

Tmax プログラミングガイド (4GL)

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 (4GL)

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	xiii
第1章 PowerBuilderインターフェース	1
1.1. 概要	1
1.2. 関数	4
1.2.1. f_data	4
1.2.2. f_datadel	5
1.2.3. f_fdata	6
1.2.4. f_fdatadel	7
1.2.5. f_form	8
1.2.6. f_getdata	9
1.2.7. isblocked	10
1.2.8. pb_etpcall	11
1.2.9. pb_getunsold	13
1.2.10. pb_IsBlocked	14
1.2.11. pb_reset	15
1.2.12. pb_tmaxreadenv	16
1.2.13. pb_tpacall	18
1.2.14. pb_tpallocc	19
1.2.15. pb_tpbroadcast	21
1.2.16. pb_tpcall	22
1.2.17. pb_tpcallw	25
1.2.18. pb_tpcancel	27
1.2.19. pb_tpcommit	29
1.2.20. pb_tpconnect	30
1.2.21. pb_tpconv	32
1.2.22. pb_tpdicon	33
1.2.23. pb_tpend	34
1.2.24. pb_tpfcall	35
1.2.25. pb_tpfree	37
1.2.26. pb_tpget	38
1.2.27. pb_tpgetrply	39
1.2.28. pb_tpgetunsol	41
1.2.29. pb_tpput	42
1.2.30. pb_tprecv	43
1.2.31. pb_tpreset	46
1.2.32. pb_tpsend	47
1.2.33. pb_tpset_timeout	49
1.2.34. pb_tpsetunsol	51
1.2.35. pb_tpstart	52
1.2.36. pb_tptobackup	54

1.2.37.	pb_tx_begin	55
1.2.38.	pb_tx_commit	56
1.2.39.	pb_tx_rollback	57
1.2.40.	pb_tx_set_commit_return	58
1.2.41.	pb_tx_set_transaction_control	59
1.2.42.	pb_tx_set_transaction_timeout	60
1.2.43.	pb_uotmax_ver	61
1.2.44.	pb_tx_info	62
1.2.45.	tp_abort	63
1.2.46.	tp_acall	64
1.2.47.	tp_begin	65
1.2.48.	tp_broadcast	66
1.2.49.	tp_call	67
1.2.50.	tp_cancel	69
1.2.51.	tp_commit	70
1.2.52.	tp_connect	71
1.2.53.	tp_conv	73
1.2.54.	tp_discon	75
1.2.55.	tp_fcall	77
1.2.56.	tp_getrply	78
1.2.57.	tp_init	79
1.2.58.	tp_recv	80
1.2.59.	tp_send	83
1.2.60.	tp_subscribe	85
1.2.61.	tp_term	86
1.2.62.	tp_unsubscrib	86
1.2.63.	tpbegin	87
1.2.64.	tuxcall	88
1.2.65.	tuxedo_compat	90
1.2.66.	tuxreadenv	92
1.3.	サンプル・プログラム	93
1.3.1.	プログラムの構成	93
1.3.2.	プログラムの特徴	93
1.3.3.	共通プログラム	94
1.3.4.	クライアント・プログラム	96
1.3.5.	サーバー・プログラム	104
第2章	Visual Basicインターフェース	113
2.1.	概要	113
2.2.	関数	114
2.2.1.	FdlErrorMsg	114
2.2.2.	FilltpstartBuf	115
2.2.3.	GETCAR	116

2.2.4.	GETCAR_BA	117
2.2.5.	GETCHR	119
2.2.6.	GETDOUBLE	120
2.2.7.	GETFLOAT	121
2.2.8.	GETINT	122
2.2.9.	GETLONG	123
2.2.10.	GETSTR	123
2.2.11.	GETVAR	124
2.2.12.	PUTCAR	125
2.2.13.	PUTCAR_BA	126
2.2.14.	PUTCHR	128
2.2.15.	PUTDOUBLE	129
2.2.16.	PUTFLOAT	130
2.2.17.	PUTINT	131
2.2.18.	PUTLONG	132
2.2.19.	PUTSTR	133
2.2.20.	PUTVAR	134
2.2.21.	TmaxError	135
2.3.	サンプル・プログラム	136
2.3.1.	プログラムの構成	136
2.3.2.	プログラムの特徴	137
2.3.3.	共通プログラム	138
2.3.4.	クライアント・プログラム	139
2.3.5.	サーバー・プログラム	156
第3章	Delphiインターフェース	165
3.1.	概要	165
3.2.	サンプル・プログラム	165
3.2.1.	プログラムの構成	165
3.2.2.	プログラムの特徴	166
3.2.3.	共通プログラム	167
3.2.4.	クライアント・プログラム	169
3.2.5.	サーバー・プログラム	189
第4章	Visual Basic .netインターフェース(Unicode)	199
4.1.	概要	199
4.2.	関数	201
4.2.1.	ErrorMsg	201
4.2.2.	FdlErrorMsg	202
4.2.3.	FilltpstartBuf	203
4.2.4.	GETCAR	204
4.2.5.	GETCAR2	205
4.2.6.	GETCAR3	206
4.2.7.	GETCHR	207

4.2.8.	GETDOUBLE	208
4.2.9.	GETFLOAT	209
4.2.10.	GETINT	210
4.2.11.	GETLONG	211
4.2.12.	GETSHORT	212
4.2.13.	GETVAR	213
4.2.14.	PUTCAR	214
4.2.15.	PUTCAR2	215
4.2.16.	PUTCAR3	216
4.2.17.	PUTCHR	218
4.2.18.	PUTDOUBLE	219
4.2.19.	PUTFLOAT	220
4.2.20.	PUTINT	221
4.2.21.	PUTLONG	222
4.2.22.	PUTSHORT	223
4.2.23.	PUTVAR	224
4.3.	サンプル・プログラム	225
4.3.1.	プログラムの構成	226
4.3.2.	プログラムの特徴	227
4.3.3.	共通プログラム	227
4.3.4.	クライアント・プログラム	229
4.3.5.	サーバー・プログラム	257
第5章	Visual Basic .netインターフェース(一般)	265
5.1.	概要	265
5.2.	関数	267
5.2.1.	ErrorMsg	267
5.2.2.	FdLErrorMsg	268
5.2.3.	FilltpstartBuf	269
5.2.4.	GETCAR	270
5.2.5.	GETCAR2	271
5.2.6.	GETCAR3	272
5.2.7.	GETCHR	273
5.2.8.	GETDOUBLE	274
5.2.9.	GETFLOAT	275
5.2.10.	GETINT	276
5.2.11.	GETLONG	277
5.2.12.	GETSHORT	278
5.2.13.	GETVAR	279
5.2.14.	PUTCAR	280
5.2.15.	PUTCAR2	281
5.2.16.	PUTCAR3	282
5.2.17.	PUTCHR	284

5.2.18.	PUTDOUBLE	285
5.2.19.	PUTFLOAT	286
5.2.20.	PUTINT	287
5.2.21.	PUTLONG	288
5.2.22.	PUTSHORT	289
5.2.23.	PUTVAR	290
5.3.	サンプル・プログラム	291
5.3.1.	プログラムの構成	292
5.3.2.	プログラムの特徴	293
5.3.3.	共通プログラム	293
5.3.4.	クライアント・プログラム	295
5.3.5.	サーバー・プログラム	323
第6章	C#.netインターフェース	331
6.1.	概要	331
6.2.	サンプル・プログラム	331
6.2.1.	プログラムの構成	332
6.2.2.	プログラムの特徴	332
6.2.3.	共通プログラム	333
6.2.4.	クライアント・プログラム	334
6.2.5.	サーバー・プログラム	357
第7章	ASPインターフェース	365
7.1.	概要	365
7.2.	Atmiインターフェース・メソッド	369
7.2.1.	OnGettperrno	369
7.2.2.	OnGettpurcode	370
7.2.3.	OnTpacall	370
7.2.4.	OnTpalloc	371
7.2.5.	OnTpcall	371
7.2.6.	OnTpcancel	372
7.2.7.	OnTpconnect	373
7.2.8.	OnTpdiscon	374
7.2.9.	OnTpend	374
7.2.10.	OnTpfree	375
7.2.11.	OnTpget	375
7.2.12.	OnTpgetrply	376
7.2.13.	OnTpput	377
7.2.14.	OnTprealloc	378
7.2.15.	OnTprecv	378
7.2.16.	OnTpseend	379
7.2.17.	OnTpstart	380
7.2.18.	OnTpypes	381
7.3.	その他のインターフェース・メソッド	382

7.3.1.	OnTmax_chk_conn	382
7.3.2.	OnTmaxreadenv	382
7.3.3.	OnTp_sleep	383
7.3.4.	OnTp_usleep	383
7.3.5.	OnTpdeq	384
7.3.6.	OnTpenq	385
7.3.7.	OnTperrordetail	386
7.3.8.	OnTpextsvcinfo	386
7.3.9.	OnTpextsvcname	387
7.3.10.	OnTpgetenv	387
7.3.11.	OnTpputenv	388
7.3.12.	OnTpqstat	389
7.3.13.	OnTpqsvcstat	389
7.3.14.	OnTpissue	390
7.3.15.	OnTpreset	391
7.3.16.	OnTpset_timeout	391
7.3.17.	OnTpsubqname	392
7.3.18.	OnTptobackup	393
7.4.	FDLインターフェース・メソッド	393
7.4.1.	OnFballoc	393
7.4.2.	OnFbcalcsz	394
7.4.3.	OnFbdelall	394
7.4.4.	OnFbdelete	395
7.4.5.	OnFbfldcount	396
7.4.6.	OnFbfree	396
7.4.7.	OnFbget	397
7.4.8.	OnFbget_fbsz	397
7.4.9.	OnFbget_fldkey	398
7.4.10.	OnFbget_fldname	398
7.4.11.	OnFbget_fldtype	399
7.4.12.	OnFbget_strfldtype	400
7.4.13.	OnFbget_unused	400
7.4.14.	OnFbget_used	401
7.4.15.	OnFbgetc	401
7.4.16.	OnFbgetf	403
7.4.17.	OnFbgetnth	403
7.4.18.	OnFbgetntht	404
7.4.19.	OnFbgetval	405
7.4.20.	OnFbgetvali	406
7.4.21.	OnFbgetvals	407
7.4.22.	OnFbgetvalt	407
7.4.23.	OnFbinit	408
7.4.24.	OnFbinsert	409

7.4.25.	OnFbisbuf	410
7.4.26.	OnFbispres	410
7.4.27.	OnFbkeynm_unload	411
7.4.28.	OnFbkeyoccur	411
7.4.29.	OnFbmake_fldkey	412
7.4.30.	OnFbnmkey_unload	413
7.4.31.	OnFbput	413
7.4.32.	OnFbputt	414
7.4.33.	OnFbrealloc	415
7.4.34.	OnFbsterror	415
7.4.35.	OnFbtypecv	416
7.4.36.	OnFbupdate	417
7.4.37.	OnGetfberrno	417
7.4.38.	OnGetfberror	418
7.5.	Txインターフェース・メソッド	418
7.5.1.	OnTx_begin	418
7.5.2.	OnTx_commit	419
7.5.3.	OnTx_rollback	419
7.5.4.	OnTx_set_transaction_timeout	419
7.6.	サンプル・プログラム	420
7.6.1.	プログラムの構成	420
7.6.2.	プログラムの特徴	420
7.6.3.	クライアント・プログラム	421
第8章	PHPインターフェース	437
8.1.	使用方法	437
8.1.1.	開発環境	437
8.1.2.	拡張モジュールの作成手順	437
8.1.3.	拡張モジュールの使用方法	438
8.1.4.	サンプル・プログラム	440
索引	443

このガイドについて

対象読者

本書は、Tmax[®](以下、Tmax)を使用時に4GL開発ツールを使用してクライアント・プログラムを開発するユーザーを対象としています。

前提知識

本書は、Tmaxシステムに関する全般的な内容と、Tmaxシステムが提供している各種機能および特性を習得するためのガイドです。

本書を理解するには、事前に下記の内容を把握しておく必要があります。

- ミドルウェア(Middleware)およびUNIXシステムに関する知識
- Tmaxの基本概念に関する知識
- Java、Cプログラミングに関する知識
- 4GL言語に関する知識

制限事項

本書では、4GLに関連した関数の内容のみ記述しています。実務における詳細な使用方法および管理、運用に関する内容については各製品のガイドを参照してください。

参考

Tmaxシステム開発に関する基本的な内容は、『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。Tmaxが提供しているコマンドとC APIに関する説明については、『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7つの章で構成されています。

各章の主な内容は下記のとおりです。

- 第1章: Power Builderインターフェース

Power Builerインターフェースで使用する関数と例について記述します。

- 第2章: Visual Basicインターフェース

Visual Basicインターフェースで使用する関数と例について記述します。

- 第3章: Delphiインターフェース

Delphiインターフェース・モジュールと例について記述します。

- 第4章: Visual Basic .netインターフェース(Unicode)

Unicodeおよび64ビットをサポートするVisual Basic .netインターフェースで使用する関数と例について記述します。

- 第5章: Visual Basic .netインターフェース(一般)

一般のVisual Basic .netインターフェースで使用する関数と例について記述します。

- 第6章: C# .netインターフェース

C# .netインターフェース・モジュールと例について記述します。

- 第7章: ASPインターフェース

ASPインターフェースで使用する関数と例について記述します。

- 第8章: PHPインターフェース

PHPでTmaxクライアントの機能が使用できる拡張モジュールの使用方法和例について記述します。

表記上の規則

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

関連ガイド

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
Tmax FDLリファレンスガイド	Tmax FDL関数の定義とサンプル・プログラムを利用して、FDLが提供する機能を活用する方法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 PowerBuilderインターフェース

本章では、PowerBuilderのインターフェースで使用する関数と例について説明します。

1.1. 概要

PowerBuilderには、ポインターという概念がなくすべてのデータを文字列として認識しているため、Tmaxクライアント・ライブラリーを直接使用することができません。そのため、各データのタイプを維持し、PowerBuilderで提供している各種PowerObjectを有効に使用するためのUser Object(tmax.pbd)を提供しています。

PowerBuilderには、データ情報を維持すると同時に多様なデータ・プレゼンテーション機能を有するDataWindowという非常に便利なPowerObjectがあります。PowerBuilderがマーケットにおける最も強力なアプリケーション開発ソフトウェア・パッケージに挙げられる理由の1つはDataWindow機能が含まれているためです。

しかし、既存のクライアント・ライブラリーで提供するデータ・フォーマットをそのままDataWindowに適用することができないため、開発に多くの時間と手間がかかりました。このような問題を解決するために、tmax.pbdでは結果データ・フォーマットをPowerBuilderのDataWindowで直接使用できるような形式に変更しました。これにより、開発者は別途の作業を行わずにデータを操作することができます。

下記は、tmax.pbdで提供する関数の一覧です。

関数	説明
f_data	データ文字列を作成します
f_datadel	データ文字列を作成します
f_fdata	データ文字列を作成します
f_fdatadel	データ文字列を作成します
f_form	Form文字列を作成します
f_getdata	Formと同様な構成で作成されたアウトプット内容の中で、特定した位置の文字列に返します
isblocked	ブロッキング状態をチェックします
pb_etpcall	イメージ・データを処理します
pb_getunsold	非要求メッセージを受信します
pb_IsBlocked	ブロッキング状態をチェックします
pb_reset	現在の接続状態を即解除します
pb_tmaxreadenv	ファイルに保存されている接続するシステムの情報を読み込み、環境変数に新しい値を設定します

関数	説明
pb_tpacall	非同期サービス呼び出しを実行します
pb_tpalloc	itypeで指定されたバッファを割り当て、アドレスを返します
pb_tpbroadcast	指定したクライアントにメッセージを渡します
pb_tpcall	同期処理関数です
pb_tpcallw	DataWindowを使用して同期型でTmaxサービスを呼び出します
pb_tpcancel	pb_tpacall()が返した記述子を取り消します
pb_tpcommit	トランザクション作業をコミットするために使用されます
pb_tpconnect	会話型サービスと通信を接続します
pb_tpconv	cdによって指定された会話型サービスにメッセージを送受信します
pb_tpdison	会話の接続を強制切断し、TPEV_DISCONIMMイベントを発生させます
pb_tpend	Tmaxシステムとの接続を解除します
pb_tpfcall	ファイルから入力データが取得可能で、結果データをファイルに保存します
pb_tpfree	pb_tpalloc()で取得したバッファを解除します
pb_tpget	1番目のパラメータで指定したアドレスに該当するデータを、2番目のパラメータで指定したバッファに3番目パラメータで指定したサイズだけ読み込む関数です
pb_tpgetrply	pb_tpacall()を使用して送った要求に対する応答を取得します
pb_tpgetunsol	非要求受信メッセージを処理します
pb_tpput	1番目のパラメータで指定したアドレスに、2番目のパラメータで指定したデータを3番目のパラメータで指定したサイズだけ保存します
pb_tprecv	会話型通信でデータを受信するために使用されます
pb_tpreset	現在の接続状態を即解除します
pb_tpsend	会話型通信で相手にデータを送信するために使用されます
pb_tpset_timeout	サーバーに設定されているサービス制限時間のブロッキング・タイムアウト時間を変更するために使用されます
pb_tpsetunsol	非要求受信メッセージを受信するように設定します
pb_tpstart	Tmaxとの接続を確立します
pb_tptobackup	クライアントがTmaxシステムに接続する際に、最初からバックアップ・システムに接続する場合に使用します
pb_tx_begin	グローバル・トランザクションを開始します
pb_tx_commit	トランザクション作業をコミットするために使用されます
pb_tx_rollback	トランザクション作業をロールバックするために使用されます
pb_tx_set_commit_return	when_return値を利用してグローバル・トランザクションのコミット時点を指定します

関数	説明
pb_tx_set_transaction_control	コントロールに指定された値を利用して、pb_tx_commit()またはpb_tx_rollback()が返される前に新しいトランザクションの開始可否を指定します
pb_tx_set_transaction_timeout	トランザクションのタイムアウトを設定します
pb_uotmax_ver	tmax.pbddのバージョンを表示します
pb_tx_info	グローバル・トランザクションの情報を表示します
tp_abort	トランザクションの作業を中止(abort)します
tp_acall	非同期サービス呼び出しを実行します
tp_begin	グローバル・トランザクションを開始します
tp_broadcast	指定したクライアントにメッセージを渡します
tp_call	同期型でTmaxサービスを呼び出します
tp_cancel	tp_acall()で返された記述子を取り消します
tp_commit	トランザクション作業をコミットするために使用します
tp_connect	会話型サービスと通信を接続します
tp_conv	cdによって指定された会話型サービスのメッセージを送受信します
tp_discon	会話の接続を強制切断し、TPEV_DISCONIMMイベントを発生させます
tp_fcall	同期型でTmaxサービスを呼び出すために使用されます
tp_getrply	非同期型通信でデータを受信します
tp_init	Tmaxとの接続を確立します
tp_rcv	会話型通信でデータを受信します
tp_send	会話型通信で相手にデータを送信します
tp_subscribe	サーバーでtpost()によって渡されるメッセージを取得します
tp_term	Tmaxシステムとの接続を解除します
tp_unsubscrib	ウィンドウ(hwnd)でメッセージを取得するように指定されたのを、tp_subscribe()、tp_setunsol()を使用して解除します
tpbegin	pb_tx_set_transaction_timeout()とpb_tx_begin()の機能が一気に実行できます
tuxcall	同期型でTmaxサービスを呼び出すために使用されます
tuxedo_compat	一部のAPIの動作に対してTuxedoライブラリーとの互換性を提供します
tuxreadenv	ファイルに保存されている接続するシステムの情報を読み込み、環境変数に新しい値を設定します

1.2. 関数

1.2.1. f_data

データ文字列を作成する関数です。

DataWindow(またはDataWindowChild、Datastore)に存在する列名とデータをタブ("~t")で区別し、各列名-データのペアは改行("~n")で区別する文字列を作成します。この文字列は、pb_tpcallのインプット部分に使用できます。DataWindow、DataWindowChild、Datastoreのデータの中から修正された行のみ抽出します。

- プロトタイプ

```
f_data ( {DataWindow | DataWindowChild | Datastore} ds_in )
returns string
```

- パラメータ

パラメータ	説明
ds_in	入力で使用されるPowerBuilderオブジェクトです

- 戻り値

DataWindow(またはDataWindowChild、DataStore)に存在する列名とデータをタブ("~t")と改行("~n")で区別された形式の文字列で返します。

- 例

```
string itype, input, form[], output[]

...
itype = "STRING~t~n"
input = uo_tmax.f_data(dw_input) + & "~n"

...
ret = uo_tmax.pb_tpcall("TOUPPER", itype, input, form[], output[])
if ret < 0 then
    error processing
    ...
end if
...
```

- 関連関数

f_datadel(), f_fdatadel(), pb_tpcall(), pb_tpacall(), pb_tpconnect()

1.2.2. f_datadel

DataWindowのDeletedBufferでデータ文字列を作成する関数です。DataWindow(またはDataWindowChild、DataStore)に存在する列名とデータをタブ("~t")で区別して、各列名-データのペアは改行("~n")で区別する文字列を作成します。この文字列は、pb_tpcallにてinput部分で使用できます。

f_data()は、dwDelBuf変数がTRUEの場合、DeletedBufferのデータと一緒に文字列を作成します。一方、f_datadel()は、dwDelBuf変数の状態を問わずDeletedBufferのデータに対してのみ文字列を作成します。DeletedBufferは、ユーザーがDeleteRow()関数を使用するときに削除されるデータが一時保存されるバッファです。f_datadel()が正常に動作するには、DataWindowがアップデート可能である必要があります。

DeleteRowを実行する際、DeletedBufferにデータが格納されるためには下記の事項に注意する必要があります。

– DataWindow「**Allow Update**」(PowerBuilderの[Rows] > [Update Properties..]メニューを参照)がチェックされている必要があります。

– ImportStringの実行後にResetUpdateを行わずにDeleteRowを実行した場合は、DeleteBufferに当該行が保存されません。

● プロトタイプ

```
f_datadel ( {DataWindow | DataWindowChild | Datastore} ds_in )  
returns string
```

● パラメータ

パラメータ	説明
ds_in	入力で使用されるPowerBuilderオブジェクト(DataWindow、DataWindowChild、DataStore可能)です

● 戻り値

DataWindow(またはDataWindowChild、DataStore)のDeletedBufferに存在するデータをタブ("~t")と改行("~n")で区別された形式の文字列で返します。

● 例

```
string itype, input, form[], output[]  
...  
itype = "STRING~t~n"  
input = uo_tmax.f_datadel(dw_input) + & "~n"  
...  
ret = uo_tmax.pb_tpcall("TOUPPER", itype, input, form[], output[])  
if ret < 0 then  
    error processing  
...  

