[-x エクスポート関数抽出 スクリプト・ファイルのパス]	エクスポート関数の抽出のためのスクリプト・ファイルのパスを指定します
[-m ライブラリー名]	自動エクスポート関数一覧照会のためのライブラリー名を指定します
[-v -V]	バージョン情報を出力します
[-h]	使用方法を出力します

- 例

以下は、mylibraryファイルのエクスポート関数一覧を照会する例です。

```
$ tdlrm -m mylibrary
```

2.3.4. tdlrm

TDLを今後使用しない場合、共有メモリーを完全に削除するためのコマンドです。tdlrmコマンドを行う場合、**tdlcall()**を使用できません。

- 使用方法

```
$ tdlrm [-p TDLルート・ディレクトリーのパス] [-v | -V] [-h]
```

項目	説明
[-p TDLルート・ディレ クトリーのパス]	TDLルート・ディレクトリーは、 \${TDLDIR} または \${TMAXDIR} を使用します。ルート・パスを別途指定して実行する場合、[-p]オプションを使用してパスを指定できます
[-v -V]	バージョン情報を出力します
[-h]	使用方法を出力します

- 例

以下は、TDL共有メモリーを削除する例です。

```
$ tdlrm
```

2.3.5. tdlshm

TDL共有メモリー情報を照会し、統計モニタリングの有効化およびモジュールの有効化を設定するコマンドです。

● 使用方法

```
$ tdlshd [-p TDLルート・ディレクトリーのパス] [-r] [-S] [-n ノード名] [-m 関数名]
          [-M ライブラリー名] [-C] [-c start_index end_index] [-s e|d|r]
          [-u e|d] [-I 最小衝突回数] [-v | -V] [-h]
```

項目	説明
[-p TDLルート・ディレクトリーのパス]	TDLルート・ディレクトリーは、 <code>\$_TDLDIR</code> または <code>\$_TMAXDIR</code> を使用します。ルート・パスを別途指定して実行する場合、[-p]オプションを使用してパスを指定できます
[-r]	マルチノード環境でアップデート同期化中のモジュール情報を照会します
[-S]	動的モジュール統計情報を照会します。統計情報は、各モジュール別の実行時間AVG/MIN/MAXとCPU AVG/MIN/MAXが照会されます
[-n ノード名]	マルチノード環境で照会するノードを指定します
[-m 関数名]	照会する関数名を指定します
[-M ライブラリー名]	照会するライブラリー名を指定します
[-C]	全体モジュールに対してdlopen、dlsymを実行します
[-c start_index end_index]	指定した索引範囲のモジュールに対してdlopen、dlsymを実行します
[-s e d r]	動的モジュール統計収集を設定します – e : 有効化 – d: 無効化 – r : 初期化
[-u e d]	動的モジュールを設定します。 – e : 有効化 – d: 無効化 1度に1つのモジュールのみ設定可能なため、必ず[-m]または[-M]オプションと一緒に使用します。 – VERSION=1、VERSION=2の場合、モジュールは全体ライブラリーで一意であることを保証する必要があるため、[-m]オプションのみ指定します(関数名が全体的に一意である必要があります) – VERSION=3でライブラリーが異なるのであれば、関数名は重複しても構わないため、[-m]オプションと[-M]オプションは必ず入力します。[-m]オプションなしで[-M]オプションのみを使用してはいけません

項目	説明
[-l 最小衝突回数] (大文字i)	<p>モジュールごとにハッシュ衝突によるバケットを検索するための比較回数を出力します。</p> <p>0以上の値を指定でき、衝突回数が指定した回数以上のモジュールに対してのみ出力します。</p> <p>[-p]、[-m]、[-M]、[-r]オプションと一緒に使用できます</p>
[-v -V]	バージョン情報を出力します
[-h]	使用方法を出力します

- 例

– 以下は、tdlshmdの基本的な使用例です。

```
$ tdlshmd -S TDLDIR = /home/jeffry/tmax LOGDIR = /home/jeffry/tmax/log/dlog
- BACKUP = /home/jeffry/tmax/log/dlog/tdl.bak VERSION = 2, SHMKEY = 0x90000,
- IPCPERM = 0750 MAXMODULES = 256, CURMODULES = 3, Global SEQNO = 45e27d28,
- MONITOR = Y MODE = SINGLE, DOMAINID = 1 Index = 125, Funcname = myfunction1,
- Libname = myfunction1, Seqno = 45e27d28, Active = Y Count = 0 SVC: Avg = 0.000,

- MinTime = 0.000, Maxtime = 0.000 CPU: Avg = 0.000, MinTime = 0.000,
- Maxtime = 0.000 Index = 126, Funcname = myfunction2, Libname = myfunction2,
- Seqno = 45e27d28, Active = Y Count = 0 SVC: Avg = 0.000, MinTime = 0.000,
- Maxtime = 0.000 CPU: Avg = 0.000, MinTime = 0.000, Maxtime = 0.000 Index =
127,
- Funcname = myfunction3, Libname = myfunction3, Seqno = 45e27d28, Active = Y
- Count = 0 SVC: Avg = 0.000, MinTime = 0.000, Maxtime = 0.000 CPU: Avg = 0.000,

- MinTime = 0.000, Maxtime = 0.000 $ tdlshmd TDLDIR = /home/jeffry/tmax LOGDIR =
/home/jeffry/tmax/log/dlog
- BACKUP = /home/jeffry/tmax/log/dlog/tdl.bak VERSION = 2, SHMKEY = 0x90000,
- IPCPERM = 0750 MAXMODULES = 256, CURMODULES = 3, Global SEQNO = 45e27d28,
- MONITOR = Y MODE = SINGLE, DOMAINID = 1 Index = 125, Funcname = myfunction1,
- Libname = myfunction1, Seqno = 45e27d28, Active = Y Index = 126,
- Funcname = myfunction2, Libname = myfunction2, Seqno = 45e27d28, Active = Y,
- Index = 127, Funcname = myfunction3, Libname = myfunction3, Seqno = 45e27d28,

Active = Y
```

– 以下は、TDL統計情報を照会する例です。

```
$ tdlshmd -S TDLDIR = /home/jeffry/tmax LOGDIR = /home/jeffry/tmax/log/dlog
- BACKUP = /home/jeffry/tmax/log/dlog/tdl.bak VERSION = 2, SHMKEY = 0x90000,
- IPCPERM = 0750 MAXMODULES = 256, CURMODULES = 3, Global SEQNO = 45e27d28,
- MONITOR = Y MODE = SINGLE, DOMAINID = 1 Index = 125, Funcname = myfunction1,
```

```

- Libname = myfunction1, Seqno = 45e27d28, Active = Y Count = 0 SVC: Avg = 0.000,
- MinTime = 0.000, Maxtime = 0.000 CPU: Avg = 0.000, MinTime = 0.000,
- Maxtime = 0.000 Index = 126, Funcname = myfunction2, Libname = myfunction2,
- Seqno = 45e27d28, Active = Y Count = 0 SVC: Avg = 0.000, MinTime = 0.000,
- Maxtime = 0.000 CPU: Avg = 0.000, MinTime = 0.000, Maxtime = 0.000 Index =
127,
- Funcname = myfunction3, Libname = myfunction3, Seqno = 45e27d28, Active = Y
- Count = 0 SVC: Avg = 0.000, MinTime = 0.000, Maxtime = 0.000 CPU: Avg = 0.000,
- MinTime = 0.000, Maxtime = 0.000

```

- 以下は、[-s]または[-u]オプションの使用例です。

```

$ tdlshm -s r
$ tdlshm -s e -m myfunction
$ tdlshm -s d -m myfunction
$ tdlshm -u e -m myfunction
$ tdlshm -u d -m myfunction

```

- 以下は、[-C]または[-c]オプションの使用例です。

```

$ tdlshm -C
$ tdlshm -c 0 1024
$ tdlshm -c 1024