```

```
end if
...
```

- 関連関数

`f_fdata()`、`f_fdatadel()`、`pb_tpcall()`、`pb_tpacall()`、`pb_tpconnect()`

1.2.3. f_fdata

データ文字列を作成する関数です。DataWindow(またはDataWindowChild、DataStore)データの中から修正された行のみ抽出します。DataWindowに存在する修正された行の列名とデータをf_dataと同様なフォーマットでfilehandle値のファイルに記録します。すなわち、列名とデータはタブ("~t")で区別し、各列名とデータのペアは改行("~n")で区別する文字列を作成します。

- プロトタイプ

```
f_fdata ( {Datastore|DataWindow|DataWindowChild} ds_in, integer filehandle )
returns integer
```

- パラメータ

パラメータ	説明
ds_in	入力で使用するPowerBuilderオブジェクト(DataWindow、DataWindowChild、DataStore可能)です
filehandle	一時ファイルのファイル記述子です。サーバー呼び出し後に返されたデータから抽出する、フィールド名を指定する文字列の整数型(integer)filehandleは、pb_tpfcall()で4番目のパラメータの一時ファイルに使用できます

- 戻り値

戻り値	説明
0	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です(例えば、ファイルへの書き込みに失敗した場合です)

- 例

```
...
InputFile = "c:\temp\fddata0.txt"
InputFile = FileOpen(InputFile, StreamMode!, Write!, LockReadWrite!, Replace!)
...
uo_tmax.f_fdata(dw_input, InputFile)
```

```

if FileClose(InputFile) <> 1 then
    error processing
    ...
end if
...
if uo_tmax.tp_fcall(input, form, output, inputfilename, fform, foutput) = -1
then
    error processing
    ...
end if
...

```

- 関連関数

f_data(), f_datadel(), f_fdatadel()

1.2.4. f_fdatadel

データ文字列を作成する関数です。DataWindow(またはDataWindowChild、DataStore)データの中からDeleteBufferに保存されている行のみ抽出します。DataWindowに存在する修正された行の列名とデータをf_dataと同様なフォーマットでfilehandle値のファイルに記録します。使用方法は、[f_data\(\)](#)関数を参照してください。

DeleteRowを行うとき、データがDeletedBufferに入るためには下記の事項に注意する必要があります。

- DataWindowの「Allow Update」(Power Builderの[Rows] > [Update Properties..]メニューを参照)がチェックされている必要があります。ImportStringの実行後、ResetUpdateを行わずにDeleteRowした場合は、DeleteBufferに当該行が保存されません。

- プロトタイプ

```

f_fdatadel ( {Datastore|DataWindow|DataWindowChild} ds_in, integer filehandle )

returns integer

```

- パラメータ

パラメータ	説明
ds_in	入力で使用されるPowerBuilderオブジェクト(DataWindow、DataWindowChild、DataStore可能)です
filehandle	一時ファイルのファイル記述子です

- 戻り値

戻り値	説明
0	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です(例えば、ファイルへの書き込みに失敗した場合です)

- 関連関数

f_data(), f_datadel(), f_data()

1.2.5. f_form

Form文字列を作成する関数です。DataWindow(またはDataWindowChild、DataStore)に存在する列名をpb_tpcallの4番目の引数で使用可能な形式の文字列で返します。

各列名を改行("~n")で区別してソートし、最後に改行("~n")をもう1つ追加した文字列を作成します。

- プロトタイプ

```
f_form ( {Datastore|DataWindow|DataWindowChild} ds_in)
returns string
```

- パラメータ

パラメータ	説明
ds_in	入力で使用されるPowerBuilderオブジェクト(DataWindow、DataWindowChild、DataStore可能)です

- 戻り値

DataWindow(またはDataWindowChild、DataStore)に存在する列名を改行("~n")で区別された形式の文字列で作成して返します。

- 例

```
string itype, input, form[], output[]
...
itype = "STRING~t~n"
input = uo_tmax.f_data(dw_input) + & "~n"
form[1] = uo_tmax.f_form(dw_input)
...
ret = uo_tmax.pb_tpcall("TOUPPER", itype, input, form[], output[])
if ret < 0 then
    error processing
...

```

```
end if
...
```

- 関連関数

pb_tpcall()、pb_tpgetrply()、pb_tprecv()、pb_tpconv()

1.2.6. f_getdata

Formと同様な構成になっているアウトプット内容の中で、arg_row配列のarg_col_name値を文字列で返す関数です。

- プロトタイプ

```
f_getdata ( string form, string odata, integer arg_row, string arg_col_name )
returns string
```

- パラメータ

パラメータ	説明
form	サービス処理結果、odataの構成形式の構造体バッファの場合は、使用された構造体の各メンバーで構成されます。また、フィールド・キー・バッファの場合は、使用されたフィールド名で構成されます。このとき、各メンバーまたはフィールド名は改行("~n")で区別され、最後に改行("~n")を追加してFormのエンドを表示します。最終的なForm配列のエンドを表すためには、改行("~n")で構成されたFormを追加します。
odata	サービス処理結果を保存するバッファデータは、PowerBuilderのImportString関数を使用して簡単にデータをDataWindowに格納できるようにタブ("~t")で区別されており、各行は改行("~n")で区別されています
arg_row	目的の文字列が存在している行です
arg_col_name	目的の文字列が存在している列です

- 戻り値

該当する文字列を返します。

- 例

```
string itype, input, form[], output[]
...
itype = "STRING~t~n"
input = uo_tmax.f_data(dw_input) + & "~n"
form[1] = uo_tmax.f_form(dw_input)
...
```

```

ret = uo_tmax.pb_tpcall("TOUPPER", itype, input, form[], output[])
if ret < 0 then
    error processing
    ...
end if
...

```

1.2.7. isblocked

ブロッキング状態なのかをチェックする関数です。ブロッキング状態とは、サービス呼び出し中を意味します。

サービスを呼び出している状態では、他のサービスを呼び出すことはできません。サービスの呼び出し中に、他のボタンをクリックするか、またはボタンをダブルクリックしてサービスを再び呼び出すとエラーが発生します。そのため、サービスを呼び出すボタンで最初にこの関数を呼び出してサービスが実行中の際には、待機メッセージを出力するのが望ましいです。

- プロトタイプ

```

isblocked ()
returns integer

```

- 戻り値

戻り値	説明
1	ブロッキング状態の場合です

- 例

```

...
ret = uo_tmax.isBlocked()
if ret = 0 then
    MessageBox("pb_isBlocked", "NO BLOCK " + string(uo_tmax.tperrno))
else
    MessageBox("pb_isBlocked", "BLOCKING")
end if
...

```

- 関連関数

pb_isblocked()

1.2.8. pb_etpcall

関数の使用法はpb_tpcallと同様で、既存のpb_tpcallが有する機能以外にイメージ・データを処理する関数があります。この場合は、必ずFIELD KEYバッファーを使用し、特定のフィールド名を使用しなければならない制約があります。イメージ・データを送信するときは「FILENAM」で、イメージ・データを受信するときは「NEWFILE」というフィールド名でファイルが指定される必要があります。さらに、実質的なイメージ・データはTP_BITMAPというCARRAY型のフィールド名で転送されます。

イメージを送信するときは、転送するファイル名をFILENAMというフィールド名でインプットに保存します(このデータは4MB以上になっても処理できます)。クライアント・ライブラリーでは、このフィールド名で指定されたファイルからイメージ・データを読み込み、TP_BITMAPというフィールド名のデータをサーバーに転送します。

イメージを受信するときは、サーバーに転送するイメージ・データを保存するファイル名を「NEWFILE」というフィールド名でidataに指定します。サービス・プログラム内では、「TP_BITMAP」というフィールド名でデータを転送し、クライアント・ライブラリーでは、「NEWFILE」フィールド名で指定されたファイルにイメージ・データを保存します。この場合、受信バッファー内に「NEWFILE」フィールド名のデータが存在する必要があります。

● プロトタイプ

```
pb_etpcall(string svcname, string itype, string idata, string form[],  
           ref string odata[])  
returns integer
```

● パラメータ

パラメータ	説明
svcname	呼び出すサービス名です
itype	使用したバッファー・タイプの文字列型です。タブ("~t")を使用して文字列を区別し、改行("~n")を使って行のエンドを表示します
idata	処理するデータが保存されるバッファーの文字列型です。タブ("~t")を使用して入力データを区別し、改行("~n")を使って行のエンドを表示します
form	サービス処理結果、odataの構成形式の構造体バッファーの場合は、使用された構造体の各メンバーで構成されます。フィールド・キー・バッファーの場合は、使用されたフィールド名で構成されます。各メンバーあるいはフィールド名は、改行("~n")で区別され、最後に改行("~n")を追加してFormのエンドを表示します。最終的なForm配列のエンドを表すために、改行("~n")で構成されたFormを追加します
odata	サービス処理結果を保存するバッファーの各データは、PowerBuilderのImportString関数を使用して簡単にDataWindowに格納できるようにタブ("~t")で区別されており、各行は改行("~n")で区別されています

下記は、Tmaxのインスタンス変数です。この中で、flagsとtimeoutはサービス呼び出し前に設定される値です。以外はサービス呼び出し後に設定される値です。

変数	説明
msg	サービス・プログラムから送信されたメッセージです
tperrno	エラーが発生した場合、コードが保存されるバッファです
tpurcode	サーバーから送信されたユーザーコード(tpreturnの2番目の引数)が保存されるバッファです
flags	<p>pb_tpcallで使用するフラグは下記のとおりです。</p> <ul style="list-style-type: none"> TPNOTRAN：非トランザクション・モードのサービスを呼び出す場合に使用します TPNOCHANGE：初期の割り当てバッファ・タイプの一致を保証します TPNOBLOCK：非ブロッキング・モードで呼び出す場合に使用します TPNOTIME：ブロッキング・タイムアウトを無視します <p>[参考]</p> <p>tuxedo_compat(true)設定の場合、フラグの動作が異なります。詳細についてはtuxedo_compat()を参照してください</p>
timeout	pb_tx_set_transaction_timeoutで使用するブロッキング・タイムアウトの時間です

- 戻り値

戻り値	説明
0	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です。tperrnoにエラーコードが保存され、msgにエラー・メッセージが保存されます

- 例

```

long ret
string itype, idata, form[], odata[]
string FileName = "c:\abc.bmp"

if isnull(sle_1.text) or sle_1.text = '' then
    error processing
    ...
end if

itype = "FML~t~n"
/*FILEをアップロードするためにFILENAMフィールドにファイル名を設定します。*/
idata = "FILENAM~t" + FileName + "~n"

```

```

form[1] = "~n"
odata[1] = space(1024)

ret = uo_tmax.pb_etpcall("IMGSAVE", itype, idata, form[], odata[])

if ret < 0 then
    error processing
    ...
end if

```

- 関連関数

f_data(), f_datadel(), f_form(), tuxedo_compat()

1.2.9. pb_getunsold

非要求メッセージを受信するための関数です。PowerBuilderでは、ライブラリー内のメモリーにアクセスできません。そのため、非要求メッセージを受信するには、PowerBuilderでメモリーを確保し、これをライブラリーに渡して受信する必要があります。

非要求メッセージを受信すると、pb_tpsetunsol()で設定されたウィンドウにmidとデータ長を渡します。当該ウィンドウでは、データの長さの分だけメモリーを確保してpb_getunsold()関数を呼び出すと、指定されたメモリーにデータが保存されます。pb_tpsetunsol()、pb_getunsold()を使用して非要求の受信メッセージを取得するときは、必ずu_tmaxのインスタンス変数であるflagsにTPUNSOL_HNDで設定した後、pb_tpstart()を行います。

- プロトタイプ

```

pb_getunsold ( ref string odata, integer len, unsignedinteger mid )
returns integer

```

- パラメータ

パラメータ	説明
odata	非要求受信データを取得するバッファーを設定します
len	非要求受信データ長です
mid	メッセージの記述子です

- 戻り値

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

- エラー

tperrnoにエラーコードが設定されます。

エラーコード	説明
TPEINVAL	入力パラメータが有効でない場合です。rcvbufがNULLか、midが0以下の場合です
TPEITYPE	PowerBuilderで確保されたメモリー領域が受信データより小さい場合に発生します
TPEOTYPE	受信データが存在しないか、不適切な場合です
TPEMATCH	与えられたmidに該当するデータが存在しない場合です

- 例

```
uo_tmax.flags = uo_tmax.TPUNSOL_HND
...
ret = uo_tmax.pb_tpsetunsol ( HANDLE(this), "", 1025, uo_tmax.TPBROADCAST )
if ret = -1 then
    error processing
    ...
end if
...
unsoldata = space(1024)
ret = uo_tmax.pb_getunsold(unsoldata, rlen, 1025)
if ret < 0 then
    error processing
    ...
end if
...
```

- 関連関数

pb_tpsetunsol()

1.2.10. pb_IsBlocked

ブロッキング状態をチェックする関数です。ブロッキング状態とは、サービス呼び出し中を意味します。

サービスを呼び出している状態では、他のサービスを呼び出すことはできません。サービスの呼び出し中に、他のボタンをクリックするか、またはボタンをダブルクリックしてサービスを再び呼び出すとエラーが発生します。そのため、サービスを呼び出すボタンで最初にこの関数を呼び出してサービスが実行中の際には、待機メッセージを出力するのが望ましいです。

- プロトタイプ

```
pb_isblocked ()  
returns integer
```

- 戻り値

戻り値	説明
1	ブロッキング状態の場合です

- 例

```
...  
ret = uo_tmax.pb_isBlocked()  
if ret = 0 then  
    MessageBox("pb_isBlocked", "NO BLOCK " + string(uo_tmax.tperrno))  
else  
    MessageBox("pb_isBlocked", "BLOCKING")  
end if  
...
```

- 関連関数

isBlocked()

1.2.11. pb_reset

pb_resetは、現在の接続を即解除する関数です。クライアント・モジュールにTPESYSTEMエラーが発生する場合の大半はネットワーク・エラーなので、Tmaxシステムに再接続することをお勧めします。pb_reset()関数は、このような場合に使用されます。まず、pb_reset()で接続を解除してサービスを再要求するか、Tmaxシステムに再接続します。

- プロトタイプ

```
pb_reset ()  
returns integer
```

- 戻り値

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

- エラー

tperrnoにエラー状況に該当するコードが設定されます。

エラーコード	説明
TPESYSTEM	Tmaxシステム・エラーが発生することを意味します。詳細情報はログファイルに記録されます
TPEOS	OSにエラーが発生することを意味します

- 例

```
long ret
u_tmax uo_tmax
uo_tmax = CREATE u_tmax
uo_tmax.usr_name = 'tmaxclient'
uo_tmax.clt_name = 'cilent01'
uo_tmax.dom_pwd = 'tmax'
uo_tmax.usr_pwd = 'client01'
uo_tmax.flags = uo_tmax.TPUNSOL_IGN
...
ret = uo_tmax.pb_tpstart()
if ret < 0 then
    error processing
    ...
end if
...
ret = uo_tmax.pb_tpreset()
if ret < 0 then
    error processing
    ...
endif
```

1.2.12. pb_tmaxreadenv

pb_tmaxreadenv()は、ファイルに保存された接続するシステムの情報を読み込み、環境変数に新しい値を設定する関数です。Tmaxシステムと接続するための情報を環境変数に設定するため、この関数はTmaxシステムに接続する前に実行される必要があります、Tmaxシステムに接続するには、いくつかの環境変数をシステムに登録する必要があります。このように登録された環境変数を参照して、pb_tpstart()でTmaxシステムに接続します。

通常、環境変数はオペレーティング・システムによって定義するファイルが異なります。UNIXの場合、cshは<.cshrc>に、kshは<.profile>に定義します。DOSの場合は、<autoexec.bat>ファイルに定義します。しかし、接続するシステムが2つ以上の場合、クライアントは状況に応じて接続しようとするシステムを変更すること

ができます。また、管理上の問題により環境変数ファイルを利用する場合があります。この場合は、環境変数に2つのシステムに関するシステム情報が登録できないため、ファイルに環境変数を登録して使用します。

```
[TMAX]
TMAX_HOST_ADDR=168.126.185.131
TMAX_HOST_PORT=8800
SDLFILE=C:\tmax\sample\sdl\tmax.sdl
FDLFILE=C:\tmax\sample\fdl\tmax.fdl
TMAX_CONNECT_TIMEOUT=2
```

● プロトタイプ

```
pb_tmaxreadenv(string env_file, string section)
returns integer
```

● パラメータ

パラメータ	説明
env_file	接続するシステムの環境情報が保存されたファイル名です。このファイルはテキスト型で、一定した形式で登録されている必要があります
section	ファイル内に登録された環境情報の記述子です。2つ以上のシステム情報を1つのファイルに登録する場合に各システムが区別できる値です

● 戻り値

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

● エラー

エラーコードの詳細内容については、『*Tmax* リファレンスガイド』の「3.1.37. tmaxreadenv」を参照してください。

● 例

```
long ret
u_tmax uo_tmax
uo_tmax = CREATE u_tmax
...
ret = uo_tmax.pb_tmaxreadenv("./tmax.env", "TMAX")
if ret < 0 then
    error processing
...
```

```
end if
...
```

- 関連関数

pb_tpstart()、pb_tpend()

1.2.13. pb_tpacall

非同期サービス呼び出しを実行する関数です。非同期型通信は、クライアントがサービスを要求してからも継続して他の作業が行えます。さらに、必要なタイミングで当該要求の応答が受けられます。要求の応答を受ける際に、(pb_tpgetrply())はpb_tpacall()の呼び出し時に返される参照識別子を使用します。サービスを要求するpb_tpacall()は呼び出し時に即リターンされ、他の作業が処理できます。応答を受けるために呼び出すpb_tpgetrply()は、応答が受信されるか、タイムアウトされるまでブロッキング状態で待機します。

- プロトタイプ

```
pb_tpacall ( string svcname, string itype, string idata )
returns integer
```

- パラメータ

パラメータ	説明
svcname	呼び出すサービス名です
itype	使用したバッファ・タイプの文字列型です。タブ("~t")を利用して文字列を区別し、改行("~n")を利用して行のエンドを表示します
idata	処理するデータが保存されるバッファの文字列型です。タブ("~t")を利用して入力データを区別し、改行("~n")を利用して行のエンドを表示します

- 戻り値

戻り値	説明
1以上のcd値	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です。tperrnoにエラーコードが保存されます

- エラー

エラーコードの詳細内容については、『Tmax リファレンスガイド』の「3.1.41. tpacall」を参照してください。

- 例


```

string input, itype, form[], output[]
int length, cd, ret

length = len(before.text)

itype = "STRUCT~tkstrdata~n"
input = string(length) + "~t" + before.text + "~n"
form[1] = "len~n" + "sdata~n" + "~n"
form[2] = "~n"
output[1] = space(1024)

cd = uo_tmax.pb_tpacall("SYNC", itype, input)
if cd < 0 then
    error processing
    ...
end if

ret = uo_tmax.pb_tpgetrply(cd, form, output)
if ret < 0 then
    error processing
    ...
end if

```

- 関連関数

pb_tpgetrply()、pb_tpcancel()、pb_tpcall()

1.2.14. pb_tpalloc

pb_tpalloc()はitypeで指定されたバッファ・タイプを割り当て、これに対するアドレスを返す関数です。<tmax4gl.dll>のtpalloc()を呼び出します。タイプによってisubtypeとilenは選択的に指定できます。その後、新規作成されたpb_b~関数とpb_tpput、pb_tpgetなどの関数のbufファクターに使用されます。

指定されたバッファ・タイプが下位タイプを使用することができる場合は、pb_tpalloc()の呼び出し時に必ずisubtypeを指定します。指定されたバッファ・タイプが下位タイプを使用しない場合は、subtypeは無視されます(通常0が使用されます)。割り当てられたバッファ・サイズは、デフォルト値の1024Byte以上です。

参考

関数の詳細内容については、『*Tmax リファレンスガイド*』の「3.1.44. tpalloc」を参照してください。

- プロトタイプ

```
pb_tmalloc ( string itype, string isubtype, long ilen )
returns long
```

- パラメータ

パラメータ	説明
itype	割り当てるバッファ・タイプです(例 : STRUCT、STRING、CARRAY、FIELD)
isubtype	指定されたバッファ・タイプの下位タイプです
ilen	割り当てるバッファ・サイズです(デフォルト値 : 1024)

- 戻り値

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

- エラー

tperrnoにエラーコードが設定されます。

エラーコード	説明
TPEINVAL	パラメータが有効でないことを意味します(例 : NULL形式)
TPENOENT	不明か、または下位タイプです。構造体バッファ・タイプの場合、下位タイプが(構造体のタグ名)SDLFILEに存在しない場合に発生します。 クライアントでtpstart()を呼び出さないか、呼び出す前にtpalloc()で構造体バッファを割り当てた場合にもこのエラーが発生します
TPESYSTEM	Tmaxシステムエラーが発生したことを意味します。詳細情報はログファイルに記録されます
TPEOS	メモリー割り当てが受けられないOSにエラーが発生したことを意味します
TPEOTYPE	要求されるサーバーのデータ型が構造体バッファに関わらず、当該サーバーのコンパイル時に構造体ファイルと一緒にコンパイルされない場合です

- 例

```
long ret
long buf
u_tmax uo_tmax
uo_tmax = CREATE u_tmax
```

```

uo_tmax.usr_name = 'tmaxclient'
uo_tmax.clt_name = 'cilent01'
uo_tmax.dom_pwd = 'tmax'
uo_tmax.usr_pwd = 'client01'
uo_tmax.flags = uo_tmax.TPUNSOL_IGN
...
ret = uo_tmax.pb_tpstart()
if ret < 0 then
    error processing
    ...
end if
...
buf = uo_tmax.pb_tpalloc("CARRAY", "", 40000)
...