```

2.3.6. tdlsync

TDL共有メモリーとバックアップ・ファイルの同期化を実行するコマンドで、自動バックアップを使用しない場合に必要な管理ツールです。

- 使用方法

```
$ tdlsync [-p TDLルート・ディレクトリーのパス] [-B バックアップ・ファイルのパス] [-v | -V] [-h]
```

項目	説明
[-p TDLルート・ディレクトリーのパス]	TDLルート・ディレクトリーは\${TDLDIR}または\${TMAXDIR}を使用します。ルート・パスを別途指定して実行する場合、[-p]オプションを使用してパスを指定できます
[-B バックアップ・ファイルのパス]	バックアップ・ファイルのパスを指定します
[-v -V]	バージョン情報を出力します
[-h]	使用方法を出力します

- 例

以下は、TDL環境ファイル(tdl.cfg)のBACKUPパラメータに指定されたファイルで共有メモリーをバックアップする例です。

```
$ tdlsync
```

2.3.7. tdlupdate

指定した動的モジュールをアップデートするコマンドです。[-m]オプションでライブラリー名を必ず指定します。指定したライブラリーがすでに登録されている場合、指定したライブラリーをアップデートします。まだ登録されていない場合、新規追加します。マルチノード環境では、マスター／スレーブの両ノードで実行が可能です。

- 使用方法

```
$ tdlupdate [-p TDLルート・ディレクトリーのパス] [-x エクスポート関数抽出スクリプト・ファイルのパス] [-f]
            [-m ライブラリー名] [-b] [-c] [-l ファイル名] [-v | -V] [-h] [-r ディレクトリー・パス]
```

項目	説明
[-p TDLルート・ディレクトリーのパス]	TDLルート・ディレクトリーは、 \$(TDLDIR) または \$(TMAXDIR) を使用します。 ルート・パスを別途指定して実行する場合、[-p]オプションを使用してパスを指定できます
[-x エクスポート関数抽出スクリプト・ファイルのパス]	エクスポート関数の抽出のためのスクリプト・ファイルのパスを指定します
[-f]	VERSION=2で、別のライブラリーに関数名がすでに存在しているとしても強制的にアップデートします
[-m ライブラリー名]	アップデートするライブラリー名を指定します
[-b]	TDL環境ファイル(tdl.cfg)にBACKUPパラメータが指定されていても、共有メモリー・ファイルのバックアップを実行しません
[-c]	マルチノード環境でノード間のバージョン同期化を実行します
[-l ファイル名]	アップデート一覧をファイルで作成してアップデートを実行します。モジュールの区切り子はコンマ(,)、または<Enter>です
[-v -V]	バージョン情報を出力します
[-h]	使用方法を出力します
[-r ディレクトリー・パス]	tdlupdateを実行するリモート・ノードのTDLDIRを指定します。 [-p]オプションを使用する場合、2つ以上のTDLルート・ディレクトリーを持ちますが、それを指定するために使用します

- 例

- 以下は、<mylibrary.so>ファイルをアップデートする例です。

```
$ tdlupdate -m mylibrary
```

- 以下は、複数のモジュールをアップデートする例です。区切り子はコンマ(,)であり、空白なしで作成する必要があります。個数の制限はなく、マルチノードの場合は1024個に制限します。

```
$ tdlupdate -m mylibrary,mylibrary2,mylibrary3
```

- 以下は、アップデートするリストをファイルに作成して、アップデートを実行する例です。ファイル内に指定するモジュールの数に制限はなく、マルチノードの場合は1024個に制限します。

```
$ tdlupdate -l update.list
```

- update.listの作成方法です。以下のように、コンマ(,)や<Enter>キーを区切り子として使用して作成します。

```
mylibrary,mylibrary2
mylibrary3
```

- 以下は、マルチノード環境で共有メモリーの整合性に問題があるときに、同期化を実行する例です。

```
$ tdlupdate -c
```

- 参考

VERSION=4に設定した場合には、エクスポート関数を抽出するためのスクリプト・ファイルが必要です。

```
/* Example of exp file */
/* dlib.exp */

#! dlib.so
TmaxSoft::Airplain
Car
```

2.3.8. tdlseqno

特定のモジュールと関数のシーケンス番号を照会します。

- 使用方法

```
$ tdlseqno [-m module|-M library [-p tldir]] [-V] [-h]
```

項目	説明
[-p ディレクトリー・パス]	TDLルート・ディレクトリーは、 \${TDLDIR} または \${TMAXDIR} を使用します。 ルート・パスを別途指定して実行する場合、[-p]オプションを使ってパスを指定できます
[-M library]	当該ライブラリー名を指定します
[-m module]	当該関数名を指定します
[-V]	バージョン情報を出力します
[-h]	使用方法を出力します

- 例

- **\${TDLDIR}**または**\${TMAXDIR}**の当該モジュールと関数のシーケンス番号を照会する例です。

```
$ tdlseqno -m module1 -M library1
```

- 特定のディレクトリー下の当該モジュールと関数のシーケンス番号を照会する例です。

```
$ tdlseqno -m module1 -M library1 -p dirname
```

2.3.9. tdltrace

TDLの環境設定情報と統計情報を照会します。

- 使用方法

```
$ tdltrace [-p ディレクトリー・パス] [-P PID] [-V] [-h] [-c]
```

項目	説明
[-p ディレクトリー・パス]	TDLルート・ディレクトリーは、 \${TDLDIR} または \${TMAXDIR} を使用します。 ルート・パスを別途指定して実行する場合、[-p]オプションを使ってパスを指定できます
[-P PID]	特定のサーバー・プロセスの情報を照会します
[-V]	バージョン情報を出力します
[-h]	使用方法を出力します
[-c]	ゾンビ・プロセスを探して関連情報を削除します

- 例

```
$ tdltrace
```

```
$ tdltrace -P 20113
```

```
$ tdltrace -P 20113 -p dirname
```

```
$ tdltrace -c
```


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

TDL APIはTmaxサーバーやクライアント・ライブラリーに含まれており、tdlcall.hヘッダー・ファイルを含む(include)必要があります。Tmaxサーバーの場合はTCSおよびUCSタイプだけサポートし、クライアントの場合はマルチスレッド・ライブラリーはサポートしません。以下はTDL APIの一覧です。

API	説明
tdlcall	最新バージョンの動的モジュール関数を呼び出す関数です。TDL環境ファイル(tdl.cfg)にVERSION=1またはVERSION=2が設定されている場合に使用できます
tdlcall2	最新バージョンの動的モジュール関数を呼び出す関数です。TDL環境ファイル(tdl.cfg)にVERSION=3が設定されている場合に使用できます
tdlcall2v	最新バージョンの動的モジュール関数を呼び出す関数です。TDL環境ファイル(tdl.cfg)にVERSION=3が設定されている場合に使用できます
tdlcall2s	最新バージョンの動的モジュール関数を呼び出す関数です。TDL環境ファイル(tdl.cfg)にVERSION=3が設定されている場合に使用できます
tdlcallva	最新バージョンの動的モジュール関数を呼び出す関数です。TDL環境ファイル(tdl.cfg)にVERSION=1またはVERSION=2が設定されている場合に使用できます
tdlcallva2	最新バージョンの動的モジュール関数を呼び出す関数です。TDL環境ファイル(tdl.cfg)にVERSION=3が設定されている場合に使用できます
tdlcreate	最新バージョンの動的モジュールで、クラス・ファクトリーを使用してクラス・インスタンスを作成する関数です。TDL環境ファイル(tdl.cfg)にVERSION=4が設定されている場合に使用できます
tdldestroy	最新バージョンの動的モジュールで、クラス・ファクトリーを使用してクラス・インスタンスを破棄する関数です。TDL環境ファイル(tdl.cfg)にVERSION=4が設定されている場合に使用できます
tdlerror	tdlcall()に対するエラーが発生した場合、文字列形式に変換する関数です
tdlgetseqno	グローバル・シーケンス番号を取得する関数です
tdlstart	明示的なバージョン整合性(Explicit Version Consistency)の維持を開始する関数です
tdlend	明示的なバージョン整合性(Explicit Version Consistency)の維持を終了する関数です

API	説明
tdlsuspend	一時的にバージョン整合性の維持を停止する関数です
tdlresume	一時的に停止していたバージョン整合性の維持を再開する関数です
tdlclose	当該モジュールの参照カウントを0に初期化するか、モジュールを直接メモリーから解除します
tdlload	共有メモリーにモジュールをロードする関数です。TDL環境ファイル(tdl.cfg)にVERSION=1またはVERSION=2が設定されている場合に使用できます
tdlload2	共有メモリーにモジュールをロードする関数です。TDL環境ファイル(tdl.cfg)にVERSION=3が設定されている場合に使用できます
tdlinit	共有メモリーを初期設定する関数です
tdldone	共有メモリーを初期化する関数です
tdlfind	モジュールの索引を検索する関数です。TDL環境ファイル(tdl.cfg)にVERSION=1またはVERSION=2が設定されている場合に使用できます
tdlfind2	モジュールの索引を検索する関数です。TDL環境ファイル(tdl.cfg)にVERSION=3が設定されている場合に使用できます
tdlstat	TDLの統計情報を出力する関数です。TDL環境ファイル(tdl.cfg)にVERSION=1またはVERSION=2が設定されている場合に使用できます。 環境設定でMONITOR=Yに設定する必要があります
tdlstat2	TDLの統計情報を出力する関数です。TDL環境ファイル(tdl.cfg)にVERSION=3が設定されている場合に使用できます。 環境設定でMONITOR=Yに設定する必要があります

3.2. tdlcall

最新バージョンの動的モジュール関数を呼び出す関数です。動的モジュール関数は、必ず**long funcname(void *args)**形式のプロトタイプをもちます。TDL環境ファイル(tdl.cfg)のVERSIONが1または2に設定された場合に使用できます。動的モジュールは、初めてtdlcall()関数を使用される際にロードされます。特にアップデート(tdlupdate)されない場合は再使用され、性能低下を解消します。バージョンの整合性 (Version Consistency)を維持する状況では、整合性を反映したバージョンが使用されます。

- プロトタイプ

```
#include <tdlcall.h>
int tdlcall(char *funcname, void *args, long *urcode, int flags)
```

- パラメータ

パラメータ	説明
funcname	動的モジュールの関数名です(最大サイズ:TDL_FUNCNAME_SIZE - 1)
args	呼び出される動的モジュール関数のパラメータです
urcode	呼び出された動的モジュール関数の戻り値です
flags	TDLTRANに設定可能です。設定時は、必ずurcodeでグローバル・シーケンス番号を渡す必要があります。一方、使用しない場合は、TDL_NOFLAGSに設定します

- 戻り値

戻り値	説明
TDL_OK	正常に終了した場合です
TDL_OPEN_ERROR	dlopen()の使用時にエラーが発生した場合です
TDL_SYM_ERROR	dlsym()の使用時にエラーが発生した場合です
TDL_CLOSE_ERROR	dlclose()の使用時にエラーが発生した場合です
TDL_SYSTEM_ERROR	その他のシステム・コールを使用して、エラーが発生した場合です
TDL_INT_ERROR	ライブラリー内部でエラーが発生した場合です
TDL_ENOFUNC	該当モジュールあるいは関数を見付けることができない場合です
TDL_ENV_ERROR	環境変数の設定にエラーがある場合です
TDL_VER_ERROR	TDL共有メモリーのバージョンと環境ファイル(tdl.cfg)のバージョンが一致しない場合です
TDL_ARG_ERROR	設定された引数が正しくない場合です
TDL_ENOLIB	指定したライブラリーを見付けることができない場合です
TDL_TRAN_ERROR	バージョンの整合性処理中にエラーが発生した場合です
TDL_EINACTIVE	モジュールの使用が一時的に停止された状態です

3.3. tdlcall2

最新バージョンの動的モジュール関数を呼び出す関数です。ライブラリー名と関数名をキーとして使用します。動的モジュール関数は必ず**long funcname(void *args)**形式のプロトタイプをもちます。ライブラリー名と関数名をキーとして使用することを除けば、tdlcall()関数と同じ機能を提供します。動的モジュール関数は、必ず**long funcname(void *args)**形式のプロトタイプをもちます。TDL環境ファイル(tdl.cfg)にVERSION=3と設定されている場合に使用できます。

- プロトタイプ

```
#include <tdlcall.h>
int tdlcall2(char *libname, char *funcname, void *args, long *urcode, int flags)
```

● パラメータ

パラメータ	説明
libname	動的モジュールのライブラリー名です(最大サイズ:TDL_FUNCNAME_SIZE - 1)
funcname	動的モジュールの関数名です(最大サイズ:TDL_FUNCNAME_SIZE - 1)
args	呼び出される動的モジュール関数のパラメータです
urcode	呼び出された動的モジュール関数の戻り値です
flags	TDLTRANに設定可能です。設定時は、必ずurcodeでグローバル・シーケンス番号を渡す必要があります。一方、使用しない場合は、TDL_NOFLAGSに設定します

● 戻り値

戻り値	説明
TDL_OK	正常に終了した場合です
TDL_OPEN_ERROR	dlopen()の使用時にエラーが発生した場合です
TDL_SYM_ERROR	dlsym()の使用時にエラーが発生した場合です
TDL_CLOSE_ERROR	dlclose()の使用時にエラーが発生した場合です
TDL_SYSTEM_ERROR	その他のシステム・コールを使用して、エラーが発生した場合です
TDL_INT_ERROR	ライブラリー内部でエラーが発生した場合です
TDL_ENOFUNC	該当モジュールあるいは関数を見付けることができない場合です
TDL_ENV_ERROR	環境変数の設定にエラーがある場合です
TDL_VER_ERROR	TDL共有メモリーのバージョンと環境ファイル(tdl.cfg)のバージョンが一致しない場合です
TDL_ARG_ERROR	設定された引数が正しくない場合です
TDL_ENOLIB	指定したライブラリーを見付けることができない場合です
TDL_TRAN_ERROR	バージョンの整合性処理中にエラーが発生した場合です
TDL_EINACTIVE	モジュールの使用が一時的に停止された状態です

3.4. tdlcall2v

最新バージョンの動的モジュール関数を呼び出す関数です。ライブラリー名と関数名をキーとして使用します。動的モジュール関数は、必ず**long funcname(int argc, char *argv[])**形式のプロトタイプをもちます。

ライブラリー名と関数名をキーとして使用することを除けば、tdlcall()関数と同じ機能を提供します。TDL環境ファイル(tdl.cfg)にVERSION=3と設定されている場合に使用できます。

- プロトタイプ

```
#include <tdlcall.h>

int tdlcall2v(char *libname, char *funcname, int argc, char *argv[],
              long *urcode, int flags)
```

- パラメータ

パラメータ	説明
libname	動的モジュールのライブラリー名です(最大サイズ:TDL_FUNCNAME_SIZE - 1)
funcname	動的モジュールの関数名です(最大サイズ:TDL_FUNCNAME_SIZE - 1)
argc	呼び出される動的モジュール関数のパラメータ数です
argv	呼び出される動的モジュール関数のパラメータ・ベクトルです
urcode	呼び出された動的モジュール関数の戻り値です
flags	TDLTRANに設定可能です。設定時は、必ずurcodeでグローバル・シーケンス番号を渡す必要があります。一方、使用しない場合は、TDL_NOFLAGSに設定します