```

1.2.15. pb_tpbroadcast

指定したクライアントにメッセージを転送する関数です。

- プロトタイプ

```

pb_tpbroadcast ( string nodename, string username, string cltname, string data )

returns integer

```

- パラメータ

パラメータ	説明
nodename	メッセージを転送するクライアントが属しているノード名です
username	メッセージを転送するユーザー名です。pb_tpstart()のときに設定するusernameと一致する場合のみメッセージが転送されます
cltname	メッセージを転送するクライアント名です。pb_tpstart()のときに設定するcltnameと一致する場合のみメッセージが転送されます
data	処理するデータが保存されるバッファの文字列型です。タブ("~t")を利用して入力データを区別し、改行("~n")を利用して行のエンドを表示します

参考

nodename、username、cltnameはすべて対象クライアントの選択時に使用される論理的な名前です。名前の指定には、ワイルドカード文字が使用できます。usernameとcltnameは、設定される名前に疑問符(?) やアスタリスク(*)などのワイルドカードが使用できますが、nodenameは名前全体に代わってアスタリスク

(*)のみ使用可能です。また、NULL値が使用できますが、これはすべてのクライアントに対応するワイルドカードとして動作します。

- 戻り値

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

- エラー

エラーコードの詳細内容については、『*Tmax リファレンスガイド*』の「3.1.46. tpbroadcast」を参照してください。

- 例

```
string nodename, username, cltname, data
int ret

...
nodename = "tmax"
username = "*"
cltname = "*"
data = sle_input.text
ret = uo_tmax.pb_tpbroadcast(nodename, username, cltname, data)
if ret < 0 then
    error processing
    ...
end if
...
```

1.2.16. pb_tpcall

同期処理関数で、処理結果を取得するまで待機します。

処理するデータをidataに保存して指定されたサービスをsvcnameに入力した後サービスを呼び出します。処理結果は、Formに指定された列のデータとしてodataに保存されます。

- プロトタイプ

```
pb_tpcall(string svcname, string itype, string idata, string form[],
          ref string odata[])
returns integer
```

● パラメータ

パラメータ	説明
svcname	呼び出すサービス名です
itype	<p>使用したバッファ・タイプの文字列型です。タブ("~t")を利用して文字列を区別し、改行("~n")を利用して行のエンドを表示します</p> <p>– STRUCTバッファ・タイプ itype = "STRUCT~tstruct_name~n"</p> <p>– STRINGバッファ・タイプ itype = "STRING~t~n"</p> <p>– FIELD KEYバッファ・タイプ itype = "FIELD~t~n"</p>
idata	処理するデータが保存されるバッファの文字列型です。タブ("~t")を利用して入力データを区別し、改行("~n")を利用して行のエンドを表示します
form	<p>サービス処理結果、odataの構成であり、構造体バッファの場合は、使用された構造体の各メンバーで構成され、フィールド・キー・バッファの場合には、使用されたフィールド名で構成されます。</p> <p>各メンバーまたはフィールド名は改行("~n")で区別され、最後に改行("~n")を追加してFormのエンドを表示します。最終的なForm配列のエンドを表示するには、改行("~n")で構成されたFormを追加します</p>
odata	サービス処理結果を保存するバッファの各データは、PowerBuilderのImportString関数を使用して簡単にデータをDataWindowに格納できるようにタブ("~t")で区別されており、各行は改行("~n")で区別されています

下記はTmaxのインスタンス変数です。flagsとtimeoutは、サービス呼び出しの前に設定される値です。以外はサービス呼び出し後に設定される値です。

変数	説明
msg	サービス・プログラムから転送されたメッセージです
tperrno	エラーが発生した場合、エラーコードが保存されるバッファです
tpurcode	サーバーから転送されたユーザーコード(tpreturnの2番目の引数)が保存されるバッファです
flags	<p>pb_tpcallで使用するフラグは下記のとおりです</p> <p>– TPNOTRAN : 非トランザクション・モードのサービスを呼び出すときに使用します</p> <p>– TPNOCHANGE : 初期割り当てバッファと応答バッファ形式の一致を保証します</p> <p>– TPNOBLOCK : 非ブロッキング・モードで呼び出すときに使用します</p> <p>– TPNOTIME : ブロッキング・タイムアウトを無視します</p>

変数	説明
timeout	pb_tx_set_transaction_timeoutで使用するブロッキング・タイムアウトの時間値です

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です。tperrnoにエラーコードが設定され、msgにエラー・メッセージが保存されます

- エラー

エラーコードの詳細内容については、『*Tmax リファレンスガイド*』の「3.1.47. tpcall」を参照してください。

- 例

- 構造体バッファーを使用する場合

```

string itype, input, form[], output[]
int ret
itype = "STRUCT~t" + "kstrdata~n"
input = string(len(before.text)) + "~t" + before.text + "~n"
form[1] = "len" + "~n" + &
    "sdata" + "~n" + &
    "~n"
form[2] = "~n"
output[1] = space(1024)

uo_tmax.flags = uo_tmax.TPNOCHANGE
uo_tmax.timeout = 50

ret = uo_tmax.pb_tpcall("SYNC", itype, input, form, output)
if ret < 0 then
    messagebox("ERROR", "tpcall Failed : Error Code =" + string(uo_tmax.tperrno))

    return
end if

afterlen.text = uo_tmax.f_getdata(form[1], output[1], 1, "len")
afterdata.text = uo_tmax.f_getdata(form[1], output[1], 1, "sdata")

```

- フィールド・キー・バッファーを使用する場合

```

long ret
string itype, input, form[], output[]

if isnull(sle_1.text) or sle_1.text = '' then
...
end if

itype = "FIELD~t~n"
input = "INPUT~t" + sle_1.text + "~n"

form[1] = "OUTPUT~n~n"
form[2] = "~n"
output[1] = space(40)

ret = uo_tmax.pb_tpcall("FDLToupper", itype, input, form[], output[])
if ret < 0 then
    error processing
    ...
end if
...

```

- 関連関数

pb_tpcall(), pb_tpgetrply()

1.2.17. pb_tpcallw

DataWindowを使用して同期型でTmaxサービス呼び出すための関数です。

iobjectに指定されたPowerObjectを入力して呼び出した後、返された結果をoobjectに指定されたPowerObjectに格納します。この関数はフィールド・バッファを使用します。

- プロトタイプ

```

pb_tpcallw ( string svcname, powerobject iobject[], powerobject oobject[] )
returns integer

```

- パラメータ

パラメータ	説明
svcname	呼び出すサービス名です
iobject	iobjectに指定されたPowerObjectを入力してサービス呼び出します
oobject	サービス呼び出し後に返された結果を、oobjectに指定されたPower Objectに格納します

インプットに使用できるオブジェクトは下記のとおりです。

オブジェクト	説明
DataWindow, DataWindowChild, DataStore	列名をフィールド・バッファのフィールド名で使⽤します。修正された⾏のみデータとして使⽤します
SingleLineEdit, MultiLineEdit	Control.Tag名をフィールド・バッファのフィールド名で使⽤します。Control.Textをデータとして使⽤します
StaticText, EditMask	Control.Tag名をフィールド・バッファのフィールド名で使⽤します。Control.Textをデータとして使⽤します
ListBox	Control.Tag名をフィールド・バッファのフィールド名で使⽤します。選択されたすべての項目をデータとして使⽤します
DropDownListBox	Control.Tag名をフィールド・バッファのフィールド名で使⽤します。選択された項目のみデータとして使⽤します
CheckBox	Control.Tag名をフィールド・バッファのフィールド名で使⽤します。チェックされている場合のみデータとして使⽤します
RadioButton	Control.Tag名をフィールド・バッファのフィールド名で使⽤します。チェックされている場合のみデータとして使⽤します

アウトプットに使用できるオブジェクトは、インプットに使用可能なオブジェクトと同様です。ListBoxとDropDownListBoxの場合はデータを追加します。下記は、使用されるuo_tmaxのインスタンス変数に関する説明です。

変数	説明
Append	TRUEの場合、アウトプットにDataWindowが指定されると、DataWindowに存在していた既存のデータを削除せず継続して追加します(デフォルト値: FALSE)
DelBuf	TRUEの場合、インプットにDataWindowが指定されると、DataWindowのDeleteBufferとPrimaryBufferからデータを抽出します(デフォルト値: TRUE)
DataOnFail	TRUEの場合、サービスが失敗してもデータを画面に出力します (デフォルト値: FALSE)
AppData	サービスを呼び出すたびに自動で追加されるデータをpb_tpcall()のインプット形式で指定します
DateFmt, TimeFmt, DateTimeFmt	Date/Time/DateTime型の列値の場合、文字列型に変換する際に参照されるフォーマットを指定します

pb_tpcallwは内部的にpb_tpcallを使用するため、上記の変数以外にも、[pb_tpcall\(\)](#)で使用される変数を参考にしてください。

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です。tperrnoにエラーコードが設定され、msgにエラー・メッセージが保存されます

- 例

```
long ret
windowobject indata[], outdata[]

indata[1] = dw_1
indata[2] = sle_input
outdata[1] = dw_2
uo_tmax.flags = 0;

ret = uo_tmax.pb_tpcallw ("SVC1", indata, outdata)
if ret < 0 then
    error processing
    ...
end if
```

- 関連関数

pb_tpcall()、tuxcall()

1.2.18. pb_tpcancel

pb_tpcancel()はpb_tpcall()が返した記述子を取り消す関数です。呼び出したサービスの受信を取り消します。

グローバル・トランザクションに該当する記述子を取り消すとエラーが発生します。呼び出しに成功するとこれ以上cdを使用することができません。この関数を利用して受信を取り消した場合、当該トランザクションはコミットできず、ロールバックのみ可能です。

- プロトタイプ

```
pb_tpcancel ( integer cd ) returns integer
```

- パラメータ

パラメータ	説明
cd	pb_tpacall()が返した参照記述子です

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です。tperrnoにエラーコードが設定され、msgにエラー・メッセージが保存されます

- エラー

エラーコードの詳細内容については、『Tmax リファレンスガイド』の「3.1.49. tpcancel」を参照してください。

- 例

```
string input, itype, form[], output[]
int length, cd, ret

length = len(before.text)

itype = "STRUCT~tkstrdata~n"
input = string(length) + "~t" + before.text + "~n"
form[1] = "len~n" + "sdata~n" + "~n"
form[2] = "~n"
output[1] = space(1024)

cd = uo_tmax.pb_tpacall("SYNC", itype, input)
if cd < 0 then
    error processing
    ...
end if

ret = uo_tmax.pb_tpcancel(cd)
if ret < 0 then
    error processing
    ...
end if
```

- 関連関数

pb_tpacall()、pb_tpgetrply()

1.2.19. pb_tpcommit

トランザクション作業のコミット時に使用される関数です。

- プロトタイプ

```
pb_tpcommit ( ) returns integer
```

- 戻り値

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

- エラー

エラーコードの詳細内容については、『*Tmax リファレンスガイド*』の「3.1.105. tx_commit」を参照してください。

- 例

```
...
if _is_tx <> true then
...
end if

ret = uo_tmax.pb_tpcommit()
if ret < 0 then
    error processing
    ...
end If

_is_tx = false
...
```

- 関連関数

pb_tx_begin()、pb_tx_rollback()、pb_tpbegin()

1.2.20. pb_tpconnect

pb_tpconnect()は会話型サービスと通信を接続する関数です。接続されたサーバーとクライアントは相互pb_tpsend()、pb_tprecv()でデータを送受信します。

- プロトタイプ

```
pb_tpconnect ( string svcname, string itype, string idata, long arg_flag )
returns integer
```

- パラメータ

パラメータ	説明
svcname	呼び出す会話型サービス名です
itype	使用したバッファ・タイプの文字列型です。タブ("~t")を利用して文字列を区別し、改行("~n")を利用して行のエンドを表示します
idata	処理するデータが保存されるバッファの文字列型です。タブ("~t")を利用して入力データを区別し、改行("~n")を利用して行のエンドを表示します
arg_flag	フラグ値です

pb_tpconnect()で利用できるフラグ値は下記のとおりです。

フラグ値	説明
TPNOTRAN	pb_tpconnect()のcallerが、トランザクション・モード状態でこのフラグを設定してsvcサービスを要求した場合は、svcサービスはトランザクション・モードから除外され実行されます。 トランザクション・モードにおいて、svcがトランザクションをサポートできないサービスの場合は、トランザクション・モードでpb_tpconnect()が呼び出されるときにフラグは必ずTPNOTRANに設定します。トランザクションpb_tpconnect()の呼び出し時に、TPNOTRANで設定されていても依然としてトランザクション・タイムアウトの影響を受けます。TPNOTRANで呼び出されたサービスが失敗した場合、callerのトランザクションには影響を与えません
TPSENDONLY	接続が完了された後、関数callerは最初のデータ送信のみ可能です。要求されたサービスのみ行えるように設定するフラグで、callerが最初の通信制御権を有します。TPSENDONLYまたはTPRECVONLYのうち1つは必ず指定します
TPRECVONLY	接続が完了された後、関数callerはデータ受信のみ可能です。要求されたサービスが最初のデータ送信を開始するようにするフラグで、要求されたサービスが最初の通信制御権を有します。TPSENDONLYまたはTPRECVONLYのうち1つは必ず指定します

フラグ値	説明
TPNOTIME	TPNOTIMEフラグは関数callerがブロッキング・タイムアウトを無視し、応答が受信されるまで継続して待機することを意味します。しかし、トランザクション・タイムアウト内でpb_tpconnect()を行った場合は、依然としてブロッキング・タイムアウトが適用されます
TPSIGSTRT	シグナル割り込み(Interrupt)を受け入れるときに使用します。内部でシグナル割り込みが発生し、システム関数呼び出しに問題があるときにシステム関数呼び出しが再実行されます。フラグが設定されずにシグナル割り込みが発生した場合は、関数は失敗しtperrnoにTPGOTSIGが設定されます

- 戻り値

戻り値	説明
参照される記述子 (1以上のcd値)	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です。tperrnoにエラーコードが設定され、msgにエラー・メッセージが保存されます

- エラー

エラーコードの詳細内容については、『*Tmax リファレンスガイド*』の「3.1.51. tpconnect」を参照してください。

- 例

```
int ret, cd
string itype, otype, idata, form[], odata[]

itype = "STRING~t~n"
otype = "STRING~t~n"
idata = string(len(before.text)) + "~t" + before.text + "~n"
odata[1] = space(1024)

cd = uo_tmax.pb_tpconnect("CONV", itype, idata, uo_tmax.TPRECVONLY)
if cd < 0 then
    error processing
    ...
end if

ret = uo_tmax.pb_tprecv(cd, form, odata, uo_tmax.TPFLAGS)
if ret < 0 then
    if uo_tmax.revent <> uo_tmax.TPEV_SENDOONLY or &
```

```

        uo_tmax.revent = uo_tmax.TPEV_SVCSUCC then
            error processing
        ...
    end if
    ...
end if

```

- 関連関数

pb_tpsend(), tp_tprecv(), pb_tpdicon()

1.2.21. pb_tpconv

cdによって指定された会話型サービスのメッセージを送受信する関数です。tp_conv()はTmaxでサポートするバッファ・タイプのうち、フィールド・バッファを使用します。

- プロトタイプ

```

pb_tpconv ( long cd, string itype, string idata, ref string form[],
            ref string odata[], long arg_flag )
returns integer

```

- パラメータ

パラメータ	説明
cd	tp_connect()が返した参照cd記述子です
itype	<p>使用したバッファ・タイプの文字列型です。タブ("~t")を利用して文字列を区別し、改行("~n")を利用して行のエンドを表示します</p> <ul style="list-style-type: none"> STRUCTバッファ・タイプ itype = "STRUCT~tstruct_name~n" STRINGバッファ・タイプ itype = "STRING~t~n" FIELD KEYバッファ・タイプ itype = "FIELD~t~n"
idata	サービス名を含めてデータ文字列を構成します。文字列型であり、タブ("~t")を利用して入力データを区別し、改行("~n")を利用して行のエンドを表示します
form	サービス処理結果、odata構成形式のフィールド名は改行("~n")で区別され、最後に改行("~n")を追加してFormのエンドを表示します。最終的なForm配列のエンドを表示するには、改行("~n")で構成されたFormを追加します
odata	サービス処理結果を保存するバッファの各データは、PowerBuilderのImportString関数を使用して簡単にデータをDataWindowに格納できるようにタブ("~t")で区別されており、各行は改行("~n")で区別されています
arg_flag	pb_tpsend(), pb_tprecv()で使用したフラグ値がすべて使用できます

- 例

```
int ret, cd
string idata, itype
itype = "FIELD~t~n"
input = "INPUT~t" + sle_1.text + "~n"

form[1] = "OUTPUT~n~n"
form[2] = "~n"
output[1] = space(1024)

cd = uo_tmax.tp_connect(idata, uo_tmax.TPRECVONLY)
if cd < 0 then
    error processing
    ...
end if

ret = uo_tmax.pb_tpconv(cd, itype, idata, form, output, uo_tmax.TPNOFLAGS)
if ret < 0 then
    if uo_tmax.revent <> uo_tmax.TPEV_SENDOONLY or &
        uo_tmax.revent = uo_tmax.TPEV_SVCSUCC then
        error processing
        ...
    end if
    ...
end if
...
```

1.2.22. pb_tpdicon

pb_tpdicon()は会話の接続を強制解除し、TPEV_DISCONIMMイベントを発生させる関数で、会話型通信を開始した側でのみ呼び出せます。

- プロトタイプ

```
pb_tpdicon ( integer cd ) returns integer
```

- パラメータ

パラメータ	説明
cd	pb_tpconnect()が返した参照記述子です

- 戻り値

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

- エラー

エラーコードの詳細内容については、『Tmax リファレンスガイド』の「3.1.54. tpdiscon」を参照してください。

- 例

```
int ret, cd
string itype, otype, idata, form[], odata[]

itype = "STRING~t~n"
otype = "STRING~t~n"
idata = string(len(before.text)) + "~t" + before.text + "~n"
odata[1] = space(1024)

cd = uo_tmax.pb_tpconnect("CONV", itype, idata, uo_tmax.TPSENDONLY)
if cd < 0 then
    error processing
    ...
end if

ret = uo_tmax.pb_tpdiscon(cd)
if ret < 0 then
    error processing
    ...
end if
```

- 関連関数

pb_tpconnect()、tp_tpsend()、pb_tprecv()

1.2.23. pb_tpend

Tmaxシステムとの接続を解除する関数です。

- プロトタイプ

```
pb_tpend() returns integer
```


- 戻り値

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

- エラー

エラーコードの詳細内容については、『Tmax リファレンスガイド』の「3.3.3. tpend」を参照してください。

- 例

```
long ret
u_tmax uo_tmax
uo_tmax = CREATE u_tmax
...
ret = uo_tmax.pb_tpend()
if ret < 0 then
    error processing
...
end if
```

- 関連関数

tp_term()

1.2.24. pb_tpfcall

pb_tpcall()関数を拡張した形式です。ファイルから入力データが取得可能で、結果データをファイルに保存できます。Tmaxでサポートするバッファ・タイプのうち、フィールド・バッファを使用しています。結果データの中でformに指定されたフィールドのデータはアウトプットに保存され、fformに指定されたフィールドのデータはfodataのファイルに保存されます。

- プロトタイプ

```
pb_tpfcall ( string svcname, string itype, string idata, string form[],
             ref string odata[], string ifilename, string fform[],
             ref string fodata[] )
returns integer
```

- パラメータ

パラメータ	説明
svcname	呼び出すサービス名です

パラメータ	説明
itype	使用したバッファ・タイプの文字列型です。タブ("~t")を利用して文字列を区別し、改行("~n")を利用して行のエンドを表示します
idata	処理するデータが保存されるバッファの文字列型です。タブ("~t")を利用して入力データを区別し、改行("~n")を利用して行のエンドを表示します
form	<p>サービス処理結果、odata構成形式の構造体バッファの場合は、使用された構造体の各メンバーで構成され、フィールド・キー・バッファの場合は、使用されたフィールド名で構成されます。</p> <p>各メンバーまたはフィールド名は改行("~n")で区別され、最後に改行("~n")を追加してFormのエンドを表示します。最終的なForm配列のエンドを表示するには、改行("~n")で構成されたFormを追加します</p>
odata	サービス処理結果を保存するバッファの各データは、PowerBuilderのImportString関数を使用して簡単にデータをDataWindowに格納できるようにタブ("~t")で区別されており、各行は改行("~n")で区別されています
ifilename	<p>入力するデータが保存されているファイル名ファイルの形式は、フィールド名とデータをタブ("~t")で区別し、データのエンドを表示する改行("~n")を付けます</p> <p>例) FIELD_NAME1+ "~t" + VALUE1+ "~n" + & FIELD_NAME2+ "~t" + VALUE2+ "~n" + & "~n"</p>
fform	<p>サービス処理結果、fodata構成形式の構造体バッファの場合は、使用された構造体の各メンバーで構成され、フィールド・キー・バッファの場合は、使用されたフィールド名で構成されます。</p> <p>各メンバーまたはフィールド名は、改行("~n")で区別され、最後に改行("~n")を追加してFormのエンドを表示します。最終的なForm配列のエンドを表示するには、改行("~n")で構成されたFormを追加します</p>
fodata	サービス処理結果を保存するファイル名です

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です。tperrnoにエラーコードが設定され、msglにエラー・メッセージが保存されます

- 例

```
...
InputFile = "c:\temp\fdata0.txt"
```

```

InputFile = FileOpen(InputFile, StreamMode!, Write!, LockReadWrite!, Replace!)
...
uo_tmax.f_fdata(dw_input, InputFile)

if FileClose(InputFile) <> 1 then
    error processing
    ...
end if
...
ret = uo_tmax.pb_tpfcall("FCALL", itype, input, form, output,
                        inputfilename, fform, foutput)

if ret = -1 then
    error processing
    ...
end if
...

```

- 関連関数

f_fdata

1.2.25. pb_tpfree

pb_tpfree()はpb_tpalloc()で取得したバッファを解除する関数です。tpfree()はバッファを解除する前にこれに関連する情報も削除します。pb_tpfree()が正常に実行されると、bufはXATMIルーチンにパラメータとして渡されることができず、他の方式で使用できません。

- プロトタイプ

```
pb_tpfree ( long buf )
```

- パラメータ

パラメータ	説明
buf	以前、pb_tpalloc()で取得したバッファのアドレスです。bufがNULLの場合は何にも起こりません。バッファを削除するプロセスにおいて、一部のバッファ・タイプは関連データや状態情報を解除する必要があります

- 戻り値

pb_tpfree()は、関数callerに何の値も返しません。

- 例

```

long ret
long buf

u_tmax uo_tmax
uo_tmax = CREATE u_tmax
...
ret = uo_tmax.pb_tpstart()
if ret < 0 then
    error processing
    ...
end if
...
buf = uo_tmax.pb_tpalloc("CARRAY", "", 40000)
if buf = 0 then
    error process
    ...
end if
...
pb_tpfree(buf)

```

- 関連関数

pb_tpalloc()

1.2.26. pb_tpget

1番目のパラメータで指定したアドレスに該当するデータを、2番目のパラメータで指定したバッファに3番目のパラメータで指定したサイズだけ読み込む関数です。

PowerBuilderでは、char *というアドレスの概念がないため、long型のアドレスを<pb.dll>に渡すことになります。<pb.dll>から当該アドレスに存在するstringデータを取得して2番目のパラメータで入力します。

- プロトタイプ

```

pb_tpget ( long buf, ref string loc, long size )
returns long

```

- パラメータ

パラメータ	説明
buf	読み込むデータのアドレスをlong型で指定します
loc	指定したアドレスに該当するデータを<pb.dll>から取得した後、locに保存します
size	読み込むデータの長さを指定します

- 戻り値

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

- エラー

tperrnoにエラーコードが設定されます。

エラーコード	説明
TPEINVAL	入力パラメータが有効でない場合です。rcvbufがNULLか、midが0以下の場合です
TPEITYPE	PowerBuilderで確保されたメモリの領域が受信データより小さい場合に発生します
TPEOTYPE	受信データが存在しないか、不適切な場合です
TPEMATCH	与えられたmidに該当するデータが存在しない場合です

- 例

```
ret_string = Space(20)
ll_ret = uo_tmax.pb_tpget(buf, ret_string, 20)
```

- 関連関数

pb_tpput()

1.2.27. pb_tpgetrply

pb_tpacall()からの要求に対する応答を取得するための関数で、基本的にブロッキング通信です。一旦呼び出すと、応答を受けるか、ブロッキング・タイムアウトが発生するまで待機します。タイムアウトが発生すると呼び出しは失敗し、tperrnoにTPETIMEが設定されます。

- プロトタイプ

```
pb_tpgetrply ( integer cd, ref string form[], ref string odata[] )
returns integer
```

- パラメータ

パラメータ	説明
cd	pb_tpacall()が返した参照記述子です。要求に対応する応答が識別できるようにします

パラメータ	説明
form	<p>サービス処理結果、odata構成形式の構造体バッファの場合は、使用された構造体の各メンバーで構成され、フィールド・キー・バッファの場合は、使用されたフィールド名で構成されます。</p> <p>各メンバーまたはフィールド名は改行("~n")で区別され、最後に改行("~n")を追加してFormのエンドを表示します。最終的なForm配列のエンドを表示するには、改行("~n")で構成されたFormを追加します</p>
odata	サービス処理結果を保存するバッファの各データは、PowerBuilderのImportString関数を使用して簡単にデータをDataWindowに格納できるようにタブ("~t")で区別されており、各行は改行("~n")で区別されています

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です。tpernoにエラーコードが設定され、msglにエラー・メッセージが保存されます

- エラー

エラーコードの詳細内容については、『Tmax リファレンスガイド』の「3.1.71. tpgetrply」を参照してください。

- 例

```

string input, itype, form[], output[]
int length, cd, ret

length = len(before.text)

itype = "STRUCT~tkstrdata~n"
input = string(length) + "~t" + before.text + "~n"
form[1] = "len~n" + "sdata~n" + "~n"
form[2] = "~n"
output[1] = space(1024)

cd = uo_tmax.pb_tpacall("SYNC", itype, input)
if cd < 0 then
    error processing
    ...
end if