- 戻り値

戻り値	説明
TDL_OK	正常に終了した場合です
TDL_OPEN_ERROR	dlopen()の使用時にエラーが発生した場合です
TDL_SYM_ERROR	dlsym()の使用時にエラーが発生した場合です
TDL_CLOSE_ERROR	dlclose()の使用時にエラーが発生した場合です
TDL_SYSTEM_ERROR	その他のシステム・コールを使用して、エラーが発生した場合です
TDL_INT_ERROR	ライブラリー内部でエラーが発生した場合です
TDL_ENOFUNC	該当モジュールあるいは関数を見付けることができない場合です
TDL_ENV_ERROR	環境変数の設定にエラーがある場合です
TDL_VER_ERROR	TDL共有メモリーのバージョンと環境ファイル(tdl.cfg)のバージョンが一致しない場合です
TDL_ARG_ERROR	設定された引数が正しくない場合です
TDL_ENOLIB	指定したライブラリーを見付けることができない場合です
TDL_TRAN_ERROR	バージョンの整合性処理中にエラーが発生した場合です
TDL_EINACTIVE	モジュールの使用が一時的に停止された状態です

3.5. tdlcall2s

最新バージョンの動的モジュール関数を呼び出す関数です。ライブラリー名と関数名をキーとして使用します。動的モジュール関数は、必ず**long funcname(void *input, void *output)**形式のプロトタイプをもちます。ライブラリー名と関数名をキーとして使用することを除けば、tdlcall()関数と同じ機能を提供します。TDL環境ファイル(tdl.cfg)にVERSION=3と設定されている場合に使用できます。

- プロトタイプ

```
#include <tdlcall.h>
int tdlcall2s(char *libname, char *funcname, void *input, void *output,
              long *urcode, int flags)
```

- パラメータ

パラメータ	説明
libname	動的モジュールのライブラリー名です(最大サイズ:TDL_FUNCNAME_SIZE - 1)
funcname	動的モジュールの関数名です(最大サイズ:TDL_FUNCNAME_SIZE - 1)
input	呼び出される動的モジュール関数の入力バッファ・ポインターです
output	呼び出される動的モジュール関数の出力バッファ・ポインターです
urcode	呼び出された動的モジュール関数の戻り値です
flags	TDLTRANに設定可能です。設定時は、必ずurcodeでグローバル・シーケンス番号を渡す必要があります。一方、使用しない場合は、TDL_NOFLAGSに設定します

- 戻り値

戻り値	説明
TDL_OK	正常に終了した場合です
TDL_OPEN_ERROR	dlopen()の使用時にエラーが発生した場合です
TDL_SYM_ERROR	dlsym()の使用時にエラーが発生した場合です
TDL_CLOSE_ERROR	dlclose()の使用時にエラーが発生した場合です
TDL_SYSTEM_ERROR	その他のシステム・コールを使用して、エラーが発生した場合です
TDL_INT_ERROR	ライブラリー内部でエラーが発生した場合です
TDL_ENOFUNC	該当モジュールあるいは関数を見付けることができない場合です
TDL_ENV_ERROR	環境変数の設定にエラーがある場合です
TDL_VER_ERROR	TDL共有メモリーのバージョンと環境ファイル(tdl.cfg)のバージョンが一致しない場合です

戻り値	説明
TDL_ARG_ERROR	設定された引数が正しくない場合です
TDL_ENOLIB	指定したライブラリーを見付けることができない場合です
TDL_TRAN_ERROR	バージョンの整合性処理中にエラーが発生した場合です
TDL_EINACTIVE	モジュールの使用が一時的に停止された状態です

3.6. tdlcallva

最新バージョンの動的モジュール関数を呼び出す関数で、関数名をキーに使用します。動的モジュール関数のプロトタイプが固定されていない場合には、パラメータをそのまま渡して関数を呼び出します。ただし、渡されるパラメータはすべて(void *)ポインター型を保持しなければなりません。動的モジュール関数でも渡されるパラメータはすべて(void *)ポインター型で作成されている必要があります。

TDL環境ファイル(tdl.cfg)でVERSION=1またはVERSION=2に設定されている場合に使用できます。

● プロトタイプ

```
#include <tdlcall.h>
int tdlcallva(char *funcname, long urcode, int flags, int rettype, void *retval,
              int argc, ...);
```

● パラメータ

パラメータ	説明
funcname	動的モジュールの関数名です(最大サイズ: TDL_FUNCNAME_SIZE - 1)
urcode	他のtdlcall()系の関数と違い、グローバル・シーケンス番号を渡すときにのみ使用します。使用しない場合には0を入力します
flags	TDLTRANで設定可能です。設定時には、必ずurcodeでグローバル・シーケンス番号を渡す必要があります。使用しない場合には、TDL_NOFLAGSで設定します
rettype	呼び出される動的モジュール関数の返却型を定義します。設定可能な値は下の表を参照してください
retval	呼び出される動的モジュール関数の戻り値を保存するバッファのポインターを渡します
argc	呼び出される動的モジュール関数に渡される引数(argument)の数を入力します。現在、最大10個まで可能です
...	呼び出される動的モジュール関数に渡される実際のパラメータを可変引数で入力します。必ず(void *)のポインター型のパラメータを渡す必要があります

● 返却型

項目	説明
TDL_VA_CHAR	呼び出される動的モジュール関数の返却型がchar型です
TDL_VA_SHORT	呼び出される動的モジュール関数の返却型がshort型です
TDL_VA_INT	呼び出される動的モジュール関数の返却型がint型です
TDL_VA_LONG	呼び出される動的モジュール関数の返却型がlong型です
TDL_VA_FLOAT	呼び出される動的モジュール関数の返却型がfloat型です
TDL_VA_DOUBLE	呼び出される動的モジュール関数の返却型がdouble型です
TDL_VA_PVOID	呼び出される動的モジュール関数の返却型がvoid *型です
TDL_VA_VOID	呼び出される動的モジュール関数の返却型がvoid型です

- 戻り値

戻り値	説明
TDL_OK	正常に終了した場合は
TDL_OPEN_ERROR	dlopen()の使用時にエラーが発生した場合です
TDL_SYM_ERROR	dlsym()の使用時にエラーが発生した場合です
TDL_CLOSE_ERROR	dlclose()の使用時にエラーが発生した場合です
TDL_SYSTEM_ERROR	その他のシステム・コールを使用して、エラーが発生した場合です
TDL_INT_ERROR	ライブラリー内部でエラーが発生した場合です
TDL_ENOFUNC	該当モジュールあるいは関数を見付けることができない場合は
TDL_ENV_ERROR	環境変数の設定にエラーがある場合は
TDL_VER_ERROR	TDL共有メモリーのバージョンと環境ファイル(tdl.cfg)のバージョンが一致しない場合は
TDL_ARG_ERROR	設定された引数が正しくない場合は
TDL_ENOLIB	指定したライブラリーを見付けることができない場合は
TDL_TRAN_ERROR	バージョンの整合性処理中にエラーが発生した場合です
TDL_EINACTIVE	モジュールの使用が一時的に停止された状態です

- 例

- 動的モジュール関数

```
int myfunc(double *a, double *b, double *c) {
    double sum;
    return (int)((((double)(*a) + (double)(*b) + (double)(*c))/3));
}

char * myfunc2(int *type) {
```



```

char *msg;
switch (*type) {
    case 1: msg = "foo"; break;
    case 2: msg = "bar";break;
}
return msg;
}

```

– 呼び出しプログラム

```

#include <tdlcall.h>
int main(int argc, char *argv[]) {
    int n;
    long urcode;
    int retval;
    double a, b, c;
    int type;
    char *retmsg;

    urcode = tdlgetseqno();
    a = 2.5;
    b = 3.0;
    c = 3.5;
    if ((n = tdlcallva2("mylib001", "myfunc", urcode, TDLTRAN,
        TDL_VA_INT, &retval, 3, &a, &b, &c)) != TDL_OK) {
        error processing;
    }

    type = 1;
    if ((n = tdlcallva2("mylib001", "myfunc2", urcode, TDLTRAN,
        TDL_VA_PVOID, &retmsg, 1, &type)) != TDL_OK) {
        error processing;
    }
}