```

```

ret = uo_tmax.pb_tpgetrply(cd, form, output)
if ret < 0 then
    error processing
    ...
end if

```

- 関連関数

f_form()、pb_tpacall()、pb_tpcancel()、pb_tpcall()

1.2.28. pb_tpgetunsol

非要求受信メッセージを処理するための関数です。pb_tpsetunsol()とは違ってブロッキングされる場合があるため、使用時には注意が必要です。詳細内容については、『*Tmax* リファレンスガイド』の「3.3.5. tpgetunsol」を参照してください。

- プロトタイプ

```

pb_tpgetunsol ( integer types, string form[], ref string odata[],
               integer arg_flags )
returns integer

```

- パラメータ

パラメータ	説明
types	非要求メッセージの受信タイプを設定します。設定できる値はTPBROADCAST、TPSENDTOCLIです
form	非要求受信データのFormを設定します
odata	非要求受信データを受けるバッファを設定します
arg_flag	ブロッキング処理可否を指定するパラメータで、下記のうち1つを設定します <ul style="list-style-type: none"> – TPBLOCK: 関数呼び出し時に、ブロッキング状態で一方的なメッセージが送信されるまで待機します – TPNOTIME: 関数呼び出し時にブロッキングせず、メッセージが存在しない場合は直ちに次のルーチンを実行します

- 戻り値

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

- エラー

tperrnoにエラーコードが設定されます

エラーコード	説明
TPEINVAL	入力パラメータが有効でない場合です。rcvbufがNULLか、midが0以下の場合です
TPEITYPE	PowerBuilderで確保したメモリー領域が受信データより小さい場合に発生します
TPEOTYPE	受信データが存在しないか、不適切な場合です
TPEMATCH	与えられたmidに該当するデータが存在しない場合です

- 例

```
string form[], tmp[]
...
tmp[1] = space(30)

ret = uo_tmax.pb_tpgetunsol(uo_tmax.TPBROADCAST, form[], tmp[],
                           uo_tmax.TPBLOCK)

if ret < 0 then
    error processing
    ...
end if
...
```

- 関連関数

pb_tpsetunsol()、pb_getunsold()

1.2.29. pb_tpput

1番目のパラメータで指定したアドレスに、2番目のパラメータで指定したデータを、3番目のパラメータで指定したサイズだけ保存する関数です。

- プロトタイプ

```
pb_tpput ( long buf, any value, long size )
returns long
```

- パラメータ

パラメータ	説明
buf	保存するデータのアドレスをlong型で指定します
loc	当該アドレスに保存するデータを指定します

パラメータ	説明
size	保存するデータ長を指定します

- 戻り値

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

- エラー

tperrnoにエラーコードが設定されます。

エラーコード	説明
TPEINVAL	入力パラメータが有効でない場合です。rcvbufがNULLか、midが0以下の場合です
TPEITYPE	PowerBuilderで確保したメモリー領域が受信データより小さい場合に発生します
TPEOTYPE	受信データが存在しないか、不適切な場合です
TPEMATCH	与えられたmidに該当するデータが存在しない場合です

- 例

```
blob lb_tmp
bytes_read = FileRead(li_FileNum, lb_tmp)
ll_ret = uo_tmax.pb_tpput(buf, lb_tmp, bytes_read)
```

- 関連関数

pb_tpget()

1.2.30. pb_tprecv

pb_tprecv()は会話型通信でデータを受信するときに使用されます。callerは、通信制御権を有してはいけません。

- プロトタイプ

```
pb_tprecv ( integer cd, ref string form[], ref string odata[], long arg_flag )
returns integer
```

- パラメータ

パラメータ	説明
cd	pb_tpconnect()が返した参照記述子です
form	サービス処理結果、odata構成形式の構造体バッファの場合は、使用された構造体の各メンバーで構成され、フィールド・キー・バッファの場合は、使用されたフィールド名で構成されます。 各メンバーまたはフィールド名は改行("~n")で区別され、最後に改行("~n")を追加してFormのエンドを表示します。最終的なForm配列のエンドを表示するには、改行("~n")で構成されたFormを追加します
odata	サービス処理結果を保存するバッファの各データは、PowerBuilderのImportString関数を使用して簡単にデータをDataWindowに格納できるようにタブ("~t")で区別されており、各行は改行("~n")で区別されています
arg_flag	フラグ値です

pb_tprecv()で利用できるフラグ値は下記のとおりです。

フラグ値	説明
TPNOBLOCK	データが受信されるまで待機しません。受信できるデータが存在する場合はこれを返します。このフラグが指定されておらず、受信できるデータが存在しない場合、callerはデータが受信されるまで待機します
TPNOTIME	関数callerがブロッキング・タイムアウトを無視し、応答が受信されるまで継続して待機することを意味します。しかし、トランザクション・タイムアウト内でpb_tprecv()を行った場合は、依然としてブロッキング・タイムアウトが適用されます
TPSIGRSTRT	シグナル割り込み(Interrupt)を受け入れるときに使用します。内部でシグナル割り込みが発生し、システム関数呼び出しに問題があるときにシステム関数呼び出しが再実行されます。このフラグが設定されずにシグナル割り込みが発生した場合は、関数は失敗し(tperrnoにエラーコードが設定されます)、TPGOTSIGが設定されます

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

関数の戻り値が-1の場合には2つのケースあります。一つは、エラーが発生した場合で、もう一つは、イベントが発生した場合です。そのため、-1を返した場合は、どちらのケースに該当するのかを確認する必要があります。

エラーが発生した場合は、tperrnoにPB_TPEEVENT以外のエラーコードが設定され、イベントが発生した場合には、tperrnoにPB_TPEEVENT(数字では22)値が保存され、reventlには下記の値のうち1つが保存されます。

値	説明
TPEV_DISCONIMM (数字 1)	会話開始者が、pb_tpdison()を使用して接続を強制終了したことを意味します。また、通信エラーなどによって接続が終了された場合にもこのイベントが保存されます
TPEV_SVCERR (数字 2)	サービスが異常終了した場合です
TPEV_SVCFAIL (数字 4)	サービスでTPFAILにtpreturnした場合です
TPEV_SVCSUCC (数字 8)	サービスでSUCCESSに返した場合です(正常終了)
TPEV_SENDOONLY (数字 32)	接続された相手プログラムが通信制御権を諦めた場合です。TPEV_SENDOONLYイベントの受信者はデータの送信は可能ですが、受信者が制御権を渡すまではいかなるデータも受信することができません
PB_TPEV_SVCSUCC, PB_TPEV_SVCFAIL	tpurcoodeにサービスから送信したユーザーコードが保存されます

- エラー

エラーコードの詳細内容については、『Tmax リファレンスガイド』の「3.1.84. tprecv」を参照してください。

- 例

```
int ret, cd
string itype, otype, idata, form[], odata[]

itype = "STRING~t~n"
otype = "STRING~t~n"
idata = string(len(before.text)) + "~t" + before.text + "~n"
odata[1] = space(1024)

cd = uo_tmax.pb_tpconnect("CONV", itype, idata, uo_tmax.TPSENDONLY)
if cd < 0 then
    error processing
    ...
end if
```

```

ret = uo_tmax.pb_tprecv(cd, form, odata, uo_tmax.TPNOFLAGS)
if ret < 0 then
    if uo_tmax.revent <> uo_tmax.TPEV_SENDOONLY or &
        uo_tmax.revent = uo_tmax.TPEV_SVCSUCC then
        error processing
    ...
end if
...
end if

```

- 関連関数

pb_tpconnect()、tp_tprecv()、pb_tpdison()

1.2.31. pb_tpreset

pb_tpreset()は現在の接続を即解除する関数です。クライアント・モジュールにTPESYSTEMエラーが発生する場合の大半はネットワーク・エラーであるため、Tmaxシステムに再接続することをお勧めします。この場合、pb_tpreset()を使用して接続を解除してサービスを再要求するか、Tmaxシステムに再接続します。

- プロトタイプ

```

pb_tpreset ( )
returns integer

```

- 戻り値

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

- エラー

エラーコードの詳細内容については、『Tmax リファレンスガイド』を参照してください。

- 例

```

long ret
u_tmax uo_tmax
uo_tmax = CREATE u_tmax

uo_tmax.usr_name = 'tmaxclient'
uo_tmax.clt_name = 'cilent01'
uo_tmax.dom_pwd = 'tmax'

```

```

uo_tmax.usr_pwd = 'client01'
uo_tmax.flags = uo_tmax.TPUNSOL_IGN
...
ret = uo_tmax.pb_tpstart()
if ret < 0 then
    error processing
    ...
end if
...

ret = uo_tmax.pb_tpreset()
if ret < 0 then
    error processing
    ...
endif

```

1.2.32. pb_tpsend

pb_tpsend()は会話型通信において相手にデータを送信するために使用される関数です。callerは、通信制御権を有する必要があります。

● プロトタイプ

```

pb_tpsend ( integer cd, string itype, string idata, long arg_flag )
returns integer

```

● パラメータ

パラメータ	説明
svcname	pb_tpconnect()が返した参照記述子です
itype	使用したバッファ・タイプの文字列型です。タブ("~t")を利用して文字列を区別し、改行("~n")を利用して行のエンドを表示します
idata	処理するデータが保存されるバッファの文字列型です。タブ("~t")を利用して入力データを区別し、改行("~n")を利用して行のエンドを表示します
arg_flag	フラグ値です

pb_tpsend()で使用できるフラグ値は下記のとおりです。

フラグ値	説明
TPRECVONLY	TPRECVONLYフラグはcallerがデータを送信した後、相手に通信制御権を渡すことを意味します。そのためcallerは、通信制御権を再び取得するまでpb_tpsend()を呼び出すことができません。

フラグ値	説明
	会話通信の相手受信者は、pb_tprecv()でデータを受信すると同時に通信制御権を取得するTPEV_SENDOONLYイベントを受信することになります。受信者もまた再び通信制御権を相手に渡すまでpb_tprecv()を呼び出すことができません
TPNOBLOCK	ブロッキング状態で、(例えば、内部バッファが受信されるメッセージでfullになった場合)データとイベントは送信されません。TPNOBLOCKフラグを設定せずにpb_tpsend()を呼び出す場合、ブロッキング状況が発生するとcallerはタイムアウト(トランザクションまたはブロッキング・タイムアウトのうち1つ)が発生するか、状況が緩和されるまで待機します
TPNOTIME	TPNOTIME flagは、関数callerがブロッキング・タイムアウトを無視して応答が受信されるまで待機することです。しかし、トランザクション・タイムアウト内でpb_tpconnect()を使用した場合は、依然としてトランザクション・タイムアウトが適用されます
TPSIGSTRT	シグナル割り込みを受け入れるときに使用します。内部でシグナル割り込みが発生し、システム関数呼び出しに問題があるときにシステム関数呼び出しが再実行されます。このフラグが設定されずにシグナル割り込みが発生した場合は、関数は失敗しtperrnoにTPGOTSIGが設定されます

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

関数の戻り値が-1の場合には2つのケースがあります。一つは、エラーが発生した場合で、もう一つはイベントが発生した場合です。そのため、-1を返した場合はどちらに該当するのかを確認する必要があります。

イベントが発生した場合は、tperrnoにエラーコード(PB_TPEEVEVENT以外の値)が保存され、イベントが発生した場合は、tperrnoにPB_TPEEVEVENT(数字では22)値が保存され、reventには下記の値のうち1つが保存されます。

値	説明
TPEV_DISCONIMM (数字1)	会話開始者がpb_tpdicon()を使用して接続を強制終了した場合です。また、通信エラーなどにより接続が切断された場合にもこのイベントが保存されます
TPEV_SVCFAIL (数字4)	サービスでTPFAILにtpreturnした場合です
TPEV_SVCERR	サービスが異常終了した場合です

値	説明
(数字2)	
PB_TPEV_SVCSUCC, PB_TPEV_SVCFAIL	サービスで送信したユーザーコードがtpurcoodeに保存されます

- エラー

エラーコードの詳細内容については、『*Tmax* リファレンスガイド』の「3.1.88. tpsend」を参照してください。

- 例

```
int ret, cd
string itype, otype, idata, form[], odata[]

itype = "STRING~t~n"
otype = "STRING~t~n"
idata = string(len(before.text)) + "~t" + before.text + "~n"
odata[1] = space(1024)

cd = uo_tmax.pb_tpconnect("CONV", itype, idata, uo_tmax.TPSENDONLY)
if cd < 0 then
    error processing
    ...
end if

ret = uo_tmax.pb_tpsend(cd, itype, idata, uo_tmax.TPRECVONLY)
if ret < 0 then
    error processing
    ...
endif
```

- 関連関数

pb_tpconnect()、tp_tprecv()、pb_tpdicon()

1.2.33. pb_tpset_timeout

pb_tpset_timeout()はサーバーに設定されているサービス制限時間、ブロッキング・タイムアウト時間を変更するときに使用する関数です。サービス制限時間を設定した場合、arg_timeoutの間サービス要求に対する応答を待ちます。arg_timeoutに設定された時間が過ぎても応答が受信できない場合は、タイムアウト・エラーが発生し、要求したサービスの応答を待たずにサービス失敗として返します。

pb_tpset_timeout()はpb_tpset_timeout()が呼び出された以降のサービス要求に適用され、再びpb_tpset_timeout()が呼び出されるか、クライアントやサーバー・プロセスが終了されるまで有効です。

pb_tpset_timeout()が使用されない場合は、ブロッキング・タイムアウトでTmax環境ファイルに設定されたBLOCKTIMEが適用されます。

- プロトタイプ

```
pb_tpset_timeout ( integer arg_timeout )
returns integer
```

- パラメータ

パラメータ	説明
arg_timeout	サービス制限時間です。arg_timeoutの間、サービス要求の応答を待ちます(単位: 秒)

- 戻り値

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

- エラー

エラーコードの詳細内容については、『Tmax リファレンスガイド』の「3.1.90. tpset_timeout」を参照してください。

- 例

```
long ret
long buf
uo_tmax uo_tmax
uo_tmax = CREATE u_tmax
uo_tmax.usr_name = 'tmaxclient'
uo_tmax.clt_name = 'cilent01'
uo_tmax.dom_pwd = 'tmax'
uo_tmax.usr_pwd = 'client01'
uo_tmax.flags = uo_tmax.TPUNSOL_IGN
...
ret = uo_tmax.pb_tpstart()
if ret < 0 then
    error processing
    ...
end if
...
buf = uo_tmax.pb_tpalloc("CARRAY", "", 40000)
...
```



```

ret = uo_tmax.pb_tpset_timeout(5)
if ret < 0 then
    error processing
    ...
end if
...
ret = uo_tmax.pb_tpcall("SYNC", itype, input, form, output)

```

1.2.34. pb_tpsetunsol

非要求受信メッセージを取得するように設定する関数です。受信されたメッセージは、pb_getunsold()を利用して非要求受信メッセージを処理することになります。pb_tpsetunsol()、pb_getunsold()を使用して非要求受信メッセージを取得するには、u_tmaxのインスタンス変数であるflagsにTPUNSOL_HNDで設定した後pb_tpstart()を行う必要があります。

● プロトタイプ

```

pb_tpsetunsol (unsignedinteger hwnd, string ename, unsignedinteger mid,
               long arg_flags)
returns integer

```

● パラメータ

パラメータ	説明
hwnd	ウィンドウハンドルです
ename	イベント名にtppostを使用する場合です。このときデータを受信するには、サービス名と一致する必要があります
mid	メッセージ記述子です。これを利用して自身のメッセージを見つけて転送します
arg_flags	非要求メッセージ・タイプを設定します。設定できる値はTPBROADCAST、TPSENDTOCLIです

● 戻り値

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

● エラー

tpernoにエラーコードが設定されます。

エラーコード	説明
TPEINVAL	入力パラメータが適切でない場合です (例: イベント長が最大値の(16)を超える場合)

- 例

```

...
uo_tmax.flags = uo_tmax.TPUNSOL_HND
...
ret = uo_tmax.pb_tpsetunsol ( HANDLE(this), "", 1025, uo_tmax.TPBROADCAST )
if ret = -1 then
    error processing
    ...
end if
...
unsolddata = space(1024)
ret = uo_tmax.pb_getunsold(unsolddata, rlen, 1025)
if ret < 0 then
    error processing
    ...
end if
...

```

- 関連関数

pb_getunsold()

1.2.35. pb_tpstart

Tmaxとの接続を確立する関数です。この関数を呼び出す前に、Tmaxと接続するための環境変数が指定される必要があります。接続する対象サーバーのIPアドレス(TMAX_HOST_ADDR)、ポート番号(TMAX_HOST_PORT)を指定します。また構造体を使用する場合はSDLFILEのパスを、FIELD(FDL)バッファを使用する場合はFDLFILEのパスを環境変数に登録します。

下記は、環境変数を指定する方法です。

- DOSの場合

<autoexec.bat>を利用します。

SET TMAX_HOST_ADDR=192.168.0.1	サーバーIPアドレス
SET TMAX_HOST_PORT=8888	サーバー・ポート番号
SET SDLFILE=C:\tmax\sdl\tmax.sdl	構造体タイプを使用
SET FDLFILE=C:\tmax\fdl\tmax.fdl	フィールド・タイプを使用

- UNIX(Korn Shell)の場合

<.profile>を利用します。

export TMAX_HOST_ADDR=192.168.0.1	サーバーIPアドレス
export TMAX_HOST_PORT=8888	サーバー・ポート番号
export SDLFILE=C:\tmax\sdl\tmax.sdl	構造体タイプを使用
export FDLFILE=C:\tmax\fdl\tmax.fdl	フィールド・タイプを使用

下記は、関数についての説明です。

- プロトタイプ

```
pb_tpstart() returns integer
```

- u_tmaxのインスタンス変数

変数	説明
usr_name	ユーザー接続usr_name記述子です
clt_name	クライアント接続記述子です
dom_pwd	ドメイン接続暗号です
usr_pwd	ユーザー接続暗号です
flags	非要求メッセージ・タイプとシステム・アクセス方法を指定します – TPUNSOL_IGN：非要求メッセージを無視します – TPUNSOL_POLL：非要求メッセージを受信します(pb_tpgetunsol()で受信) – TPUNSOL_HND：非要求メッセージを受信します

- 戻り値

戻り値	説明
0または1	関数呼び出しに成功した場合です – 0：プライマリー・ホストに接続をトライした場合です – 1：バックアップ・ホストに接続をトライした場合です
-1	関数呼び出しに失敗した場合です。tperrnoにエラーコードが設定されます

- エラー

エラーコードの詳細内容については、『*Tmax* リファレンスガイド』の「3.3.10. tpstart」を参照してください。

- 例

```
long ret
uo_tmax uo_tmax
uo_tmax = CREATE u_tmax
uo_tmax.usr_name = 'tmaxclient'
uo_tmax.clt_name = 'client01'
uo_tmax.dom_pwd = 'tmax'
uo_tmax.usr_pwd = 'client01'
uo_tmax.flags = uo_tmax.TPUNSOL_IGN
...
ret = uo_tmax.pb_tpstart()
if ret < 0 then
    error processing
    ...
end if
...
```

- 関連関数

pb_tmaxreadenv()、tp_init()

1.2.36. pb_tptobackup

クライアントがTmaxシステムに接続する際に使用するpb_tpstart()関数は、接続しようとするサーバーに問題があり接続できない場合、自動的にバックアップ・システムに接続します。pb_tptobackup()は最初からバックアップ・システムに接続するときに使用する関数で、ユーザーが任意でバックアップ・システムに接続しようとする場合に使用します。

pb_tptobackup()はパラメータとしてTPSTART_Tを受けないため、セキュリティが必要なシステムには接続できません。この関数を使用して接続されるバックアップ・システムは、セキュリティに関連する項目を環境ファイルに設定すると接続することができません。tptobackup()を使用するには、TMAX_BACKUP_ADDRとTMAX_BACKUP_PORTをシステム環境ファイル(例: Korn Shellの<.profile>)に登録する必要があります。

- プロトタイプ

```
pb_tptobackup ( )
returns integer
```

- 戻り値

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

- エラー

エラーコードの詳細内容については、『*Tmax* リファレンスガイド』の「3.3.12. tptobackup」を参照してください。

- 例

```
long ret
u_tmax uo_tmax
uo_tmax = CREATE u_tmax

...
ret = uo_tmax.pb_tptobackup()
if ret < 0 then
    error processing
    ...
end if
...
```

1.2.37. pb_tx_begin

グローバル・トランザクションを開始する関数です。関数callerはトランザクション・モードになります。トランザクションが開始されると、callerは現行トランザクションを完了するためにpb_tx_commit()またはpb_tx_rollback()を呼び出します。

- プロトタイプ

```
pb_tx_begin ()
returns integer
```

- 戻り値

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

- エラー

エラーコードの詳細内容については、『*Tmax* リファレンスガイド』の「3.1.104. tx_begin」を参照してください。

- 例

```
...
if _is_tx <> true then
```

```

    ret = uo_tmax.pb_tx_begin()
    if ret < 0 then
        error processing
        ...
    end if
    _is_tx = true
end If
...

```

- 関連関数

pb_tx_commit()、pb_tx_rollback()

1.2.38. pb_tx_commit

トランザクション作業をコミットするときに使用する関数です。

- プロトタイプ

```

pb_tx_commit ( )
returns integer

```

- 戻り値

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

- エラー

エラーコードの詳細内容については、『Tmax リファレンスガイド』の「3.1.105. tx_commit」を参照してください。

- 例

```

...
if _is_tx <> true then
...
end if

ret = uo_tmax.pb_tx_commit()
if ret < 0 then
    error processing
...

```

```

end If

_is_tx = false
...

```

- 関連関数

pb_tx_begin()、pb_tx_rollback()

1.2.39. pb_tx_rollback

トランザクション作業をロールバックするときに使用される関数です。

- プロトタイプ

```

pb_tx_rollback ()
returns integer

```

- 戻り値

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

- エラー

エラーコードの詳細内容については、『Tmax リファレンスガイド』の「3.1.107. tx_rollback」を参照してください。

- 例

```

...
ret = uo_tmax.pb_tpcall("INSERT", itype, input, form[], output[])
if ret < 0 then
    if _is_tx = true then
        ret = uo_tmax.pb_tx_rollback()
        if ret < 0 then
            error processing
            ...
        end if
        _is_tx = false
    end if
    error processing
...

```

```
end if
...
```

- 関連関数

pb_tx_begin()、pb_tx_commit()、pb_tx_info()

1.2.40. pb_tx_set_commit_return

when_return値を使用してグローバル・トランザクションのコミット時点を指定する関数です。

- プロトタイプ

```
pb_tx_set_commit_return ( long w_return )
returns integer
```

- パラメータ

パラメータ	説明
w_return	使用可能な値は下記のとおりです – TX_COMMIT_DECISION_LOGGED: pb_tx_commit()を使用して、2PC(2 Phase Commit)プロトコルでフェーズ1の実行後に返すようにします。短時間で結果値が得られますが、予想外の結果値を取得する恐れがあります – TX_COMMIT_COMPLETED : pb_tx_commit()を使用して、2PC(2 Phase Commit)プロトコルを完全に実行した後、返すようにします

- 戻り値

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

- エラー

エラーコードの詳細内容については、『Tmax リファレンスガイド』の「3.1.108. tx_set_commit_return」を参照してください。

- 例

```
...
if ret = uo_tmax.pb_tx_set_commit_return(uo_tmax.TX_COMMIT_COMPLETED)
```



```

    if ret < 0 then
        error processing
    ...
end if

if _is_tx <> true then
    ret = uo_tmax.pb_tx_begin()
    if ret < 0 then
        error processing
    ...
end if
_is_tx = true
end If
...