```

3.7. tdlcallva2

最新バージョンの動的モジュール関数を呼び出す関数で、ライブラリー名と関数名をキーに使用します。動的モジュール関数のプロトタイプが固定されていない場合には、パラメータをそのまま渡して関数を呼び出します。ただし、渡されるパラメータはすべて(void *)ポインター型を保持しなければいけません。動的モジュール関数でも渡されるパラメータはすべて(void *)ポインター型で作成されている必要があります。

ライブラリー名と関数名をキーに使用することを除いては、tdlcallva()関数と同様の機能を提供します。TDL環境ファイル(tdl.cfg)にVERSION=3と設定されている場合に使用できます。

- プロトタイプ

```
#include <tdlcall.h>
int tdlcallva2(char *libname, char *funcname, long urcode, int flags, int rettype,
void *retval, int argc, ...);
```

- パラメータ

パラメータ	説明
libname	動的モジュールのライブラリー名です(最大サイズ:TDL_FUNCNAME_SIZE - 1)
funcname	動的モジュールの関数名です(最大サイズ:TDL_FUNCNAME_SIZE - 1)
urcode	他のtdlcall()系の関数と違い、グローバル・シーケンス番号を渡すときにのみ使用します。使用しない場合には0を入力します
flags	TDLTRANIに設定可能です。設定時には、必ずurcodeでグローバル・シーケンス番号を渡す必要があります。使用しない場合は、TDL_NOFLAGSに設定します
rettype	呼び出される動的モジュール関数の返却型を定義します。設定可能な値は下表の返却型を参照してください
retval	呼び出される動的モジュール関数の戻り値を保存するバッファのポインタを渡します
argc	呼び出される動的モジュール関数に渡される引数(argument)の数を入力します。現在、最大10個まで可能です
...	呼び出される動的モジュール関数に渡される実際のパラメータを可変引数で入力します。必ず(void *)のポインタ型のパラメータを渡す必要があります

- 返却型

項目	説明
TDL_VA_CHAR	呼び出される動的モジュール関数の返却型がchar型です
TDL_VA_SHORT	呼び出される動的モジュール関数の返却型がshort型です
TDL_VA_INT	呼び出される動的モジュール関数の返却型がint型です
TDL_VA_LONG	呼び出される動的モジュール関数の返却型がlong型です
TDL_VA_FLOAT	呼び出される動的モジュール関数の返却型がfloat型です
TDL_VA_DOUBLE	呼び出される動的モジュール関数の返却型がdouble型です
TDL_VA_PVOID	呼び出される動的モジュール関数の返却型がvoid *型です
TDL_VA_VOID	呼び出される動的モジュール関数の返却型がvoid型です

- 戻り値

戻り値	説明
TDL_OK	正常に終了した場合です
TDL_OPEN_ERROR	dlopen()の使用時にエラーが発生した場合です
TDL_SYM_ERROR	dlsym()の使用時にエラーが発生した場合です
TDL_CLOSE_ERROR	dlclose()の使用時にエラーが発生した場合です
TDL_SYSTEM_ERROR	その他のシステム・コールを使用して、エラーが発生した場合です
TDL_INT_ERROR	ライブラリー内部でエラーが発生した場合です
TDL_ENOFUNC	該当モジュールあるいは関数を見付けることができない場合です
TDL_ENV_ERROR	環境変数の設定にエラーがある場合です
TDL_VER_ERROR	TDL共有メモリーのバージョンと環境ファイル(tdl.cfg)のバージョンが一致しない場合です
TDL_ARG_ERROR	設定された引数が正しくない場合です
TDL_ENOLIB	指定したライブラリーを見付けることができない場合です
TDL_TRAN_ERROR	バージョンの整合性処理中にエラーが発生した場合です
TDL_EINACTIVE	モジュールの使用が一時的に停止された状態です

3.8. tdlcreate

最新バージョンの動的モジュールで、クラス・ファクトリーを使用してクラス・インスタンスを作成する関数です。ライブラリー名と名前空間、クラス名をキーとして使用し、TDL環境ファイル(tdl.cfg)にVERSION=4と設定されている場合に使用できます。

● プロトタイプ

```
#include <tdlcall.h>
int tdlcreate(char *libname, char *namespace, char *classname, void *args, void
    *object, long *urcode, int flags)
```

● パラメータ

パラメータ	説明
libname	動的モジュールのライブラリー名です(最大サイズ:TDL_FUNCNAME_SIZE - 1)
namespace	名前空間を指定します(最大サイズ:TDL_FUNCNAME_SIZE - 1)
classname	インスタンスを作成するクラス名です(最大サイズ:TDL_FUNCNAME_SIZE - 1)
args	tdlcreate_cb()コールバック関数でユーザー定義データを渡すパラメータです

パラメータ	説明
object	tdlcreate_cb()コールバック関数で生成されたクラス・インスタンスの参照を保存する変数のポインターです。関数の呼び出しに成功したら、objectパラメータを適切なタイプに変換して使用します
urcode	tdlcreate_cb()コールバック関数で実行された結果の戻り値です
flags	TDLTRANに設定可能です。設定時には、必ずurcodeでグローバル・シーケンス番号を渡す必要があります。使用しない場合は、TDL_NOFLAGSに設定します

- 戻り値

戻り値	説明
TDL_OK	正常に終了した場合です
TDL_OPEN_ERROR	dlopen()の使用時にエラーが発生した場合です
TDL_SYM_ERROR	dlsym()の使用時にエラーが発生した場合です
TDL_CLOSE_ERROR	dlclose()の使用時にエラーが発生した場合です
TDL_SYSTEM_ERROR	その他のシステム・コールを使用して、エラーが発生した場合です
TDL_INT_ERROR	ライブラリー内部でエラーが発生した場合です
TDL_ENOFUNC	該当モジュールあるいは関数を見付けることができない場合です
TDL_ENV_ERROR	環境変数の設定にエラーがある場合です
TDL_VER_ERROR	TDL共有メモリーのバージョンと環境ファイル(tdl.cfg)のバージョンが一致しない場合です
TDL_ARG_ERROR	設定された引数が正しくない場合です
TDL_ENOLIB	指定したライブラリーを見付けることができない場合です
TDL_TRAN_ERROR	バージョンの整合性処理中にエラーが発生した場合です
TDL_EINACTIVE	モジュールの使用が一時的に停止された状態です

動的モジュールでクラス・ファクトリーを使用するには、**long tdlcreate_cb(char *namespace, char *classname, void *args, void *object)**形式のコールバック関数を実装しなければなりません。コールバック関数は、tdlcreate()で渡したパラメータ情報を利用して実際にクラス・インスタンスを作成し、作成されたインスタンスの参照をobjectに渡すように実装します。

この関数で作成したインスタンスの使用が完了すると、tdldestroy()関数によりインスタンスを破棄しなければなりません。tdlcreate()で作成したインスタンスをtdldestroy()で削除するまでは、tdlupdateが途中で呼び出され、動的モジュールが新しいバージョンに変更されても、現在作成されて使用されているインスタンスは既存のバージョンで動作します。このような場合、tdldestroy()を呼び出した後、再びtdlcreate()を呼び出すと、変更された動的モジュールのインスタンスが生成されます。

クラス・ファクトリーを使用するために、動的モジュールは、tdlcreate_cb()およびtdldestroy_cb()コールバック関数を実装する必要があります。

- コールバック関数のプロトタイプ

```
long tdlcreate_cb(char *namespace, char *classname, void *args, void *object)
```