```

- 関連関数

pb_tx_info()

1.2.41. pb_tx_set_transaction_control

コントロールに指定された値を利用して、pb_tx_commit()やpb_tx_rollback()が返される前に新しいトランザクションの開始可否を指定する関数です。初期に指定されたcontrol値はTX_UNCHAINEDです。

- プロトタイプ

```

pb_tx_set_transaction_control ( long control )
returns integer

```

- パラメータ

パラメータ	説明
control	<p>使用できる値は下記のとおりです</p> <ul style="list-style-type: none"> – TX_UNCHAINED : 新しいトランザクションを開始しません。この場合、新しいトランザクションを開始するには、必ずpb_tx_begin()を使用します – TX_CHAINED : pb_tx_commit()やpb_tx_rollback()の後、自動的に新しいトランザクションが開始されます

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です

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

- エラー

エラーコードの詳細内容については、『*Tmax* リファレンスガイド』の「3.1.109. tx_set_transaction_control」を参照してください。

- 例

```

...
if ret = uo_tmax.pb_tx_set_transaction_control(uo_tmax.TX_CHAINED)
  if ret < 0 then
    error processing
  ...
end if

if _is_tx <> true then
  ret = uo_tmax.pb_tx_begin()
  if ret < 0 then
    error processing
  ...
end if
_is_tx = true
end If
...

```

- 関連関数

pb_tx_info()

1.2.42. pb_tx_set_transaction_timeout

トランザクションのタイムアウトを設定する関数です。

- プロトタイプ

```

pb_tx_set_transaction_timeout ( long timeout )
returns integer

```

- パラメータ

パラメータ	説明
timeout	時間を設定します

- 戻り値

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

- エラー

エラーコードの詳細内容については、『*Tmax* リファレンスガイド』の「3.1.110. tx_set_transaction_timeout」を参照してください。

- 例

```
...
if ret = uo_tmax.pb_tx_set_transaction_timeout(5)
    if ret < 0 then
        error processing
    ...
    end if

if _is_tx <> true then
    ret = uo_tmax.pb_tx_begin()
    if ret < 0 then
        error processing
    ...
    end if
    _is_tx = true
end if
...
```

- 関連関数

pb_tx_info()

1.2.43. pb_uotmax_ver

pb_uotmax_ver()はtmax.pbdのバージョンを表示する関数です。

- プロトタイプ

```
pb_uotmax_ver ( )
returns string
```

- 戻り値

tmax.pbdのバージョンを返します。

1.2.44. pb_tx_info

グローバル・トランザクション情報が確認できる関数です。トランザクションxidに関する情報をu_tmaxのdata、when_return、transaction_control、transaction_timeout、transaction_statに設定します。

- プロトタイプ

```
pb_tx_info ()  
returns integer
```

- 戻り値

戻り値	説明
1	トランザクション・モードで呼び出された場合です。Tmaxのインスタンス変数値が設定されます
-1	関数呼び出しに失敗した場合です。tperrnoにエラーコードが設定されます

戻り値が1の場合、Tmaxのインスタンス変数値が設定されます。

値	説明
data	現行トランザクションID値で「マシン番号:CLH番号:ブランチ番号」の構成です
when_return	commit_returnの現在設定値です。pb_tx_set_commit_return()で変更されます
transaction_control	トランザクション・コントロールの現在設定値です。pb_tx_set_transaction_control()で変更されます
transaction_timeout	現行トランザクションのタイムアウト値です。pb_tx_set_transaction_timeout()で変更されます
transaction_stat	現行トランザクションの状態値です

- エラー

エラーコードの詳細内容については、『*Tmax リファレンスガイド*』の「3.1.106. tx_info」を参照します。

- 例

```
...  
if _is_tx <> true then  
  ret = uo_tmax.pb_tx_begin()  
  if ret < 0 then  
    error processing  
  ...
```

```

        end if
        _is_tx = true
    end If
    if ret = uo_tmax.pb_tx_info()
        if ret < 0 then
            error processing
            ...
        end if
        ...
    end if

```

- 関連関数

pb_tx_begin()、pb_tx_commit()、pb_tx_rollback()

1.2.45. tp_abort

トランザクション作業を中止するときに使用する関数です。

- プロトタイプ

```
tp_abort ( long arg_flags ) returns integer
```

- パラメータ

パラメータ	説明
arg_flags	現在は使用しません。ユーザーがいかなる値を渡してもエラーが発生しないようにサポートします

- 戻り値

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

- 例

```

...
ret = uo_tmax.pb_tpcall("INSERT", itype, input, form[], output[])
if ret < 0 then
    if _is_tx = true then
        ret = uo_tmax.tp_abort(0)
        if ret < 0 then

```

```

        error processing
    ...
end if
    _is_tx = false
end if
    error processing
    ...
end if
...

```

- 関連関数

tp_begin()、tp_commit()

1.2.46. tp_acall

非同期サービス呼び出しを実行する関数です。サービス呼び出し後に応答を待つ代わりに、応答を受けるときに([1.2.56. tp_getreply](#))を使用する参照記述子を返します。複数のサービスを同時に呼び出した後、その結果を受ける場合や、サービスを呼び出した後、応答を待つ間に他の作業を処理する必要があるときに使用します。

tp_acall()関数は、Tmaxでサポートするバッファ・タイプのうちフィールド・バッファを使用します。

- プロトタイプ

```
tp_acall ( ref string idata ) returns integer
```

- パラメータ

パラメータ	説明
idata	サービス名を含めてデータ文字列を構成します。文字列型で、タブ("~t")を利用して入力データを区別し、改行("~n")を利用して行のエンドを表示します

- 戻り値

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

- 例

```

string input, form[], odata[]
int cd, ret

```

```

...

input = "SRVCNM" + "~t" + "TOUPPER" + "~n" + &
"FIELD_NAME1" + "~t" + "VALUE1" + "~n" + &
"FIELD_NAME2" + "~t" + "VALUE2" + "~n" + &
"~n"

form[1] = "OUTPUT~n~n"
form[2] = "~n"
output[1] = space(1024)

cd = uo_tmax.tp_acall(input)
if cd < 0 then
    error processing
    ...
end if

ret = uo_tmax.tp_getrply(cd, form, output)
if ret < 0 then
    error processing
    ...
end if

```

- 関連関数

pb_tpacall()、tp_getrply()

1.2.47. tp_begin

グローバル・トランザクションを開始する関数です。関数callerはトランザクション・モードになります。トランザクションが開始されると、callerは現行トランザクションを完了するためにtp_commit()またはtp_abort()を呼び出します。

- プロトタイプ

```

tp_begin ( unsignedlong timeout, long arg_flags )
returns integer

```

- パラメータ

パラメータ	説明
timeout	トランザクションが終了される制限時間です。制限時間以内にトランザクションが終了されないと、タイムアウトエラーが発生してトランザクションは中止されます(単位: 秒)
arg_flags	現在使用されません

- 戻り値

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

- 例

```
...
if _is_tx <> true then
    ret = uo_tmax.tp_begin(10, 0)
    if ret < 0 then
        error processing
    ...
    end if
    _is_tx = true
end If
...
```

- 関連関数

tp_commit()、tp_abort()

1.2.48. tp_broadcast

指定したクライアントにメッセージを転送する関数です。

- プロトタイプ

```
tp_broadcast ( string nodename, string username, string cltname, string idata )
returns integer
```

- パラメータ

パラメータ	説明
nodename	メッセージを転送するクライアントが属しているノード名です
username	メッセージを転送するユーザー名です。メッセージを転送するには、pb_tpstart()の場合に設定するusernameと一致する必要があります
cltname	メッセージを転送するクライアント名です。メッセージを転送するには、pb_tpstart()の場合に設定するcltnameと一致する必要があります
idata	サービス名を含めてデータ文字列を構成します。文字列型で、タブ("~t")を利用して入力データを区別し、改行("~n")を利用して行もエンドを表示します

参考

パラメータのnodename、username、cltnameは、すべて対象クライアントを選択するときに使用される論理的な名前です。名前の指定には、ワイルドカード(wildcard)文字が使用できます。usernameとcltnameは、設定される名前に疑問符(?)やアスタリスク(*)などのワイルドカードが指定できます。一方、nodenameは名前の代わりにアスタリスク(*)のみ使用可能です。また、NULL値が使用できますが、これはすべてのクライアントに対応されるワイルドカードで動作します。

● 戻り値

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

● 例

```
string nodename, username, cltname, data
int ret
...
nodename = "tmax"
username = "*"
cltname = "*"
data = "SRVCNM" + "~t" + "BROAD" + "~n" + &
"FIELD_NAME1" + "~t" + "VALUE1" + "~n" + &
"~n"
ret = uo_tmax.tp_broadcast(nodename, username, cltname, data)
if ret < 0 then
    error processing
    ...
end if
...
```

● 関連関数

pb_tpbroadcast()

1.2.49. tp_call

同期型でTmaxサービスを呼び出すための関数です。idataを利用してサービスを呼び出し、返された結果をformに指定された列のデータをodataに格納します。tp_call()がTmaxでサポートするバッファ・タイプのうち、FIELD(FDL)バッファを使用します。

● プロトタイプ

```
tp_call ( string idata, string form[], ref string odata[] )
returns integer
```

● パラメータ

パラメータ	説明
idata	サービス名を含めてデータ文字列を構成します。文字列型で、タブ("~t")を利用して入力データを区別し、改行("~n")を利用して行のエンドを表示します
form	サービス処理結果、odataの構成形式フィールド名は改行("~n")で区別され、最後に改行("~n")を追加してFormのエンドを表示します。最終的なForm配列のエンドを表示するには、改行("~n")で構成されたFormを追加します
odata	サービス処理結果を保存するバッファの各データは、PowerBuilderのImportString関数を使用して簡単にデータをDataWindowに格納できるようにタブ("~t")で区別されており、各行は改行("~n")で区別されています

下記は、Tmaxのインスタンス変数です。flagsとtimeoutはサービス呼び出しの前に設定される値で、以外はサービス呼び出し後に設定される値です。

変数	説明
msg	サービス・プログラムから転送されたメッセージです
tperrno	エラーが発生した場合、エラー番号が保存されるバッファです
tpurcode	サーバーから転送されたユーザーコード(tpreturnの2番目の引数)が保存されるバッファです
flags	<p>pb_tpcallで使用するフラグは下記のとおりです</p> <ul style="list-style-type: none"> – TPNOTRAN : 非トランザクション・モードのサービスで呼び出す場合に使用します – TPNOCHANGE : 初期割り当てバッファと応答バッファ・タイプの一致を保証します – TPNOBLOCK : 非ブロッキング・モードで呼び出す場合に使用します – TPNOTIME : ブロッキング・タイムアウトを無視します <p>[参考]</p> <p>tuxedo_compat(true)設定の場合、フラグの動作が異なります。詳細についてはtuxedo_compat()を参照してください</p>
timeout	pb_tx_set_transaction_timeoutに使用されるブロッキング・タイムアウト値です

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です。tpernoにエラーコードが設定され、msglにエラー・メッセージが保存されます

- 例

```

...
input = "SRVCNM" + "~t" + "TOUPPER" + "~n" + &
"FIELD_NAME1" + "~t" + "VALUE1" + "~n" + &
"FIELD_NAME2" + "~t" + "VALUE2" + "~n" + &
"~n"

form[1] = "OUTPUT~n~n"
form[2] = "~n"
output[1] = space(1024)

ret = uo_tmax. tp_call(input, form, output)
if ret < 0 then
    error processing
    ...
end if
...

```

- 関連関数

pb_tpcall()、f_data()、f_datadel()、f_form()、tuxcall()

1.2.50. tp_cancel

tp_cancel()はtp_acall()を使用して返された記述子を取り消します。したがって、呼び出したサービスに対する受信を取り消すことになります。受信を取り消した場合、当該トランザクションはコミットされず、ロールバックのみ可能です。

- プロトタイプ

```
tp_cancel ( integer cd ) returns integer
```

- パラメータ

パラメータ	説明
cd	tp_acall()が返した参照記述子です

- 戻り値

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

- 例

```
string input, form[], odata[]
int cd, ret
...
input = "SRVCNM" + "~t" + "TOUPPER" + "~n" + &
"FIELD_NAME1" + "~t" + "VALUE1" + "~n" + &
"FIELD_NAME2" + "~t" + "VALUE2" + "~n" + &
"~n"
form[1] = "OUTPUT~n~n"
form[2] = "~n"
output[1] = space(1024)

cd = uo_tmax.tp_acall(input)
if cd < 0 then
    error processing
    ...
end if

ret = uo_tmax.pb_tpcancel(cd)
if ret < 0 then
    error processing
    ...
end if
```

- 関連関数

tp_acall()、tp_getrply()

1.2.51. tp_commit

トランザクション作業をコミットするときに使用する関数です。

- プロトタイプ

```
tp_commit ( long arg_flags ) returns integer
```

- パラメータ

パラメータ	説明
arg_flags	現在は使用されません

- 戻り値

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

- 例

```
...
if _is_tx <> true then
...
end if

ret = uo_tmax.tp_commit(0)
if ret < 0 then
    error processing
    ...
end if

_is_tx = false
...
```

- 関連関数

tp_begin()、tp_abort()

1.2.52. tp_connect

tp_connect()は会話型サービスとの通信を接続する関数です。関数を使用して接続されたサーバーとクライアントは、相互にtp_send()、tp_recv()を使用してデータをやり取りします。tp_connect()関数は、Tmaxでサポートするバッファ・タイプのうち、フィールド・バッファを使用します。

- プロトタイプ

```
tp_connect ( string idata, long arg_flag )
returns integer
```

- パラメータ

パラメータ	説明
idata	サービス名を含めてデータ文字列を構成します。文字列型で、タブ("~t")を利用して入力データを区別し、改行("~n")を利用して行のエンドを表示します
arg_flag	フラグ値です

tp_connect()で利用できるフラグは下記のとおりです。

フラグ値	説明
TPNOTRAN	<p>pb_tpconnect()のcallerがトランザクション・モード状態でこのフラグを設定してsvcサービスを要求した場合、svcサービスはトランザクション・モードから除外されて実行されます。トランザクション・モードでsvcがトランザクションをサポートしないサービスである場合は、pb_tpconnect()がトランザクション・モードで呼び出される際に、フラグはTPNOTRANで設定される必要があります。</p> <p>トランザクション・モード内でpb_tpconnect()を呼び出すとき、TPNOTRANで設定された場合でも依然としてトランザクション・タイムアウトの影響を受けます。TPNOTRANで呼び出されたサービスが失敗した場合、callerのトランザクションには影響を与えません</p>
TPSENDONLY	接続が完了された後、関数callerは最初のデータのみ送信可能で、要求されたサービスのみ行えるように設定するフラグです。callerが最初の通信制御権を有します。TPSENDONLYやTPRECVONLYのうち1つを必ず指定します
TPRECVONLY	接続が完了された後、関数callerはデータのみ受信可能で、要求されたサービスが最初のデータ送信を開始するようにするフラグです。要求されたサービスが最初に通信制御権を有します。TPSENDONLYやTPRECVONLYのうち1つを必ず指定します
TPNOTIME	TPNOTIMEフラグは関数callerがブロッキング・タイムアウトを無視し、応答が受信されるまで継続して待機することを意味します。しかし、トランザクション・タイムアウト内で、pb_tpconnect()を使用した場合は、依然としてトランザクション・タイムアウトが適用されます
TPSIGSTR	シグナル割り込みを受用するときに使用します。内部でシグナル割り込みが発生して、システム関数呼び出しに問題があるときに関数呼び出しが再実行されます。フラグが設定されず、シグナル割り込みが発生した場合は、関数は失敗し、tperrnoにTPGOTSIGが設定されます

- 戻り値

戻り値	説明
参照される記述子	関数呼び出しに成功した場合です

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

- 例

```
int ret, cd
string idata

idata = "SRVCNM" + "~t" + "CONV" + "~n" + &
"FIELD_NAME1" + "~t" + "VALUE1" + "~n" + &
"FIELD_NAME2" + "~t" + "VALUE2" + "~n" + &
"~n"

cd = uo_tmax.tp_connect(idata, uo_tmax.TPRECVONLY)
if cd < 0 then
    error processing
    ...
end if
...
```

- 関連関数

tp_send(), tp_rcv(), tp_conv(), tp_discon()

1.2.53. tp_conv

tp_conv()はcdによって指定された会話型サービスのメッセージを送受信する関数です。Tmaxでサポートするバッファ・タイプのうち、フィールド・バッファを使用します。

- プロトタイプ

```
tp_conv ( integer cd, string idata, ref string form[], ref string odata[],
          long arg_flag )
returns integer
```

- パラメータ

パラメータ	説明
cd	tp_connect()が返した参照記述子です
idata	サービス名を含めてデータ文字列を構成します。文字列型で、タブ("~t")を利用して入力データを区別し、改行("~n")を利用して行のエンドを表示します

パラメータ	説明
form	サービス処理結果、odataの構成形式フィールド名は改行("~n")で区別され、最後に改行("~n")を追加してFormのエンドを表示します。最終的なForm配列のエンドを表示するには、改行("~n")で構成されたFormを追加します
odata	サービス処理結果を保存するバッファの各データは、PowerBuilderのImportString関数を使用して簡単にデータをDataWindowに格納できるようにタブ("~t")で区別されており、各行は改行("~n")で区別されています
arg_flag	pb_tpsend()、pb_tprecv()で使用したフラグ値はすべて使用可能です

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

関数の戻り値が-1の場合には2つのケースがあります。一つは、エラーが発生した場合で、もう一つは、イベントが発生した場合です。-1を返した場合は、どちらに該当するかを確認する必要があります。

エラーが発生した場合は、tperrnoにエラーコード(PB_TPEEevent以外の値)が保存され、イベントが発生した場合は、tperrnoにPB_TPEEevent(数字では22)値が保存され、reventlには下記の値のうち1つが保存されます。

値	説明
TPEV_DISCONIMM (数字1)	会話開始者がpb_tpdicon()を使用して接続を強制終了した場合です。また、通信エラーなどにより接続が終了された場合もこのイベントが保存されます
TPEV_SVCERR (数字2)	サービス関数が異常終了した場合です
TPEV_SVCFAIL (数字4)	サービスでTPFAILにtpreturnした場合です
TPEV_SENDOONLY (数字32)	接続された相手のプログラムで通信制御権を諦めた場合です。TPEV_SENDOONLYイベントの受信者はデータの送信は可能ですが、受信者が制御権を渡すまではいかなるデータの受信もできません
PB_TPEV_SVCSUCC、 PB_TPEV_SVCFAIL	tpurcoodeにサービスから送信されたユーザーコードが保存されます

- エラー

エラーコードの詳細内容については、『*Tmax リファレンスガイド*』の「3.1.41. tpacall」を参照してください。

- 例

```
int ret, cd
string idata

idata = "SRVCNM" + "~t" + "CONV" + "~n" + &
"FIELD_NAME1" + "~t" + "VALUE1" + "~n" + &
"FIELD_NAME2" + "~t" + "VALUE2" + "~n" + &
"~n"
form[1] = "OUTPUT~n~n"
form[2] = "~n"
output[1] = space(1024)

cd = uo_tmax.tp_connect(idata, uo_tmax.TPRECVONLY)
if cd < 0 then
    error processing
    ...
end if

ret = uo_tmax.tp_conv(cd, idata, form, output, uo_tmax.TPNOFLAGS)
if ret < 0 then
    if uo_tmax.revent <> uo_tmax.TPEV_SENDOONLY or &
        uo_tmax.revent = uo_tmax.TPEV_SVCSUCC then
        error processing
        ...
    end if
    ...
end if
...
```

- 関連関数

tp_connect()、tp_discon()、tp_send()、tp_recv()

1.2.54. tp_discon

tp_discon()は会話の接続を強制終了し、TPEV_DISCONIMMイベントを発生させる関数です。

会話型サービスを正常終了するには、サービスでtpreturnを実行します。会話型サービスを接続したクライアントで、必要に応じて接続を即終了させるときに使用します。関数を呼び出すと、その後サービスから送信されるデータは消失され、当該トランザクションは中止のみ可能です。

- プロトタイプ

```
tp_discon ( integer cd ) returns integer
```

- パラメータ

パラメータ	説明
cd	tp_connect()が返した参照記述子です

- 戻り値

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

- 例

```
int ret, cd
string idata

idata = "SRVCNM" + "~t" + "CONV" + "~n" + &
"FIELD_NAME1" + "~t" + "VALUE1" + "~n" + &
"FIELD_NAME2" + "~t" + "VALUE2" + "~n" + &
"~n"

cd = uo_tmax.tp_connect(idata, uo_tmax.TPRECVONLY)
if cd < 0 then
    error processing
    ...
end if

ret = uo_tmax.tp_discon(cd)
if ret < 0 then
    error processing
    ...
end if

...
```

- 関連関数

pb_tpconnect()、tp_tpsend()、pb_tprecv()

1.2.55. tp_fcalle

同期型でTmaxサービス呼び出すための関数です。拡張した形式で、メモリーやファイルから入力データを受け入れ、結果データをメモリーやファイルに同時に保存できます。Tmaxでサポートするバッファ・タイプのうち、FIELD(FDL)バッファを使用します。結果データはformに、指定されたフィールドのデータはodataに保存します。また、fformに指定されたフィールドのデータはfoutdataのファイルに保存されます。

● プロトタイプ

```
tp_fcalle ( string idata, string form[], ref string odata[], string ifilename,
            string fform[], ref string foutdata[] )
returns integer
```

● パラメータ

パラメータ	説明
idata	サービス名を含めてデータ文字列を構成します。文字列型で、タブ("\t")を利用して入力データを区別し、改行("\n")を利用して行のエンドを表示します
form	サービス処理結果、odataの構成形式フィールド名は改行("\n")で区別され、最後に改行("\n")を追加してFormのエンドを表示します。最終的なForm配列のエンドを表示するには、改行("\n")で構成されたFormを追加します
odata	サービス処理結果を保存するバッファの各データは、PowerBuilderのImportString関数を使用して簡単にデータをDataWindowに格納できるようにタブ("\t")で区別されており、各行は改行("\n")で区別されています
ifilename	入力するデータが保存されているファイル名ファイルの形式は、フィールド名とデータをタブ("\t")で区別し、データのエンズを示す改行("\n")を追加します
fform	サービス処理結果、fodataの構成形式フィールド名は改行("\n")で区別され、最後に改行("\n")を追加してFormをエンドを示します。最終的なForm配列のエンドを表示するには、改行("\n")で構成されたFormを追加します
foutdata	サービス処理結果を保存するファイル名です

● 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です。tperrnoにエラーコードが設定され、msgにエラー・メッセージが保存されます

● 例

```
...
InputFile = "c:\temp\fddata0.txt"
```

```

InputFile = FileOpen(InputFile, StreamMode!, Write!, LockReadWrite!, Replace!)
...
uo_tmax.f_fdata(dw_input, InputFile)

if FileClose(InputFile) <> 1 then
    error processing
    ...
end if
...
ret = uo_tmax.tp_fcall(input, form, output, inputfilename, fform, foutput)
if ret = -1 then
    error processing
    ...
end if
...

```

- 関連関数

pb_tpfcall()、f_data()、f_datadel()、f_form()

1.2.56. tp_getrply

非同期型通信でデータを受信する関数です。tp_acall()によって送信された要求の応答を受けるために使用されます。Tmaxでサポートするバッファ・タイプのうち、フィールド・バッファを使用します。formおよびoutputはtp_call()の構成パラメータと同様です。

- プロトタイプ

```

tp_getrply ( integer cd, ref string form[], ref string odata[] )
returns integer

```

- パラメータ

パラメータ	説明
cd	tp_acall()が返した参照記述子です。要求に対する応答が識別できるようにします
form	サービス処理結果、odataの構成形式フィールド名は改行("~n")で区別され、最後に改行("~n")を追加してFormのエンドを表示します。最終的なForm配列のエンドを表示するには、改行("~n")で構成されたFormを追加します
odata	サービス処理結果を保存するバッファの各データは、PowerBuilderのImportString関数を使用して簡単にデータをDataWindowに格納できるようにタブ("~t")で区別されており、各行は改行("~n")で区別されています

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です。tperrnoにエラーコードが設定され、msgにエラー・メッセージが保存されます

- 例

```

string input, form[], odata[]
int cd, ret
...
input = "SRVCNM" + "~t" + "TOUPPER" + "~n" + &
"FIELD_NAME1" + "~t" + "VALUE1" + "~n" + &
"FIELD_NAME2" + "~t" + "VALUE2" + "~n" + &
"~n"
form[1] = "OUTPUT~n~n"
form[2] = "~n"
output[1] = space(1024)

cd = uo_tmax.tp_acall(input)
if cd < 0 then
    error processing
    ...
end if

ret = uo_tmax.tp_getrply(cd, form, output)
if ret < 0 then
    error processing
    ...
end if

```

- 関連関数

pb_tpgetrply()、tp_acall()

1.2.57. tp_init

Tmaxとの接続を確立する関数です。pb_tpstart()のような環境変数の設定が必要です。

- プロトタイプ

```

tp_init ()
returns integer

```

- 戻り値

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

- 例

```

long ret
uo_tmax uo_tmax
uo_tmax = CREATE u_tmax

uo_tmax.usr_name = 'tmaxclient'
uo_tmax.clt_name = 'client01'
uo_tmax.dom_pwd = 'tmax'
uo_tmax.usr_pwd = 'client01'
uo_tmax.flags = uo_tmax.TPUNSOL_IGN
...
ret = uo_tmax.tp_initt()
if ret < 0 then
    error processing
    ...
end if
...

```

- 関連関数

pb_tpstart()

1.2.58. tp_recv

tp_recv()は会話型通信でデータを受信する関数で、呼び出し元が制御権を持たないプログラムでのみ使用できます。tp_recv()はTmaxでサポートするバッファ・タイプのうち、フィールド・バッファを使用します。

- プロトタイプ

```

tp_recv ( integer cd, ref string form[], ref string odata[], long arg_flag )
returns integer

```

- パラメータ

パラメータ	説明
cd	pb_tpconnect()が返した参照記述子です

パラメータ	説明
form	サービス処理結果、odataの構成形式フィールド名は改行("~n")で区別され、最後に改行("~n")を追加してFormのエンドを表示します。最終的なForm配列のエンドを表示するには、改行("~n")で構成されたFormを追加します
odata	サービス処理結果を保存するバッファの各データは、PowerBuilderのImportString関数を使用して簡単にデータをDataWindowに格納できるようにタブ("~t")で区別されており、各行は改行("~n")で区別されています
arg_flag	フラグ値です

tp_recv()で使用するフラグ値は下記のとおりです。

フラグ値	説明
TPNOBLOCK	データが受信されるまで待機しません。受信可能なデータが存在する場合はこれを返します。フラグが指定されず受信可能なデータが存在しない場合、callerはデータが受信されるまで待機します
TPNOTIME	関数callerがブロッキング・タイムアウトを無視し、応答が受信されるまで継続して待機する場合です。しかし、トランザクション・タイムアウト内でpb_tprecv()を使用した場合は、依然としてトランザクション・タイムアウトが適用されます
TPSIGRSTRT	シグナル割り込みを受用するときに使用します。内部でシグナル割り込みが発生して、システム関数呼び出しに問題があるときに関数呼び出しが再実行されます。フラグが設定されずシグナル割り込みが発生した場合、関数は失敗し、tperrnoにTPGOTSIGが設定されます

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

関数の戻り値が-1の場合には2つのケースがあります。一つはエラーが発生した場合で、もう一つはイベントが発生した場合です。-1を返した場合は、どちらに該当するのかを確認する必要があります。

エラーが発生した場合は、tperrnoにPB_TPEEVENT以外のエラーコードが保存され、イベントが発生した場合は、tperrnoにPB_TPEEVENT(数字では22)値が保存され、reventには下記の値のうち1つが保存されます。

値	説明
TPEV_DISCONIMM (数字1)	会話開始者がpb_tpdicon()を使用して接続を強制終了した場合です。また、通信エラーなどにより接続が終了された場合もこのイベントが保存されます
TPEV_SVCERR	サービス関数が異常終了した場合です

値	説明
(数字2)	
TPEV_SVCFAIL (数字4)	サービスでTPFAILにtpreturnした場合です
TPEV_SVCSUCC (数字8)	サービスでSUCCESSに返した場合です(正常終了)
TPEV_SENDOONLY (数字32)	接続された相手のプログラムで通信制御権を諦めた場合です。 TPEV_SENDOONLYイベントの受信者はデータの送信は可能ですが、受信者が制御権を渡すまではいかなるデータの受信もできません
PB_TPEV_SVCSUCC、 PB_TPEV_SVCFAIL	tpurcoodeにサービスで送信したユーザーコードが保存されます

- 例

```

int ret, cd
string itype, otype, idata, form[], odata[]

idata = "SRVCNM" + "~t" + "CONV" + "~n" + &
"FIELD_NAME1" + "~t" + "VALUE1" + "~n" + &
"FIELD_NAME2" + "~t" + "VALUE2" + "~n" + &
"~n"

cd = uo_tmax.tp_connect(idata, uo_tmax.TPRECVONLY)
if cd < 0 then
    error processing
    ...
end if

ret = uo_tmax.tp_rcv(cd, form, odata, uo_tmax.TPNOFLAGS)
if ret < 0 then
    if uo_tmax.revent <> uo_tmax.TPEV_SENDOONLY or &
        uo_tmax.revent = uo_tmax.TPEV_SVCSUCC then
        error processing
        ...
    end if
    ...
end if

```

- 関連関数

tp_connect()、tp_discon()、tp_send()、tp_conv()

1.2.59. tp_send

会話型通信で相手にデータを送信する関数です。callerは通信制御権を有する必要があります。tp_send()はTmaxでサポートするバッファ・タイプのうち、フィールド・バッファを使用します。

● プロトタイプ

```
tp_send ( integer cd, string idata, long arg_flag )  
returns integer
```

● パラメータ

パラメータ	説明
cd	pb_tpconnect()が返した参照記述子です
idata	サービス名を含めてデータ文字列を構成します。文字列型で、タブ("~t")を利用して入力データを区別し、改行("~n")を利用して行のエンドを表示します
arg_flag	フラグ値です

tp_send()で利用できるフラグ値は下記のとおりです。

フラグ値	説明
TPRECVONLY	TPRECVONLYフラグはcallerがデータを送信した後、相手に通信制御権を渡すことを意味します。そのため、callerは通信制御権を再取得するまでpb_tpsend()を呼び出すことができません。会話通信の相手受信者は、pb_tprecv()でデータを受信すると同時に通信制御権を取得するTPEV_SENDOONLYイベントを受信することになります。受信者もまた再び通信制御権を相手に渡すまでpb_tprecv()を呼び出すことができません
TPNOBLOCK	ブロッキング状態で、(例えば、内部バッファが受信されるメッセージでfullになった場合)データとイベントは送信されません。TPNOBLOCKフラグを設定せずにpb_tpsend()を呼び出す場合、ブロッキング状況が発生するとcallerはタイムアウト(トランザクションまたはブロッキング・タイムアウトのうち1つ)が発生するか、状況が緩和されるまで待機します
TPNOTIME	TPNOTIMEフラグは関数callerがブロッキング・タイムアウトを無視し、応答が受信されるまで継続して待機する場合です。しかし、トランザクション・タイムアウト内でpb_tpconnect()を使用した場合は、依然としてトランザクション・タイムアウトが適用されます
TPSIGSTRT	シグナル割り込みを受用するときに使用します。内部でシグナル割り込みが発生して、システム関数呼び出しに問題があるときに関数呼び出しが再実行されます。フラ

フラグ値	説明
	グが設定されず、シグナル割り込みが発生した場合は、関数は失敗し、tperrnoにTPGOTSIGが設定されます

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

関数の戻り値が-1の場合には2つのケースがあります。1つはイベントが発生した場合で、もう1つはエラーが発生した場合です。そのため、-1を返した場合はどちらに該当するのかを確認する必要があります。

イベントが発生した場合は、tperrnoにPB_TPEEVENT(数字では22)値が保存され、reventには下記の値のうち1つが保存されます。

値	説明
TPEV_DISCONIMM (数字1)	会話開始者がpb_tpdiscn()を使用して接続を強制終了した場合です。また、通信エラーなどにより接続が切断された場合にもこのイベントが保存されます
TPEV_SVCERR (数字2)	サービスが異常終了した場合です
TPEV_SVCFAIL (数字4)	サービスでTPFAILにtpreturnした場合です

- 例

```
int ret, cd
string itype, otype, idata, form[], odata[]

idata = "SRVCNM" + "~t" + "CONV" + "~n" + &
"FIELD_NAME1" + "~t" + "VALUE1" + "~n" + &
"FIELD_NAME2" + "~t" + "VALUE2" + "~n" + &
"~n"

cd = uo_tmax.tp_connect(idata, uo_tmax.TPSENDONLY)
if cd < 0 then
    error processing
    ...
end if
```

```

ret = uo_tmax.tp_send(idata, uo_tmax.TPRECVONLY)
if ret < 0 then
    error processing
    ...
end if
...

```

- 関連関数

tp_connect()、tp_discon()、tp_rcv()、tp_conv()

1.2.60. tp_subscribe

サーバーでtpost()によって渡されるメッセージを取得するための関数です。event_nameに指定された名前を有するメッセージがtpost()された場合、hwndで指定したウィンドウにmessage_idで指定したウィンドウ・イベントが発生するように設定します。メッセージが受信されたとき、当該ウィンドウのウィンドウ・イベント処理ルーチンにより、Message.WordParmにメッセージのIDが渡されます。この値は、get_evtname()またはget_data()でメッセージを識別するIDとして使用されます。設定を解除するには、tp_unsubscribe()を使用します。

- プロトタイプ

```

tp_subscribe (unsignedinteger hwnd, string event_name,
              unsignedinteger message_id)
returns integer

```

- パラメータ

パラメータ	説明
hwnd	ウィンドウハンドルです
event_name	イベント名を設定します
message_id	メッセージIDです

- 戻り値

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

- 関連関数

tp_unsubscribe()

1.2.61. tp_term

Tmaxシステムとの接続を解除する関数です。

- プロトタイプ

```
tp_term ( )  
returns integer
```

- 戻り値

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

- 例

```
long ret  
u_tmax uo_tmax  
uo_tmax = CREATE u_tmax  
...  
ret = uo_tmax.tp_term()  
if ret < 0 then  
    error processing  
    ...  
end if
```

- 関連関数

pb_tpend()

1.2.62. tp_unsubscrib

ウィンドウ(hwnd)でメッセージを取得するように指定されたのを、tp_subscribe()、tp_setunsol()を使用して解除する関数です。

- プロトタイプ

```
tp_unsubscribe ( unsignedinteger hwnd )  
returns integer
```

- パラメータ

パラメータ	説明
hwnd	ウィンドウハンドルです

- 戻り値

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

- 関連関数

tp_subscribe()

1.2.63. tpbegin

pb_tpbegin()は、pb_tx_set_transaction_timeout()とpb_tx_begin()の機能が同時に実行できる関数です。Tuxedoで使用する関数と同様な形式であるため、Tuxedo用で開発されたアプリケーションを変換せずに使用することができます。

- プロトタイプ

```
pb_tpbegin ( unsignedlong arg_timeout )
returns integer
```

- パラメータ

パラメータ	説明
arg_timeout	pb_tx_set_transaction_timeout()に設定する値と同様な意味です。トランザクション・タイムアウト時間を入力します(単位: 秒)

- 戻り値

「[1.2.42. pb_tx_set_transaction_timeout](#)」と「[1.2.37. pb_tx_begin](#)」を参照してください。

- 例

```
long ret
u_tmax uo_tmax
uo_tmax = CREATE u_tmax

uo_tmax.usr_name = 'tmaxclient'
```

```

uo_tmax.clt_name = 'cilent01'
uo_tmax.dom_pwd = 'tmax'
uo_tmax.usr_pwd = 'client01'
uo_tmax.flags = uo_tmax.TPUNSOL_IGN
...
ret = uo_tmax.pb_tpstart()
if ret < 0 then
    error processing
    ...
end if
...
ret = uo_tmax.pb_tpbegin(7)

```

- 関連関数

pb_tx_set_transaction_timeout()、pb_tx_begin()

1.2.64. tuxcall

同期型のTmaxサービスを呼び出すための関数です。subjectに指定されたPowerObjectを入力してサービス呼び出した後、返された結果をobjectに指定されたPowerObjectに格納します。

tuxcall()はTmaxでサポートするバッファ・タイプのうち、フィールド・バッファを使用します。subjectで利用できるオブジェクトは下記のとおりです。

オブジェクト	説明
DataWindow, DataWindowChild, DataStore	列名をフィールド・バッファのフィールド名で使用します。修正された行のみデータとして使用します
SingleLineEdit, MultiLineEdit	Control.Tag名をフィールド・バッファのフィールド名で使用します。Control.Textをデータとして使用します
taticText, EditMasks	Control.Tag名をフィールド・バッファのフィールド名で使用します。Control.Textをデータとして使用します
ListBox	Control.Tag名をフィールド・バッファのフィールド名で使用します。選択されたすべての項目をデータとして使用します
DropDownListBox	Control.Tag名をフィールド・バッファのフィールド名で使用します。選択された項目のみデータとして使用します
CheckBox	Control.Tag名をフィールド・バッファのフィールド名で使用します。チェックされている場合のみデータとして使用します

オブジェクト	説明
RadioButton	Control.Tag名をフィールド・バッファのフィールド名で使⽤します。チェックされている場合のみデータとして使⽤します

robjectに使用できるオブジェクトは、インプットで使用可能なオブジェクトと同様です。ListBoxとDropDownListBoxの場合はデータを追加します。使用されるu_tmaxのインスタンス変数は下記のとおりです

変数	説明
dwAppend	TRUEの場合、robjectにDataWindowが指定されると、DataWindowに存在していた既存のデータを削除せず継続して追加します(デフォルト値: FALSE)
dwDelBuf	TRUEの場合、subjectにDataWindowが指定されると、DataWindowのDeleteBufferとPrimaryBufferからデータを抽出します(デフォルト値: TRUE)
DataOnFail	TRUEの場合、サービスが失敗してもデータをウィンドウに出力します (デフォルト値: FALSE)
AuxData	サービスを呼び出すたびに自動で追加されるデータをtp_call()のinput形式で指定します
DateFmt, TimeFmt, DateTimeFmt	Date/Time/DateTime型の列値の場合、String型に変換する際に参照されるフォーマットを指定します

参考

*tuxcall()は内部的にtp_call()を使用するため、上記の変数以外に[tp_call\(\)](#)で使用される変数も参照してください。

● プロトタイプ

```
tuxcall ( string svcname, powerobject subject[], ref powerobject robject[] )
returns integer
```

● 戻り値

戻り値	説明
1	関数呼び出しに成功した場合は
-1	関数呼び出しに失敗した場合は。tperrnoにエラーコードが設定され、msgにエラー・メッセージが保存されます

● 例

```

long ret
windowobject indata[], outdata[]
indata[1] = dw_1
indata[2] = sle_input
outdata[1] = dw_2
uo_tmax.flags = 0;

ret = uo_tmax.tuxcall("SVC1", indata, outdata)
if ret < 0 then
    error processing
    ...
end if

```

- 関連関数

tp_call(), pb_tpcallw()

1.2.65. tuxedo_compat

tuxedo_compat()は一部のAPIの動作に対してTuxedoライブラリーとの互換性を提供します。環境変数 PB_TUXCOMPATの設定値を「Y」に指定した場合も同様に動作します。変更機能は以下のとおりです。

- isBlocked()

TPNOBLOCKフラグを設定すると、サービス要求(tp_call関数などの呼び出し)中に他の作業を実行できます。isBlocked()を呼び出すと、サービス要求が実行中の状態であるか確認できます。

- msg変数

サービスでリターン・データのSTATLINフィールドに値を設定すると、msgインスタンス変数にその内容が設定されます。もしサービスの呼び出しに失敗し、STATLINフィールドに値がない場合は、msgに tpstrerror(tperrno)情報が保存されます。

- tp_fcall(), tp_fcall32()

ファイル名を入力しなくてもTPEINVALエラーが発生しなくなります。

- flags値の機能の変化

tuxcall(), tuxfcall32(), tp_call(), tp_call32(), tp_fcall(), tp_fcall32(), pb_etpcall() APIでフラグの設定によって動作が異なります。

フラグ値	説明
TPNOTRAN	内部でtx_begin、tx_commitを実行しません。このフラグを設定しない場合、APIを呼び出すたびに、呼び出しの前後にtx_begin()、tx_commit()を呼び出します

フラグ値	説明
TPNOBLOCK	<p>内部でtpacall()を使用し、tpgetrply()を呼び出すたびにyield()を呼び出します。</p> <p>サービス要求を処理する間、ユーザー・インターフェースはブロッキングされずに他の作業を処理できます。応答が返されるまで内部では、毎回1000マイクロ秒の間tpusleep()を実行した後、yield()を呼び出します。tpusleepの時間を変更するには、tuxedo_noblock_usleepインスタンス変数の値をマイクロ秒単位で再定義します</p>

● プロトタイプ

```
tuxedo_compat(boolean sw)
returns boolean
```

● パラメータ

パラメータ	説明
sw	trueに設定すると、互換性を提供します

● 戻り値

戻り値	説明
true	関数呼び出しに成功した場合です
false	関数呼び出しに失敗した場合です

● 例

```
boolean ret
u_tmax uo_tmax
uo_tmax = CREATE u_tmax
...
ret = uo_tmax.tuxreadenv("tmax.env", "TMAX")
if ret = false then
    error processing
end if

ret = uo_tmax.tuxedo_compat(true)
if ret = false then
    error processing
end if

uo_tmax.tuxedo_noblock_usleep = 500
```

- 関連関数

pb_tmaxreadenv(), tuxcall(), tuxfcall32(), tp_call(), tp_call32(), tp_fcall(), tp_fcall32(), pb_etpcall(), isBlocked()

1.2.66. tuxreadenv

tuxreadenv()は、ファイルに保存された接続するシステムの情報を読み込み、環境変数に新しい値を設定する関数です。Tmaxシステムと接続するための情報を環境変数に設定するため、この関数はTmaxシステムに接続する前に実行される必要があります。登録された環境変数を参照してpb_tpstart()(またはtp_init())を使用してTmaxシステムと接続します。通常、環境変数はオペレーティング・システムによって定義するファイルが異なります。UNIXの場合、cshは<.cshrc>に、kshは<.profile>に定義します。DOSの場合は、<autoexec.bat>ファイルに定義します。しかし、接続するシステムが2つ以上の場合、クライアントは状況に応じて接続しようとするシステムを変更することができます。この場合は、環境変数に2つのシステムに関するシステム情報が登録できないため、ファイルに環境変数を登録して使用します。

- プロトタイプ

```
tuxreadenv ( string env_file, string section )
returns integer
```

- パラメータ

パラメータ	説明
env_file	接続するシステムの環境情報が保存されたファイル名です。このファイルはテキスト型で、一定した形式に合わせて登録されている必要があります
section	ファイル内に登録された環境情報の記述子です。2つ以上のシステム情報を1つのファイルに登録する場合に、各システムが識別できる値です

- 戻り値

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

- 例

```
long ret
u_tmax uo_tmax
uo_tmax = CREATE u_tmax
...
ret = uo_tmax.tuxreadenv("./tmax.env", "TMAX")
if ret < 0 then
```

```
error processing
end if
```

- 関連関数

pb_tmaxreadenv()

1.3. サンプル・プログラム

社員番号をキー値として、名前、役職、担当役員、入社日、給料、契約期間、部署などを照会、修正、削除、入力するプログラムです。使用されるバッファはフィールド・キー・バッファです。

1.3.1. プログラムの構成

プログラムは下記のように構成されています。

- 共通プログラム

プログラムファイル	説明
demo.f	フィールド・キー・バッファを定義したファイルです
tmconfig.m	Tmax環境設定ファイルです

- クライアント・プログラム

プログラムファイル	説明
demo.pbl	クライアント・プログラムです

- サーバー・プログラム

プログラムファイル	説明
emp_c.pc	サービス・プログラム(Oracleソース)です
emp_c.mk	メイクファイルです

1.3.2. プログラムの特徴

下記は各プログラムの特徴です。

- クライアント・プログラム

機能	説明
User Object接続	tmax.pbdをアプリケーションにリンクします
バッファ・タイプ	フィールド・キー・バッファ、フィールド・キー・ファイルをfdlcユーティリティーでコンパイルして「fdl」ファイルを作成する必要があります
通信タイプ	pb_tpcall()を利用して同期通信を行います
トランザクションの有無	照会、修正、削除、入力する場合、すべてトランザクション処理します
Tmax接続	サービスを実行するたびに接続し、サービスが完了したら接続を解除します

● サーバー・プログラム

機能	説明
サービス	FDLSELECT、FDLUPDATE、FDLDELETE、FDLINSERTを作成します
データベース指定	Oracleデータベースを使用します。システム構成ファイルのSVRGROUPに、データベース情報を指定します

1.3.3. 共通プログラム

データベースのEMP表

下記はDBを運用するための基本表の例です。

EMPNO	NUMBER	NOT NULL	P1
ENAME	VARCHAR(16)		
JOB	VARCHAR(16)		
MGR	NUMBER		
HIREDATE	DATE		
SAL	NUMBER(7,2)		
COMM	NUMBER(7,2)		
DEPTNO	NUMBER		

フィールド・キー・バッファの定義ファイル

下記は、フィールド・キー・バッファを定義したファイルの例です。

<demo.f>

#For tmax demo employee program				
EMPNO	7500	long	-	-
ENAME	7501	string	-	-
JOB	7502	string	-	-

MGR	7503	long	-	-
DATE	7504	string	-	-
SAL	7505	float	-	-
COMM	7506	float	-	-
DEPTNO	7507	long	-	-
E_TYPE	9009	long	-	-
E_CODE	9010	long	-	-
E_MSG	9011	string	-	-
E_TMP	9012	long	-	-