- パラメータ

パラメータ	説明
namespace	tdlcreate()関数から渡された名前空間です
classname	tdlcreate()関数から渡された、インスタンス作成のクラス名です
args	ユーザー定義データです
object	コールバック関数で生成されたクラス・インスタンスの参照を保存する変数のポインターです

3.9. tdldestroy

最新バージョンの動的モジュールで、クラス・ファクトリーを使用して生成されたクラス・インスタンスを破棄する関数です。ライブラリー名と名前空間、クラス名をキーとして使用し、TDL環境ファイル(tdl.cfg)にVERSION=4と設定されている場合に使用可能です。

- プロトタイプ

```
#include <tdlcall.h>
int tdldestroy(char *libname, char *namespcae, char *classname, void *args, void
    *object, long *urcode, int flags)
```

- パラメータ

パラメータ	説明
libname	動的モジュールのライブラリー名です(最大サイズ: TDL_FUNCNAME_SIZE - 1)
namespace	名前空間を指定します(最大サイズ: TDL_FUNCNAME_SIZE - 1)
classname	インスタンスを作成するクラス名です(最大サイズ: TDL_FUNCNAME_SIZE - 1)
args	tdldestroy_cb()コールバック関数でユーザー定義データを渡すパラメータです
object	破棄するクラス・インスタンスの参照を渡します。tdldestroy_cb()コールバック関数で当該参照を破棄します。tdlcreate()関数で作成したオブジェクトのみを使用する必要があります

パラメータ	説明
urcode	tdldestroy_cb()コールバック関数で実行された結果の戻り値です
flags	TDLTRANに設定可能です。設定時には、必ずurcodeでグローバル・シーケンス番号を渡す必要があります。使用しない場合は、TDL_NOFLAGSに設定します

- 戻り値

戻り値	説明
TDL_OK	正常に終了した場合です
TDL_OPEN_ERROR	dlopen()の使用時にエラーが発生した場合です
TDL_SYM_ERROR	dlsym()の使用時にエラーが発生した場合です
TDL_CLOSE_ERROR	dlclose()の使用時にエラーが発生した場合です
TDL_SYSTEM_ERROR	その他のシステム・コールを使用して、エラーが発生した場合です
TDL_INT_ERROR	ライブラリー内部でエラーが発生した場合です
TDL_ENOFUNC	該当モジュールあるいは関数を見付けることができない場合です
TDL_ENV_ERROR	環境変数の設定にエラーがある場合です
TDL_VER_ERROR	TDL共有メモリーのバージョンと環境ファイル(tdl.cfg)のバージョンが一致しない場合です
TDL_ARG_ERROR	設定された引数が正しくない場合です
TDL_ENOLIB	指定したライブラリーを見付けることができない場合です
TDL_TRAN_ERROR	バージョンの整合性処理中にエラーが発生した場合です
TDL_EINACTIVE	モジュールの使用が一時的に停止された状態です

動的モジュールでクラス・ファクトリーを使用するには、**long tdldestroy_cb(char *namespace, char *classname, void *args, void *object)**形式のコールバック関数を実装しなければなりません。コールバック関数は、tdldestroy()で渡したパラメータ情報を利用してクラス・インスタスを破棄するように実装します。

tdlcreate()で作成したインスタスの使用が完了すると、この関数によりインスタスを破棄しなければなりません。tdlcreate()で作成したインスタスをtdldestroy()で削除するまでは、tdlupdateが途中で呼び出され、動的モジュールが新しいバージョンに変更されても、現在作成されて使用されているインスタスは既存のバージョンで動作します。このような場合、tdldestroy()を呼び出した後、再びtdlcreate()を呼び出すと、変更された動的モジュールのインスタスが生成されます。

- コールバック関数のプロトタイプ

```
long tdldestroy_cb(char *namespace, char *classname, void *args, void *object)
```

- パラメータ

パラメータ	説明
namespace	tdlcreate()関数から渡された名前空間です
classname	tdlcreate()関数から渡された、インスタンス作成のクラス名です
args	ユーザー定義データです
object	コールバック関数で破棄するクラス・インスタンスの参照です

3.10. tdlerror

tdlcall()に対するエラーが発生した場合、文字列形式に変換する関数です。直前のtdlcall()関数でエラーが発生した場合、エラーに対する詳細情報を文字列形式で渡します。戻り値で渡されるポインターは内部の静的変数に対するポインターであるため、次のtdlcall()を呼び出す場合に他の内容で設定され、捨てられることがあることに注意します。

したがって、この内容を保存あるいは修正したい場合は、他の変数にコピーをして使用します。

● プロトタイプ

```
#include <tdlcall.h>
char* tdlerror(int retval)
```

● パラメータ

パラメータ	説明
retval	直前のtdlcall()関数の戻り値です

● 戻り値

戻り値	説明
dlopen fail	tdlcall()の戻り値で、TDL_OPEN_ERRORを受信した場合です
dlsym fai	tdlcall()の戻り値で、TDL_SYM_ERRORを受信した場合です
dlclose fail	tdlcall()の戻り値で、TDL_CLOSE_ERRORを受信した場合です
etc system call error	tdlcall()の戻り値で、TDL_SYSTEM_ERRORを受信した場合です
TDL lib internal error	tdlcall()の戻り値で、TDL_INT_ERRORを受信した場合です
funcname not found	tdlcall()の戻り値で、TDL_ENOFUNCを受信した場合です
environment not found	tdlcall()の戻り値で、TDL_ENV_ERRORを受信した場合です
shared version mismatch	tdlcall()の戻り値で、TDL_VER_ERRORを受信した場合です
invalid arguments	tdlcall()の戻り値で、TDL_ARG_ERRORを受信した場合です
library not found	tdlcall()の戻り値で、TDL_ENOLIBを受信した場合です

戻り値	説明
transaction error	tdlcall()の戻り値で、TDL_TRAN_ERRORを受信した場合です
inactive funcation	tdlcall()の戻り値で、TDL_EINACTIVEを受信した場合です

3.11. tdlgetseqno

グローバル・シーケンス番号を取得する関数です。取得した番号を戻り値として返します。

- プロトタイプ

```
#include <tdlcall.h>
unsigned int tdlgetseqno(void)
```

- 戻り値

戻り値	説明
0より大きい値	関数の呼出しに成功した場合です
0	関数の呼出しに失敗した場合です。 「3.10. tdlerror」 で確認できます

3.12. tdlstart

明示的なバージョン整合性(Explicit Version Consistency)の維持を開始する関数です。

- プロトタイプ

```
#include <tdlcall.h>
int tdlstart(void)
```

- 戻り値

戻り値	説明
TDL_OK	関数の呼び出しに成功した場合です
TDL_TRAN_ERROR	関数の呼び出し前にtdlstart()が実行された場合です。詳細については 「3.2. tdlcall」 を参照してください

3.13. tdlend

明示的なバージョン整合性(Explicit Version Consistency)の維持を終了する関数です。

- プロトタイプ


```
#include <tdlcall.h>
int tdlend(void)
```

- 戻り値

戻り値	説明
TDL_OK	関数が正常に実行された場合です
TDL_TRAN_ERROR	関数の呼び出し前にtdlstart()が実行された場合です。詳細については 「3.2. tdlcall」 を参照してください

3.14. tdlsuspend

一時的にバージョン整合性の維持を停止する関数です。sd(suspend descriptor)を戻り値として返します。最大同時sd数は8です。

- プロトタイプ

```
#include <tdlcall.h>
int tdlsuspend(void)
```

- 戻り値

戻り値	説明
0、または0より大きい値	関数の呼び出しに成功した場合です
TDL_TRAN_ERROR	現在のバージョン整合性の維持モードでない場合です。詳細については 「3.2. tdlcall」 を参照してください