Tmax環境設定

下記はTmax環境設定ファイルの例です。

<tmconfig.m>

```
*DOMAIN
dom1      SHMKEY = 70000,
          MAXUSER = 200,
          MINCLH = 1,
          MAXCLH = 5,
          TPORTNO = 8888,
          BLOCKTIME = 200,
          TXTIME = 200

*NODE
tmax1     TMAXDIR = "/home/tmax",
          APPDIR = "/home/tmax/appbin",
          PATHDIR = "/home/tmax/path",
          TLOGDIR = "/home/tmax/log/tlog",
          SLOGDIR = "/home/tmax/log/slog",
          ULOGDIR = "/home/tmax/log/ulog"

*SVRGROUP
svg1      NODENAME = tmax1,
          DBNAME = ORACLE,
          OPENINFO = "ORACLE_XA+Acc=P/scott/tiger+SesTm=60",
          TMSNAME = svg1_tms

*SERVER
emp_c     SVGNAME = svg1, MIN = 1

*SERVICE
FDLSELECT SVRNAME = emp_c
FDLUPDATE SVRNAME = emp_c
```

```
FDLDELETE    SVRNAME = emp_c
FDLINSERT    SVRNAME = emp_c
```

1.3.4. クライアント・プログラム

下記は、プログラムを実行するときに表示される初期画面です。画面は8つの単一行エディター、1つのデータ画面、5つのボタンで構成されます。

社員番号 sle_empno
名前 sle_name

社員情報

役職 sle_job
担当役員 sle_mgr
入社日 sle_date
給料 sle_sal
COMM sle_comm
部署 sle_dept

照会
cb_sel
修正
cb_udt
削除
cb_del
入力
cb_ins
終了
cb_exit

各項目を入力してください。
st_err

下記は、各ボタンの機能についての説明です。

- 照会

「社員番号」を入力して[照会]ボタンをクリックすると、社員番号±50の範囲内にあるデータがデータ・ウィンドウに出力されます。この機能を実装したスクリプトの例は[\[照会\]ボタンの実行スクリプト](#)を参照してください。

以下はプログラムの照会結果画面です。



[戻る]ボタンをクリックすると、プログラムの初期画面に移動します。この機能を実装したスクリプトの例は[\[戻る\]ボタンの実行スクリプト](#)を参照してください。

- 修正

「社員番号」と修正したいデータを入力して[修正]ボタンをクリックすると、入力されたデータだけが修正されます。この機能を実装したスクリプトの例は[\[修正\]ボタンの実行スクリプト](#)を参照してください。

- 削除

「社員番号」でのみ削除が可能です。この機能を実装したスクリプトの例は[\[削除\]ボタンの実行スクリプト](#)を参照してください。

- 入力

すべてのデータを入力します。この機能を実装したスクリプトの例は[\[入力\]ボタンの実行スクリプト](#)を参照してください。

- 終了

プログラムを終了します。この機能を実装したスクリプトの例は[\[終了\]ボタンの実行スクリプト](#)を参照してください。



グローバル変数

u_tmax	uo_tmax
boolean	btxRun, bConnect

ウィンドウ関数

```
*tmaxconnect ( ) returns Boolean
IF uo_tmax.pb_tpstart()<0 THEN
    st_err.text = ("サーバーに接続できません。 \n tpstart() Failed: Error Code ="
        + string(uo_tmax.tperrno))
    return FALSE
END IF
return TRUE

*inputcheck ( integer inputnum ) returns Boolean
IF isnull( sle_empno.text ) OR sle_empno.text = "" THEN
    st_err.text = "社員番号は必ず入力します。 "
    return FALSE
END IF

IF( inputnum = 1) THEN
return TRUE

IF isnull( sle_name.text ) OR sle_name.text = "" THEN
    goto errormsg
IF isnull( sle_job.text ) OR sle_job.text = "" THEN
    goto errormsg
IF isnull( sle_mgr.text ) OR sle_mgr.text = "" THEN
```



```

        goto errormsg
    IF (len(sle_date.text) <>8 ) THEN
        goto errormsg
    IF isnull( sle_sal.text ) OR sle_sal.text = "" THEN
        goto errormsg
    IF isnull( sle_comm.text ) OR sle_comm.text = "" THEN
        goto errormsg
    IF isnull( sle_dept.text ) OR sle_dept.text = "" THEN
        goto errormsg
    return TRUE

errmsg:
    st_err.text = "すべての項目を入力し、入社日は8桁(YYYYMMDD)で入力します。"
    return FALSE

*clearwindow () returns Booleansle_empno.clear()
sle_name.clear()
sle_job.clear()
sle_mgr.clear()
sle_date.clear()
sle_sal.clear()
sle_comm.clear()
sle_dept.clear()
sle_empno.setfocus()

*errorprocess ( string infomsg )
st_err.text = InfoMsg
IF( btxRun = TRUE ) THEN
    uo_tmax.pb_tx_rollback()
    btxRun = FALSE
END IF

IF( bconnect = TRUE ) THEN
    uo_tmax.pb_tpend()
    bconnect = FALSE
END IF

*successprocess ( string infomsg )
st_err.text = InfoMsg

IF( btxRun = TRUE ) THEN
    uo_tmax.pb_tx_commit()
    btxRun = FALSE
END IF

IF( bconnect = TRUE ) THEN
    uo_tmax.pb_tpend()

```

```

        bconnect = FALSE
    END IF

```

オープン・スクリプト

```

string FileName

uo_tmax = CREATE u_tmax
uo_tmax.flags = 0

FileName = "D:\PB\MyProgram\tmax.env"
IF uo_tmax.pb_tmaxreadenv( FileName, "tmax1") <0 THEN
    messagebox( "Eviroment Error","Enviroment File Loading Failed : Error Code ="

                + string(uo_tmax.tperrno))

    return
END IF
open(w_demo)

```

[照会]ボタンの実行スクリプト

下記は、**[照会]**ボタンを利用して照会を行う例です。

```

string itype, idata, form[], odata[]

IF(inputcheck(1) = FALSE) THEN return

itype = "FIELD~t~n"
idata = "EMPNO"+"~t"+sle_empno.text+"~n"
form[1] = "EMPNO~n" + &
          "ENAME~n" + &
          "JOB~n" + &
          "MGR~n" + &
          "DATE~n" + &
          "SAL~n" + &
          "COMM~n" + &
          "DEPTNO~n" + &
          "E_TYPE~nE_CODE~nE_MSG~nE_TMP~n~n"
form[2] = "~n"//"E_TYPE~nE_CODE~nE_MSG~nE_TMP~n~n"
odata[1] = space(1024)

bConnect = tmaxconnect()
IF bConnect = FALSE THEN
    errorprocess( "サーバーに接続できません。")

```

```

        return
    END IF

    IF uo_tmax.pb_tpcall("FDLSELECT", itype, idata, form[], odata[]) < 0 THEN
        errorprocess( uo_tmax.f_getdata(form[1], odata[1], 1, "E_MSG" ) )
        return
    END IF

    dw_out.importstring(odata[1])
    dw_out.visible = TRUE
    cb_back.visible = TRUE
    cb_sel.enabled = FALSE
    cb_udt.enabled = FALSE
    cb_del.enabled = FALSE
    cb_ins.enabled = FALSE
    cb_exit.enabled = FALSE
    successprocess( sle_empno.text + "社員番号の情報を見つけました。" )

```

[修正]ボタンの実行スクリプト

下記は、[修正]ボタンを利用して修正を行う例です。

```

string itype, idata, form[], odata[]

IF(inputcheck(0) = FALSE) THEN return

itype = "FIELD~t~n"
idata = "EMPNO~t" + sle_empno.text+"~n" + &
        "ENAME~t"+sle_name.text+"~n" + &
        "JOB~t"+sle_job.text+"~n" + &
        "MGR~t"+sle_mgr.text+"~n" + &
        "DATE~t" +sle_date.text+"~n" + &
        "SAL~t" +sle_sal.text+"~n" + &
        "COMM~t" +sle_comm.text+"~n" + &
        "DEPTNO~t" +sle_dept.text+"~n~n"

form[1] = "E_TYPE~nE_CODE~nE_MSG~nE_TMP~n~n"
form[2] = "~n"
odata[1] = space(1024)

bConnect = tmaxconnect()
IF bConnect = FALSE THEN
    errorprocess( "サーバーに接続できません。" )
    return
END IF

IF btxrun = FALSE THEN

```

```

    IF uo_tmax.pb_tx_begin()<0 THEN
        errorprocess( "pb_tx_begin Error")
        return
    END IF
    btxrun = TRUE
END IF

IF uo_tmax.pb_tpcall("FDLUPDATE", itype, idata, form[], odata[])<0 THEN
    errorprocess( "Error:\n"+uo_tmax.f_getdata(form[1], odata[1], 1, "E_MSG") )
    return
END IF

clearwindow()
successprocess( sle_empno.text + "社員番号のデータが修正されました。" )

```

[削除]ボタンの実行スクリプト

下記は、[削除]ボタンを利用して削除を行う例です。

```

int ret
string itype, idata, form[], odata[]

IF(inputcheck(1) = FALSE) THEN return

itype = "FIELD~t~n"
idata = "EMPNO"+"~t"+sle_empno.text+"~n~n"
form[1] = "E_TYPE~nE_CODE~nE_MSG~nE_TMP~n~n"
form[2] = "~n"
odata[1] = space(1024);

bConnect = tmaxconnect()
IF bConnect = FALSE THEN
    errorprocess( "サーバーに接続できません。" )
    return
END IF

IF btxrun = FALSE THEN
    IF uo_tmax.pb_tx_begin()<0 THEN
        errorprocess( "pb_tx_begin Error")
        return
    END IF
    btxrun = TRUE
END IF

IF uo_tmax.pb_tpcall("FDLDELETE", itype, idata, form[], odata[])<0 THEN
    errorprocess( uo_tmax.f_getdata(form[1], odata[1], 1, "E_MSG") )

```

```

        return
    END IF

    successprocess( sle_empno.text + "社員番号のデータが削除されました。" )
    clearwindow()

```

[入力]ボタンの実行スクリプト

下記は、[入力]ボタンを利用して入力を行う例です。

```

long ret
string itype, idata, form[], odata[]

IF(inputcheck(0) = FALSE) THEN return

itype = "FIELD~t~n"
idata = "EMPNO~t"+sle_empno.text+"~n" + &
        "ENAME~t"+sle_name.text+"~n" + &
        "JOB~t"+sle_job.text+"~n" + &
        "MGR~t"+sle_mgr.text+"~n" + &
        "DATE~t"+sle_date.text+"~n" + &
        "SAL~t"+sle_sal.text+"~n" + &
        "COMM~t"+sle_comm.text+"~n" + &
        "DEPTNO~t"+sle_dept.text+"~n" + &
        "~n"
form[1] = "E_TYPE~nE_CODE~nE_MSG~nE_TMP~n~n"
form[2] = "~n"
odata[1] = space(1024);

bConnect = tmaxconnect()
IF bConnect = FALSE THEN
    errorprocess( "サーバーに接続できません。" )
    return
END IF

IF btxrun = FALSE THEN
    IF uo_tmax.pb_tx_begin()<0 THEN
        errorprocess( "pb_tx_begin Error" )
        return
    END IF
    btxrun = TRUE
END IF

IF uo_tmax.pb_tpcall("FDLINSERT", itype, idata, form[], odata[])<0 THEN
    errorprocess( uo_tmax.f_getdata(form[1], odata[1], 1, "E_MSG") )
    return

```

```
END IF

successprocess( sle_empno.text + "社員番号のデータが追加されました。" )
clearwindow()
```

[終了]ボタンの実行スクリプト

下記は、[終了]ボタンを利用して終了を行う例です。

```
close(w_demo)
```

[戻る]ボタンの実行スクリプト

下記は、[戻る]ボタンを利用してプログラムの初期画面に移動する例です。

```
int i
cb_sel.enabled = TRUE
cb_udt.enabled = TRUE
cb_del.enabled = TRUE
cb_ins.enabled = TRUE
cb_exit.enabled = TRUE

FOR i=1 to (dw_out.rowcount()+2)
    dw_out.deleterow ( 1 )
NEXT

clearwindow()
dw_out.visible = FALSE
cb_back.visible = FALSE
```

1.3.5. サーバー・プログラム

サービス・プログラム

下記は、サービス・プログラムの例です。

```
#include <stdio.h>
#include <ctype.h>
#include <tuxinc/macro.h>
#include "../fdl/demo_fdl.h"

EXEC SQL include sqlca.h;
EXEC SQL INCLUDE ORACA;
```

```

EXEC ORACLE  OPTION (ORACA=YES);
EXEC ORACLE  OPTION (RELEASE_CURSOR=YES);

#define INP    1
#define ORA    2
#define TMX    3
#define APP    4

EXEC SQL begin declare section;
int  h_empno;
char h_ename[11];
char h_job[10];
int  h_mgr;
char h_date[11];
float h_sal;
float h_comm;
int  h_deptno;
EXEC SQL end declare section;

void svc_error(long type, long err_code, char *msg, long tmp);

FDLINSERT( TPSVCINFO *msg )
{
    FBUF *rcvbuf;
    int i, occurrence;
    rcvbuf = (FBUF *)msg->data;
    h_empno = h_mgr = h_sal = h_comm = h_deptno = 0;

    memset( h_ename, 0x00, sizeof( h_ename ) );
    memset( h_job, 0x00, sizeof( h_job ) );
    memset( h_date, 0x00, sizeof( h_date ) );

    occurrence = fbkeyoccur(rcvbuf, EMPNO);

    for (i=0; i< occurrence; i++){
        fbget_tu ( rcvbuf, EMPNO, i, (char *)&h_empno, 0 );
        fbget_tu ( rcvbuf, MGR,   i, (char *)&h_mgr, 0 );
        fbget_tu ( rcvbuf, SAL,   i, (char *)&h_sal, 0 );
        fbget_tu ( rcvbuf, COMM,  i, (char *)&h_comm, 0 );
        fbget_tu ( rcvbuf, DEPTNO,i, (char *)&h_deptno, 0 );
        fbget_tu ( rcvbuf, ENAME, i, (char *)h_ename, 0 );
        fbget_tu ( rcvbuf, JOB   , i, (char *)h_job, 0 );
        fbget_tu ( rcvbuf, DATE  , i, (char *)h_date, 0 );

        EXEC SQL INSERT
        INTO emp( empno, ename, job, mgr, hiredate, sal,comm, deptno)
        VALUES ( :h_empno, :h_ename, :h_job, :h_mgr,
        to_date(:h_date,'yyyymmdd'), :h_sal, :h_comm, :h_deptno );
    }
}

```

```

        if(sqlca.sqlcode != 0)
            goto LB_DBERROR;

EXEC SQL WHENEVER SQLERROR
        goto LB_DBERROR;

tpreturn(TPSUCCESS, 0L, (char *)rcvbuf, 0L, 0L);

LB_DBERROR :
        EXEC SQL WHENEVER SQLERROR CONTINUE;
        svc_error(ORA, sqlca.sqlcode, sqlca.sqlerrm.sqlerrmc, 0);
}

FDLDELETE( TPSVCINFO *msg )
{
    FBUF *rcvbuf;
    int i, occurrence;

    rcvbuf = ( FBUF *)msg->data;
    occurrence = fbkeyoccur(rcvbuf, EMPNO);

    for (i=0; i< occurrence; i++){
        fbget_tu (rcvbuf, EMPNO, i, (char *)&h_empno , 0);

        EXEC SQL DELETE
            FROM emp
            WHERE empno = :h_empno;
    }
    if(sqlca.sqlcode != 0)
        goto LB_DBERROR;

EXEC SQL WHENEVER SQLERROR
        goto LB_DBERROR ;

EXEC SQL WHENEVER NOT FOUND
        goto LB_DBERROR ;

tpreturn(TPSUCCESS, 0L, (char *)rcvbuf, 0L, 0L);

LB_DBERROR :
        EXEC SQL WHENEVER SQLERROR CONTINUE;
        svc_error(ORA, sqlca.sqlcode, sqlca.sqlerrm.sqlerrmc, 0);
}

FDLSELECT( TPSVCINFO *msg )
{
    FBUF *rcvbuf;
    FLDLEN fldlen;
    int i=0, ROWMEM=200, CHKROW=500;

```



```

rcvbuf = (FBUF *)msg->data;

if((rcvbuf=(FBUF *)tprealloc((char *)rcvbuf,ROWMEM*CHKROW))==NULL){
    rcvbuf=(FBUF *)msg->data;
    goto LB_TMAXERROR ;
}

h_empno = h_mgr = h_sal = h_comm = h_deptno = 0;

memset( h_ename, 0x00, sizeof( h_ename ) );
memset( h_job, 0x00, sizeof( h_job ) );
memset( h_date, 0x00, sizeof( h_date ) );

fbget_tu( rcvbuf, EMPNO, 0, (char *)&h_empno, &fldlen);

EXEC SQL DECLARE DB_CUR CURSOR FOR
SELECT  nvl(empno,0), nvl(ename,' '), nvl(job,' '),
        nvl(to_char(hiredate,'yyyymmdd'),' '),  nvl(mgr,0),
        nvl(sal,0), nvl(comm,0), nvl(deptno,0)
FROM    emp
WHERE empno >= :h_empno-50 AND empno <= :h_empno+50;
EXEC SQL OPEN DB_CUR;

if(sqlca.sqlcode != 0)
    goto LB_DBERROR;
EXEC SQL WHENEVER SQLERROR
    goto LB_DBERROR ;
EXEC SQL WHENEVER NOT FOUND
    Do break ;

while(1) {
    EXEC SQL FETCH DB_CUR
    INTO      :h_empno,
              :h_ename,
              :h_job,
              :h_date,
              :h_mgr,
              :h_sal,
              :h_comm,
              :h_deptno;

    fbchg_tu(rcvbuf, EMPNO,  i,(char *)&h_empno, 0);
    fbchg_tu(rcvbuf, MGR,   i,(char *)&h_mgr, 0);
    fbchg_tu(rcvbuf, SAL,   i,(char *)&h_sal, 0);
    fbchg_tu(rcvbuf, DEPTNO, i,(char *)&h_deptno, 0);
    fbchg_tu(rcvbuf, COMM,  i,(char *)&h_comm, 0);
    fbchg_tu(rcvbuf, ENAME, i,(char *)&h_ename, 0);
    fbchg_tu(rcvbuf, JOB,   i,(char *)&h_job, 0);
    fbchg_tu(rcvbuf, DATE,  i,(char *)&h_date, 0);

```

```

        i++;
    }

    if (i < 1)
        goto LB_DBERROR;

    EXEC SQL CLOSE DB_CUR;

    tpreturn(TPSUCCESS, 0L, (char *)rcvbuf, 0L, 0L);

    LB_DBERROR :
        EXEC SQL WHENEVER SQLERROR CONTINUE;
        EXEC SQL CLOSE DB_CUR ;
        svc_error(ORA, sqlca.sqlcode, sqlca.sqlerrm.sqlerrmc, 0L) ;

    LB_TMAXERROR :
        EXEC SQL WHENEVER SQLERROR CONTINUE;
        EXEC SQL CLOSE DB_CUR ;
        svc_error(TMX, tperrno, "", 0L) ;
}

FDLUPDATE( TPSVCINFO *msg )
{

    FBUF *rcvbuf ;
    int i, occurrence;
    rcvbuf = (FBUF *)msg->data;
    h_empno = h_mgr = h_sal = h_comm = h_deptno = 0;

    memset( h_ename, 0x00, sizeof( h_ename ) );
    memset( h_job, 0x00, sizeof( h_job ) );
    memset( h_date, 0x00, sizeof( h_date ) );

    occurrence = fbkeyoccur(rcvbuf, EMPNO);

    for (i=0; i< occurrence; i++) {
        fbget_tu ( rcvbuf, EMPNO, i, (char *)&h_empno, 0);
        fbget_tu ( rcvbuf, ENAME, i, (char *)&h_ename, 0);
        fbget_tu ( rcvbuf, JOB, i, (char *)&h_job, 0);
        fbget_tu ( rcvbuf, MGR, i, (char *)&h_mgr, 0);
        fbget_tu ( rcvbuf, SAL, i, (char *)&h_sal, 0);
        fbget_tu ( rcvbuf, COMM, i, (char *)&h_comm, 0);
        fbget_tu ( rcvbuf, DEPTNO, i, (char *)&h_deptno, 0);
        fbget_tu ( rcvbuf, DATE, i, (char *)&h_date, 0 );

        EXEC SQL UPDATE emp
        SET ename = nvl(:h_ename, ename),
            job = nvl(:h_job, job),
            mgr = :h_mgr,

```

```

        hiredate = nvl(to_date(:h_date,'yyyymmdd'), hiredate),
        sal      = :h_sal,
        comm     = :h_comm,
        deptno   = :h_deptno
WHERE empno = :h_empno;

if(sqlca.sqlcode != 0)
    goto LB_DBERROR;

EXEC SQL WHENEVER SQLERROR
    goto LB_DBERROR;

EXEC SQL WHENEVER NOT FOUND
    goto LB_DBERROR;
}

tpreturn(TPSUCCESS, 0L, (char *)rcvbuf, 0L, 0L);

LB_DBERROR :
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    svc_error(ORA, sqlca.sqlcode, sqlca.sqlerrm.sqlerrmc, 0L) ;
}

/*****
 * エラー処理:サービスでエラーの発生時にそのエラーをバッファに入れてクライアントに転送します。
 *****/
void svc_error(long type, long err_code, char *msg, long tmp) {
    FBUF *transf;
    char *svcname;
    char err_msg[100];
    char temp[100];
    int i = 0;

    printf("type ==>[%ld]\n", type);
    printf("err_code ==>[%ld]\n", err_code);
    printf("msg ==>[%s]\n", msg);
    strcpy(err_msg, msg);

    if ((transf = (FBUF *)tpalloc("FML", NULL, 0)) == NULL) {
        printf("tpalloc failed! errno = %d\n", tperrno);
    }

    switch(type) {
        case INP:
            switch(err_code) {
                case -1000:
                    strcpy(err_msg, "当該ユーザーには権限がありません。");
                    break;
                default:
                    strcpy(err_msg,
                        "Inputエラー・メッセージが登録されていません。");
            }
    }
}

```

```

        }
        break;
case ORA:
    if (strlen(err_msg)== 0) strcpy(err_msg, sqlca.sqlerrm.sqlerrmc);
    break;
case TMX:
    if (strlen(err_msg)== 0) strcpy(err_msg, tpstrerror(tperrno));
    break;
case APP:
    if (strlen(err_msg)== 0) {

/* err_mssg=""の場合はエラー・メッセージを設定します。 *****/
switch(err_code) {
    case -500:          /* システム関連エラー */
        strcpy(err_msg, "ファイル作成エラーです。");
        break;
    case -502:
        strcpy(err_msg, "内部サービスを呼び出せませんでした。");
        break;
    case -504:
        strcpy(err_msg, "ソケット通信エラーです。");
        break;
    case -505:  /* 他のトランザクションで修正された場合はMSG処理 */
/* "照会後に他の場所で[%s]データが変更されました。
\n\n再照会した後、処理してください。": クライアントで処理          */
        strcpy(err_msg, "がありません。");
        break;
    case -5002:
        strcpy(err_msg, "電算室にお問い合わせください。");
        break;
    default:
        strcpy(err_msg,
            "アプリケーション・エラー・メッセージが登録されていません。");
}
    }
    break;
}

/* ROLLBACK          */
EXEC SQL WHENEVER SQLERROR CONTINUE;
EXEC SQL ROLLBACK;

fbchg_tu ( transf, E_TYPE, i, (char *)&type,0);
fbchg_tu ( transf, E_CODE, i, (char *)&err_code,0);
fbchg_tu ( transf, E_MSG, i, (char *)err_msg,0);
fbchg_tu ( transf, E_TMP, i, (char *)&tmp,0);

tpreturn(TPFAIL, 0, (char *)transf, 0L, 0L);
}