3.15. tdlresume

一時的に停止していたバージョン整合性の維持を再開する関数です。

- プロトタイプ

```
#include <tdlcall.h>
int tdlresume(int sd)
```

- プロトタイプ

パラメータ	説明
sd	tdlsuspend()関数から渡された記述子です

- 戻り値

戻り値	説明
TDL_OK	正常に終了した場合です
TDL_TRAN_ERROR	sdが有効な値でない場合です。詳細については「 3.2. tdlcall 」を参照してください

3.16. tdlclose

当該モジュールの参照カウントを0に初期化するか、モジュールを直接メモリーから解除します。

- プロトタイプ

```
#include <tdlcall.h>
int tdlclose(char *name, int flags)
```

- パラメータ

パラメータ	説明
name	当該モジュールの名前です
flags	0に設定した場合は、当該モジュールの参照カウントのみを0に初期化し、メモリーから解除しません。TDLCLOSE_HARDに設定する場合は、tdlcloseして、当該モジュールをメモリーから解除します

- 戻り値

戻り値	説明
TDL_OK	関数の呼び出しに成功した場合です
TDL_ENOLIB	該当する名前のモジュールが存在しない場合です

3.17. tdlload

tdlcall()を使って特定モジュールを初めて呼び出す場合、共有メモリーのハッシュ表(Hashtable)の検索およびライブラリーのメモリーへのロードによって若干の時間がかかります。これによって初回呼び出しがやや遅れてしまうことを防ぐために、ハッシュ表の検索とライブラリーのロードを、tdlcall()を使用する前に実行しておく、ローカル・キャッシュに当該モジュールの情報を保存する関数です。

TDLの環境ファイル(tdl.cfg)の設定が、VERSION=1またはVERSION=2の場合はtdlload()を使用します。また、VERSION=3の場合はtdlload2()を使用し、VERSION=4の場合はtdlload3()を使用します。

- プロトタイプ

```
#include <tdlcall.h>
int tdlload(char *funcname, int flags)
```

- パラメータ

パラメータ	説明
funcname	ロードを実行する関数の名前です
flags	現在使われていません

- 戻り値

戻り値	説明
0	関数を正常に呼び出した場合です
TDL_ENOFUNC	関数の引数としてlibnameやfuncnameがNULLで渡された場合や、ハッシュ表に存在しない値が渡された場合です
TDL_OPEN_ERROR	ハッシュ表に存在するダイナミック・ライブラリーをメモリーにロードできない場合です
TDL_SYSTEM_ERROR	ローカル・キャッシュを作成するためのシステム・リソースの割当に失敗した場合です

3.18. tdlload2

tdlloadと同様の関数です。詳細内容は「[3.17. tdlload](#)」を参照してください。

- プロトタイプ

```
#include <tdlcall.h>
int tdlload2(char *libname, char *funcname, int flags)
```

- パラメータ

パラメータ	説明
libname	ロードを実行するライブラリーの名前です
funcname	ロードを実行する関数の名前です
flags	現在使われていません

- 戻り値

tdlloadと同様の戻り値を持ちます。詳細内容は「[3.17. tdlload](#)」を参照してください。

3.19. tdlinit

共有メモリーを初期設定する関数です。

- プロトタイプ

```
#include <tdlcall.h>
int tdlinit(int flags)
```

- パラメータ

パラメータ	説明
flags	現在使われていません

- 戻り値

戻り値	説明
TDL_OK	関数の呼出しに成功した場合です
0より小さい値	関数の呼出しに失敗した場合です。 「3.10. tdlerror」 で確認できます

3.20. tdlldone

共有メモリーを初期化する関数です。

- プロトタイプ

```
#include <tdlcall.h>
int tdlldone(int flags)
```

- パラメータ

パラメータ	説明
flags	現在使われていません

- 戻り値

戻り値	説明
TDL_OK	関数の呼出しに成功した場合です
0より小さい値	関数の呼出しに失敗した場合です。 「3.10. tdlerror」 で確認できます

3.21. tdlfind

モジュールの索引を探す関数です。TDL環境ファイル(tdl.cfg)がVERSION=1またはVERSION=2に設定されている場合に使用します。

- プロトタイプ

```
#include <tdlcall.h>
int tdlfind(char *funcname, int flags)
```

- パラメータ

パラメータ	説明
funcname	検索する関数の名前です
flags	現在使われていません

- 戻り値

戻り値	説明
0、または0より大きい値	関数の呼出しに成功した場合です
0より小さい値	関数の呼出しに失敗した場合です。 「3.10. tdlerror」 で確認できます

3.22. tdlfind2

モジュールの索引を探す関数です。TDL環境ファイル(tdl.cfg)がVERSION=3に設定されている場合に使用します。

- プロトタイプ

```
#include <tdlcall.h>
int tdlfind2(char *libname, char *funcname, int flags)
```

- パラメータ

パラメータ	説明
libname	検索するライブラリーの名前です
funcname	検索する関数の名前です
flags	現在使われていません

- 戻り値

戻り値	説明
0、または0より大きい値	関数の呼出しに成功した場合です
0より小さい値	関数の呼出しに失敗した場合です。 「3.10. tdlerror」 で確認できます

3.23. tdlstat

TDLの統計情報を出力する関数です。

TDLの環境ファイル(tdl.cfg)がVERSION=1またはVERSION=2に設定されている場合に使用します。環境設定をMONITOR=Yに設定する必要があります。

- プロトタイプ

```
#include <tdlcall.h>
int tdlstat(char *funcname, struct timeval svc_time, struct timeval cpu_time)
```

- パラメータ

パラメータ	説明
funcname	統計情報を収集する関数の名前です
svc_time	統計情報のうちサービス・タイムが記録される変数です
cpu_time	統計情報のうちCPUタイムが記録される変数です

- 戻り値

戻り値	説明
TDL_OK	関数の呼出しに成功した場合です
0より小さい値	関数の呼出しに失敗した場合です。 「3.10. tdlerror」 で確認できます

3.24. tdlstat2

TDLの統計情報を出力する関数です。

TDL環境ファイル(tdl.cfg)がVERSION=3に設定されている場合に使用します。環境設定をMONITOR=Yに設定する必要があります。

- プロトタイプ

```
#include <tdlcall.h>
int tdlstat2(char *libname, char *funcname, struct timeval svc_time, struct
timeval cpu_usertime, struct timeval cpu_systemtime)
```

- パラメータ

パラメータ	説明
libname	統計情報を収集するライブラリーの名前です
funcname	統計情報を収集する関数の名前です
svc_time	統計情報のうちサービス・タイムが記録される変数です
cpu_usertime	統計情報のうちCPUのユーザー・タイムが記録される変数です
cpu_sysstime	統計情報のうちCPUのシステム・タイムが記録される変数です

- 戻り値

戻り値	説明
TDL_OK	関数の呼出しに成功した場合です
0より小さい値	関数の呼出しに失敗した場合は。 「3.10. tdllerror」 で確認できます

第4章 サンプル

本章では、APIを使用したプログラムのサンプルを紹介します。

4.1. 基本的な使用例

1. 以下の関数を動的モジュールにコンパイルして、my_function.slを作成します。

```
#include <tdlcall.h>

/* tdlcallで呼び出される関数 */
long my_function(void *args)
{
    long urcode;

    urcode = (long)time (NULL);
    strcpy((char*)args, "my_function");
    return urcode;
}
```

2. このモジュールの使用側では以下のとおり使用します。

```
/* TDL関連関数を使用するためには必ずインクルードが必要 */
#include <tdlcall.h>

int retval;
char data[256];
long urcode;

/* tdlcall関数の呼び出し : TDL Version 1 */
retval = tdlcall("my_function", (void*)data, &urcode, TDL_NOFLAGS);
if (retval != TDL_OK) {
    printf("tdlcall(%s) = %d, with err[%s]\n",
        "my_function", retval, tdlerror(retval));
} else {
    /* my_function関数で渡された値を出力 */
    printf("tdlcall(%s) = %d, urcode = %#08x, data = %s\n", "my_function",
        retval, urcode, data);
}
```

4.2. C++インターフェースの使用例

本節では、C++でTDLを使用するインターフェースについて説明します。

1. 以下の関数を動的モジュールにコンパイルして、dlib.soを作成します。

- dlib.h

```
/* dlib.h */

namespace TmaxSoft {
    class Airplain {
    private:
        int state;
    public:
        Airplain();
        virtual char *Start();
        virtual char *Stop();
        virtual int getState();
    };
}

class Car {
    private:
        int state;
    public:
        Car();
        virtual char *Start();
        virtual char *Stop();
        virtual int getState();
};
```

- dlib.cpp

```
/* dlib.cpp */

#include <string.h>
#include <usrinc/tdlcall.h>
#include "dlib.h"

long tdlcreate_cb(const char *snm, const char *cnm, void *args, void **obj) {

    if (0) {
    } else if (!strcmp(snm, "TmaxSoft") && !strcmp(cnm, "Airplain")) {
        *obj = (void *) new TmaxSoft::Airplain();
        if (*obj != NULL) return 1;
        else return -1;
    }
```

```

        } else if (!strcmp(snm, "") && !strcmp(cnm, "Car")) {
            *obj = (void *) new Car();
            if (*obj != NULL) return 1;
            else return -1;
        }
        return -1;
    }

long tdldestroy_cb(const char *snm, const char *cnm, void *args, void *obj) {

    if (0) {
    } else if (!strcmp(snm, "TmaxSoft") && !strcmp(cnm, "Airplain")) {
        delete (TmaxSoft::Airplain *)obj;
        return 1;
    } else if (!strcmp(snm, "") && !strcmp(cnm, "Car")) {
        delete (Car *)obj;
        return 1;
    }
    return -1;
}

TmaxSoft::Airplain::Airplain() { state = 0; }
char* TmaxSoft::Airplain::Start() {
state = 2; return (char *)"TmaxSoft::Airplain::Start "; }
char* TmaxSoft::Airplain::Stop() {
state = 1; return (char *)"TmaxSoft::Airplain::Stop "; }
int TmaxSoft::Airplain::getState() { return state; }

Car::Car() { state = 0; }
char* Car::Start() { state = 2; return (char *)"Car::Start "; }
char* Car::Stop() { state = 1; return (char *)"Car::Stop "; }
int Car::getState() { return state; }

```

2. このモジュールの使用側では以下のとおり使用します。

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <usrinc/tdlcall.h>
#include "dlib.h"

int
main(int argc, char *argv[])
{
    int ret;
    long urcode;

```

```

TmaxSoft::Airplain *obj;

if ((ret = tdlcreate("dlib", "TmaxSoft", "Airplain", NULL, (void **)&obj,

                    &urcode, 0)) < 0)

{
    printf("obj is NULL\n");
    printf("%s\n", tdlerror(ret));
} else {
    printf("obj %s\n", obj->Start());
    printf("obj %s\n", obj->Stop());
    printf("obj %d\n", obj->getState());
}

return 0;
}

```

3. クラスはnmでシンボルを正確に確認しにくいので、tdlinit、tdlupdateの場合は、必ずエクスポート・ファイルが必要です。エクスポート・ファイルの使用方法は以下のとおりです。

VERSION=1, 2, 3に設定されている場合には、関数名を1行に1つずつ記述しましたが、VERSION=4に設定されている場合には、クラス名を1行に1つずつ記述します。名前空間が適用されている場合は、「namespace::classname」の形式で1行に1つずつ記述します。

```

# libuser.so
foo
bar

# libuserclass.so
classCar
classBus
tmaxsoft::classCar
tmaxsoft::classBus
tiberio::myClass

```

4.3. 明示的なバージョン保護機能の使用例

```

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

TSVC1(TPSVCINFO *msg)
{

```

```

int i;
int n;
long urcode;
char *rcvbuf;
long rcvlen;
int ret;

printf("TSVC1 service is started!\n");

/* tdlstartを呼び出して明示的なバージョン保護を開始 */
tdlstart();
n = tdlcall("myfunction1", (void*)msg->data, &urcode, 0);
if (n != TDL_OK) {
    printf("tdlcall(myfunction1) = %d, with err[%s]\n", n, tdlerror(n));
} else {
    printf("tdlcall(myfunction1) = %d, urcode = %#08x\n", n, urcode);
}
sprintf((char *)msg->data, "[TSVC1]tdlcall(myfunction1)=%d", urcode);
if ((rcvbuf = (char *)tpalloc("STRING", NULL, 0)) == NULL) {
    printf("rcvbuf alloc failed !\n");
    tpreturn (TPFAIL, -1, NULL, 0, 0);
}

if(tpcall("TSVC2", rcvbuf, 0, &rcvbuf, &rcvlen, 0)==-1) {
    printf("Can't send request to service TSVC2[%d]\n", tperrno);
    tpfree((char *)rcvbuf);
    tpreturn (TPFAIL, -1, NULL, 0, 0);
}

/* tdlendを呼び出して明示的なバージョン保護を終了 */
tdlend();
strcat((char *)msg->data, rcvbuf);
msg->len = strlen((char *)msg->data);
tpreturn(TPSUCCESS,0,(char *)msg->data, msg->len, 0);
}

```

4.4. 追加機能の例

本節では、TDL環境設定のうち既存のMAXMODULESを変更する機能と、tdlsyncとtdlinitを用いたMAXMODULESの変更方法について説明します。

4.4.1. 既存のMAXMODULESの変更

以下は、既存のMAXMODULESを変更する方法です。モジュールの再配置が必要なため、時間がかかるデメリットがあります。

```
> tdlrm  
> tdlinit
```

4.4.2. tdlsyncとtdlinitを利用したMAXMODULESの変更

以下は、tdlsyncとtdlinitを用いたMAXMODULESの変更方法です。

変更されたMAXMODULESの値をベースに、新しい索引だけを計算して初期化します。モジュールの再配置プロセスがないため、[「4.4.1. 既存のMAXMODULESの変更」](#)の方法より時間とコストが削減されます。

```
> tdlsync -B original_maxmodules.bak  
- tdl.cfgのMAXMODULESの値を変更  
> tdlinit -B original_maxmodules.bak
```

索引

C

configディレクトリー, 5

M

modディレクトリー, 5

R

runディレクトリー, 5

T

TDL(Tmax Dynamic Library), 1

tdlcall, 24

tdlcall2, 25

tdlcall2s, 28

tdlcall2v, 26

tdlcallva, 29

tdlcallva2, 31

tdlclean, 12

tdlclose, 40

tdlcreate, 33

tdldestroy, 35

tdldone, 42

tdlend, 38

tdlerror, 37

tdlfind, 43

tdlfind2, 43

tdlgetseqno, 38

tdlinit, 11, 42

tdlload, 40

tdlload2, 41

tdlnm, 13

tdlresume, 39

tdlrm, 14

tdlseqno, 19

tdlshm, 14

tdlstart, 38

tdlstat, 44

tdlstat2, 44

tdlsuspend, 39

tdlsync, 17

tdltrace, 20

tdlupdate, 18

TDL環境設定

BACKUP, 7

COMMAND, 7

DIRMAPPINGLIST, 7

DOMAINID, 7

HASHFUNC, 7

IPCPERM, 7

LANG, 7

MAXMODULES, 7

MODE, 7

MONITOR, 7

RACFILE, 7

SHMKEY, 6

VERSION, 6

あ

暗示的なバージョン整合性(Implicit Consistency), 2	Version
---------------------------------------	---------

ま

明示的なバージョン整合性(Explicit Consistency), 2	Version
---------------------------------------	---------