```

メイクファイル(Makefile)

下記は、emp_c.pcソースをTmaxアプリケーションに作成するメイクファイルの例です。

```
include $(ORACLE_HOME)/precomp/lib/env32.mk
ORALIBDIR = $(LIBHOME)
ORALIB = -L/home/oracle/OraHome/lib32/ -lclntsh -lld -lm
        `cat /home/oracle/OraHome/lib32/sysliblist` -lm -lc_r -lpthreads
TARGET = emp_c
APOBJS = emp_c.o
NSDLOBJ = $(TMAXDIR)/lib/sdl.o
#CC
CC=cc

#Oracle
LIBS = -lsvr -loras

OBS = $(APOBJS) $(SVCTOBJ)
SVCTOBJ = $(TARGET)_svctab.o

CFLAGS = -q32 -O -I$(TMAXDIR)
LDFLAGS = -brtl
APPDIR = $(TMAXDIR)/appbin
SVCTDIR = $(TMAXDIR)/svct
TMAXLIBDIR = $(TMAXDIR)/lib
#
.SUFFIXES : .c

.c.o:
        $(CC) $(CFLAGS) $(LDFLAGS) -c $<

all: $(TARGET)

$(TARGET): $(OBS)
        $(CC) $(CFLAGS) $(LDFLAGS) -L$(TMAXLIBDIR) -o $(TARGET)
        -L$(ORALIBDIR)
        $(ORALIB) $(OBS) $(LIBS) $(NSDLOBJ)
        mv $(TARGET) $(TMAXDIR)/appbin

$(APOBJS): $(TARGET).pc
        proc iname=emp_c include=$(TMAXDIR)
        define=__LINUX_ORACLE_PROC__
        $(CC) $(CFLAGS) $(LDFLAGS) -c $(TARGET).c

$(SVCTOBJ):
        touch $(SVCTDIR)/$(TARGET)_svctab.c
```

```
$(CC) $(CFLAGS) $(LDFLAGS) -c $(SVCTDIR)/$(TARGET)_svctab.c
#
clean:

-rm -f *.o core $(TARGET) $(TARGET).lis
```

第2章 Visual Basicインターフェース

本章では、Visual Basicインターフェースで使用する関数と例について説明します。

2.1. 概要

Visual BasicインターフェースにはPowerBuilderとは違って別途のモジュールがありません。単純にクライアント・ライブラリーが呼び出せるように、関数のプロトタイプを定義したインターフェース・モジュールが存在します。

Visual Basicインターフェース・モジュールは下記のとおりです。

モジュール	説明
atmi.bas	atmi関数のプロトタイプ定義ファイルです
fdl. Bas	フィールド・キー関数のプロトタイプ定義ファイルです
comm.bas	ユーザーの便宜のためのマクロ定義ファイルです
winapi.bas	Windowsで提供する関数定義ファイルです

開発者は、上記のインターフェース・モジュールをインストールすると、Tmaxクライアント・ライブラリーで提供する関数を呼び出して使用することができます。しかしVisual Basicは、値の転送方法が既存のアプリケーションとは大きく異なるため、この点に注意して使用する必要があります。さらに、直接的なバッファ操作が困難なため、マクロ(comm.bas)を提供してデータを入力/出力する作業を内部的に処理します。

下記は、マクロで提供する関数の一覧です。

関数	説明
FdlErrorMsg	画面に与えられたStrErrメッセージと一緒にfberrnoをメッセージボックスで出力する関数です
FilltpstartBuf	Tmaxシステムと接続するために設定するバッファのtpstart_t構造体を構成する関数です
GETCAR	CARRAYバッファにString型データのtextを入力する関数です
GETCAR_BA	FIELD(FDL)バッファにFieldに指定されたフィールド名で入力されたフィールド順idxのImage型データをバイト配列に保存する関数です
GETCHR	Tmax FDL関数のfbchg_tu()を利用して指定したnthに該当する順番でFieldに指定された名前にvalue値を取得する関数です
GETDOUBLE	FIELD(FDL)バッファにFieldに指定されたフィールド名で入力されたフィールド順idxのDouble型データをdDataに保存する関数です

関数	説明
GETFLOAT	FIELD(FDL)バッファにFieldに指定されたフィールド名で入力されたフィールド順idxのSingle型データをdDataに保存する関数です
GETINT	FIELD(FDL)バッファにFieldに指定されたフィールド名で入力されたフィールド順idxのInteger型データをiDataに保存する関数です
GETLONG	FIELD(FDL)バッファにFieldに指定されたフィールド名で入力されたフィールド順idxのLong型データをlDataに保存する関数です
GETSTR	STRINGバッファのデータをtextに保存する関数です
GETVAR	FIELD(FDL)バッファにFieldに指定されたフィールド名で入力されたフィールド順idxのString型データをtextに保存する関数です
PUTCAR	CARRAYバッファにString型データのtextを入力する関数です
PUTCAR_BA	FIELDバッファにImageデータをフィールド順で入力する関数です
PUTCHR	Tmax FDL関数のfbchg_tu()を利用して指定したnthに該当する順番でfldNameに指定されたフィールド名にvalue値を保存する関数です
PUTDOUBLE	FIELDバッファにDouble型データをフィールド順に入力する関数です
PUTFLOAT	FIELDバッファにSingle型データをフィールド順に入力する関数です
PUTINT	FIELDバッファにInteger型データをフィールド順に入力する関数です
PUTLONG	FIELDバッファにLong型データをフィールド順に入力する関数です
PUTSTR	STRINGバッファのデータをtextに保存する関数です
PUTVAR	FIELDバッファにString型データをフィールド順に入力する関数です
TmaxError	与えられたStrErrメッセージと一緒にtperrnoに該当する文字列エラーが、メッセージボックスで画面に出力される関数です

参考

atmi関数とフィールド・キー関数のプロトタイプおよび機能は、『Tmax リファレンスガイド』および『Tmax アプリケーション開発ガイド』を参照してください。

2.2. 関数

2.2.1. FdlErrorMsg

画面に与えられたStrErrメッセージと一緒にfberrnoをメッセージボックスで出力する関数です。

- プロトタイプ

```
Sub FdlErrorMsg(StrErr As String)
```


- パラメータ

パラメータ	説明
StrErr	エラー・メッセージです

- 戻り値

メッセージ画面を出力します。

- 例

```
lret = fbput(ByVal lsendbuf, ByVal lfid, ByVal txtInput.text, ByVal 0)
If lret = -1 Then
    FdlErrorMsg("fbput")
    error processing
    ...
End If
```

- 関連関数

TmaxError()

2.2.2. FilltpstartBuf

Tmaxシステムと接続するために設定するバッファのtpstart_t構造体を構成する関数です。

- プロトタイプ

```
Function FilltpstartBuf(sndbufp As Long, startinfop As tpstart_t)
As Long
```

- パラメータ

パラメータ	説明
sndbufp	tpalloc関数で、メモリーに割り当てられたバッファです
startinfop	tpstart_t型で宣言された変数です

- 例

```
Dim lsndbuf As Integer
Dim lret As Integer
Dim cinfo As tpstart_t

lsndbuf = tpalloc("TPSTART", "", 0)
```

```

If lsndbuf = 0 Then
    error processing
End If

cinfo.cltname = "tmax" + Chr(0)
cinfo.usrname = "tmax" + Chr(0)
cinfo.dompwd = "xamt" + Chr(0)
cinfo.usrpwd = "batman" + Chr(0)
cinfo.flags = TPUNSOL_HND

lret = FilltpstartBuf(lsndbuf, cinfo)
If ret < 0 Then
    error processing
End If

ret = tpstart(lsndbuf)
If ret < 0 Then
    error processing
End If
...

```

- 関連関数

TmaxError()

2.2.3. GETCAR

CARRAYバッファにString型データのtextを入力する関数です。

- プロトタイプ

```

Function GETCAR(ByVal Carptr&, text As String, Datalen As Long)
As Long

```

- パラメータ

パラメータ	説明
Carptr	FDLバッファのポインターです
text	データです
Datalen	データ長です

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 例

```

Dim lsendbuf As Long
Dim lrecvbuf As Long
Dim text As String
Dim lret As Long
Dim lrbuflen As Long
...
lsendbuf = tpalloc("CARRAY", "", 1024)
lrecvbuf = tpalloc("CARRAY", "", 1024)
...
lret = tpcall(ByVal "SVC1", ByVal lsendbuf, ByVal DataLen&, lrecvbuf, lrbuflen,

                ByVal 0)
...
'CARRAYバッファでは必ずデータ長の値を入力します。
lret = GETCAR(lsendbuf, text, lrbuflen)
...

```

- 関連関数

GETINT()、GETLONG()、GETDOUBLE()、GETVAR()、GETSTR()、GETFLOAT()、GETSTR()、GETCHR()

2.2.4. GETCAR_BA

FIELD(FDL)バッファにFieldに指定されたフィールド名で入力されたフィールド順idxのImage型データをバイト配列に保存する関数です。

- プロトタイプ

```

Function GETCAR_BA (ByVal Fdlptr&, Field As String, idx As Long,
                    ByRef ByteArray() As Byte, datalen As Long)
As Long

```

- パラメータ

パラメータ	説明
Fdlptr	FDLバッファのポインターです

パラメータ	説明
Field	フィールド名です
idx	フィールド順です
ByteArray	データです
datalen	バイト配列の長さです

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 例

```

Dim rbuffer(102400) As Byte
Dim datalen As Long
...
lrecvbuf = tpalloc("FIELD", "", 102400)
...
lret = PUTCAR_BA(ByteVal lsendbuf, "TP_BITMAP", 0, lbuffer, datalen)
if lret < 0 then
    error processing
    ...
end if
...
lret = tpcall(ByteVal "FILEUP", ByteVal lsendbuf, ByteVal datalen&, lrecvbuf,
              lrbufen, ByteVal 0)
lblComment.Caption = lblComment.Caption & " -> tpcall : " & lret
If lret = -1 Then
    error processing
    ...
end If
...
lret = GETCAR_BA(ByteVal lrecvbuf, "TP_BITMAP", 0, rbuffer, datalen)
if lret < 0 then
    error processing
    ...
end if
...

```

- 関連関数

GETINT()、GETLONG()、GETDOUBLE()、GETCAR()、GETVAR()、GETFLOAT()、GETSTR()、GETCHR()

2.2.5. GETCHR

Tmax FDL関数のfbchg_tu()を利用して指定したnthに該当する順番でFieldに指定されたフィールド名のvalue値を取得する関数です。このとき、value値は文字型(Char)になります。fbget_tu()を使用するため、フィールド順(idx)のデータ(char)を取得します。

- プロトタイプ

```
Function GETCHR(ByVal Fdlptr&, ByVal Field As String, ByVal idx As Integer,  
                ByRef data As String, ByVal datalen As Integer)  
As Integer
```

- パラメータ

パラメータ	説明
Fdlptr	STRINGバッファのポインターです
Field	フィールド名です
idx	フィールド順です
data	保存するString型データです
datalen	保存するString型データの長さです

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 例

```
Dim lsendbuf As Long  
Dim lret As Long  
Dim chr As String  
lsendbuf = fballocc(10, 100)  
lrecvbuf = fballocc(10, 100)  
...  
lret = tpcall(ByVal "SVC1", ByVal lsendbuf, ByVal DataLen&, lrecvbuf, lrbuflen,  
              ByVal 0)  
...  
lret = GETSTR(lrecvbuf, "CHR", 0, chr, 1 )  
...
```

- 関連関数

GETINT()、GETLONG()、GETDOUBLE()、GETVAR()、GETSTR()、GETCAR()、GETFLOAT()

2.2.6. GETDOUBLE

FIELD(FDL)バッファにFieldに指定されたフィールド名で入力されたフィールド順idxのDouble型データをdDataに保存する関数です。

- プロトタイプ

```
Function GETDOUBLE(ByVal Fdlptr&, Field As String, idx As Long, dData As Double)
    As Long
```

- パラメータ

パラメータ	説明
Fdlptr	FDLバッファのポインターです
Field	フィールド名です
idx	フィールド順です
dData	データです

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 例

```
Dim lsendbuf As Long
Dim lret As Long
Dim doubledata As Double
lsendbuf = tmalloc("FIELD ", "", 0)
...
lret = GETDOUBLE(ByVal lsendbuf, "DOUBLEDATA", 0, doubledata)
...
```

- 関連関数

GETINT()、GETLONG()、GETVAR()、GETCAR()、GETFLOAT()、GETSTR()、GETCHR()

2.2.7. GETFLOAT

FIELD(FDL)バッファにFieldに指定されたフィールド名で入力されたフィールド順idxのSingle型データをdDataに保存する関数です

- プロトタイプ

```
Function GETFLOAT (ByVal Fdlptr&, Field As String, idx As Long, dData As Single)
    As Long
```

- パラメータ

パラメータ	説明
Fdlptr	FDLバッファのポインターです
Field	フィールド名です
idx	フィールド順です
dData	データです

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 例

```
Dim lsendbuf As Long
Dim lret As Long
Dim doubledata As Single

lsendbuf = tmalloc("FIELD ", "", 0)
...
lret = GETFLOAT (ByVal lsendbuf, "DOUBLEDATA", 0, doubledata)
...
```

- 関連関数

GETINT(), GETLONG(), GETVAR(), GETCAR(), GETSTR(), GETCHR(), GETDOUBLE()

2.2.8. GETINT

FIELD(FDL)バッファにFieldに指定されたフィールド名で入力されたフィールド順idxのInteger型データをiDataに保存する関数です

- プロトタイプ

```
Function GETINT(ByVal Fdlptr&, Field As String, idx As Long, iData As Integer)
As Long
```

- パラメータ

パラメータ	説明
Fdlptr	FDLバッファのポインターです
Field	フィールド名です
idx	フィールド順です
iData	データです

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 例

```
Dim lsendbuf As Long
Dim lret As Long
Dim intdata As Integer

lsendbuf = tpalloc("FIELD ", "", 0)

lret = GETINT(ByVal lsendbuf, "INTDATA", 0, intdata)
...
```

- 関連関数

GETLONG()、GETDOUBLE()、GETVAR()、GETCAR()、GETFLOAT()、GETSTR()、GETCHR()

2.2.9. GETLONG

FIELD(FDL)バッファにFieldに指定されたフィールド名で入力されたフィールド順idxのLong型データをIDataに保存する関数です

- プロトタイプ

```
Function GETLONG(ByVal Fdlptr&, Field As String, idx As Long, lData As Long)
As Long
```

- パラメータ

パラメータ	説明
Fdlptr	FDLバッファのポインターです
Field	フィールド名です
idx	フィールド順です
IData	データです

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 例

```
Dim lsendbuf As Long
Dim lret As Long
Dim longdata As Long

lsendbuf = tmalloc("FIELD ", "", 0)
...
lret = GETLONG(ByVal lsendbuf, "LONGDATA", 0, longdata)
...
```

- 関連関数

GETINT()、GETDOUBLE()、GETVAR()、GETCAR()、GETFLOAT()、GETSTR()、GETCHR()

2.2.10. GETSTR

STRINGバッファのデータをtextlに保存する関数です

- プロトタイプ

```
Function GETSTR(ByVal strptr&, text As String)
As Long
```

- パラメータ

パラメータ	説明
strptr	STRINGバッファのポインターです
text	データです

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 例

```
Dim lsendbuf As Long
Dim lret As Long
Dim text As String

lsendbuf = fmalloc(10, 100)
lrecvbuf = fmalloc(10, 100)

...
lret = tpcall(ByVal "SVC1", ByVal lsendbuf, ByVal DataLen&, lrecvbuf, lrbuflen,

               ByVal 0)

...
lret = GETSTR(lrecvbuf, text)
...
```

- 関連関数

GETINT()、GETLONG()、GETDOUBLE()、GETVAR()、GETCAR()、GETFLOAT()、GETSTR()、GETCHR()

2.2.11. GETVAR

FIELD(FDL)バッファにFieldに指定されたフィールド名で入力されたフィールド順idxのString型データをtextに保存する関数です

- プロトタイプ

```
Function GETVAR(ByVal Fdlptr&, Field As String, idx As Long, text As String)
As Long
```

- パラメータ

パラメータ	説明
Fdlptr	FDLバッファのポインターです
Field	フィールド名です
idx	フィールド順です
text	データです

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 例

```
Dim lsendbuf As Long
Dim lret As Long

lsendbuf = tmalloc("FIELD ", "", 0)
...
lret = GETVAR(ByVal lsendbuf, "INPUT", 0, txtOutput.text)
...
```

- 関連関数

GETINT()、GETLONG()、GETDOUBLE()、GETCAR()、GETFLOAT()、GETSTR()、GETCHR()

2.2.12. PUTCAR

CARRAYバッファにString型データのtextを入力する関数です

- プロトタイプ

```
Function PUTCAR(ByVal Carptr&, text As String, Datalen As Long)
As Long
```

- パラメータ

パラメータ	説明
Carptr	FDLバッファのポインターです
text	データです
Datalen	データ長です

- 戻り値

戻り値	説明
-1以外の値	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 例

```
Dim lret As Long
Dim DataLen As Long
lsendbuf = tpalloc("CARRAY", "", 1024)
'CARRAYバッファでは必ずデータ長の値を入力します。

DataLen = LenB(txtInput.text)

lret = PUTCAR(ByVal lsendbuf, txtInput.text, DataLen)
...
```

STRINGバッファは、システムで提供するlstrcpy関数を使用してデータをバッファに入れることができます。

```
Dim lsendbuf As Long
Dim lret As Long

lsendbuf = tpalloc("STRING", "", 1024)
lret = lstrcpy(ByVal lsendbuf, ByVal txtInput.text)
```

- 関連関数

PUTINT()、PUTLONG()、PUTDOUBLE()、PUTVAR()、PUTCAR()、PUTFLOAT()、PUTSTR()

2.2.13. PUTCAR_BA

FIELDバッファにImage型データをフィールド順で入力する関数です。FIELD(FDL)バッファにFieldに指定されたフィールド名でImage型データのバイト配列をフィールド順idxで入力します。

- プロトタイプ

```
Function PUTCAR_BA(ByVal Fdlptr&, Field As String, idx As Long, ByteArray()  
                  As Byte, datalen As Long)  
As Long
```

- パラメータ

パラメータ	説明
Fdlptr	FDLバッファのポインターです
Field	フィールド名です
idx	フィールド順です
ByteArray	データです
datalen	バイト配列長です

- 戻り値

戻り値	説明
-1以外の値	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 例

```
Dim iSrc As Integer  
Dim lCopy As Long  
Dim lSize As Long  
Dim lbuffer() As Byte  
Dim datalen As Long  
...  
lsendbuf = tmalloc("FIELD", "", 102400)  
...  
iSrc = FreeFile  
Open ".\nmlogo.bmp" For Binary Access Read As iSrc  
lSize = LOF(iSrc)  
datalen = lSize  
Do  
    If lSize > MAX_BUFFER Then  
        lSize = lSize - MAX_BUFFER  
        lCopy = MAX_BUFFER  
    Else  
        lCopy = lSize  
    End If  
  
    ReDim lbuffer(lCopy - 1)
```

```

    Get iSrc, , lbuffer
    Loop Until lCopy = lSize

    Close iSrc

    lret = PUTCAR_BA(ByVal lsendbuf, "TP_BITMAP", 0, lbuffer, datalen)
    if lret < 0 then
        error processing
        ...
    end if
    ...

```

- 関連関数

PUTINT()、PUTLONG()、PUTDOUBLE()、PUTCAR()、PUTVAR()、PUTFLOAT()、PUTSTR()、PUTCHR()

2.2.14. PUTCHR

Tmax FDL関数のfbchg_tu()を利用して指定したnthに該当する順番でfldNameに指定されたフィールド名にvalue値を保存する関数です。このとき、value値は文字型(Char)になります。Fbchg_tu()を使用するため、フィールド順(nth)に既存のデータがある場合は与えられたデータ(value)に変更され、フィールド順に既存のデータがない場合は、与えられたデータが自動的にフィールドに追加されます。

- プロトタイプ

```

Function PUTCHR(ByVal Fdlptr&, ByVal Field As String, ByVal idx As Integer,
                ByRef data As String, ByVal datalen As Integer)
As Integer

```

- パラメータ

パラメータ	説明
Fdlptr	STRINGバッファのポインターです
Fieldt	フィールド名です
idx	フィールド順です
data	保存するString型データです
datalen	保存するString型データの長さです

- 戻り値

戻り値	説明
-1以外の値	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 例

```
Dim lsendbuf As Long
Dim lret As Long
Dim chr As String

lsendbuf = fmalloc(10, 100)
chr = "a"
...
lret = PUTSTR(lsendbuf, "CHR", 0, chr, 1 )
...
Function PUTCHR(ByVal Fdlptr&, ByVal Field As String, ByVal idx As Integer,
                ByRef data As Char, ByVal datalen As Integer)
```

- 関連関数

PUTINT()、PUTLONG()、PUTDOUBLE()、PUTVAR()、PUTCAR()、PUTFLOAT()、PUTSTR()

2.2.15. PUTDOUBLE

FIELDバッファにDouble型データをフィールド順で入力する関数です。FIELD(FDL)バッファにFieldに指定されたフィールド名でDouble型データのdDataをフィールド順idxで入力します。

- プロトタイプ

```
Function PUTDOUBLE(ByVal Fdlptr&, Field As String, idx As Long, dData As Double)

As Long
```

- パラメータ

パラメータ	説明
Fdlptr	FDLバッファのポインターです
Field	フィールド名です
idx	フィールド順です
dData	データです

- 戻り値

戻り値	説明
-1以外の値	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 例

```
Dim lsendbuf As Long
Dim lret As Long
Dim doubledata As Double

lsendbuf = tpalloc("FIELD ", "", 0)
...
lret = PUTDOUBLE(ByVal lsendbuf, "DOUBLEDATA", 0, doubledata)
...
```

- 関連関数

PUTINT()、PUTLONG()、PUTVAR()、PUTCAR()、PUTFLOAT()、PUTSTR()、PUTCHR()

2.2.16. PUTFLOAT

FIELDバッファにSingle型データをフィールド順で入力する関数です。FIELD(FDL)バッファにFieldに指定されたフィールド名でSingle型データのdDataをフィールド順idxで入力します。

- プロトタイプ

```
Function PUTFLOAT(ByVal Fdlptr&, Field As String, idx As Long, dData As Single)
    As Long
```

- パラメータ

パラメータ	説明
Fdlptr	FDLバッファのポインターです
Field	フィールド名です
idx	フィールド順です
dData	データです

- 戻り値

戻り値	説明
-1以外の値	関数呼び出しに成功した場合です

戻り値	説明
-1	関数呼び出しに失敗した場合です

- 例

```
Dim lsendbuf As Long
Dim lret As Long
Dim doubledata As Single

lsendbuf = tmalloc("FIELD ", "", 0)
...
lret = PUTFLOAT(ByVal lsendbuf, "DOUBLEDATA", 0, doubledata)
...
```

- 関連関数

PUTINT(), PUTLONG(), PUTVAR(), PUTCAR(), PUTDOUBLE(), PUTSTR(), PUTCHR()

2.2.17. PUTINT

FIELDバッファにInteger型データをフィールド順で入力する関数です。FIELD(FDL)バッファにFieldに指定されたフィールド名でInteger型データのiDataをフィールド順idxで入力します。

- プロトタイプ

```
Function PUTINT(ByVal Fdlptr&, Field As String, idx As Long, iData As Integer)
As Long
```

- パラメータ

パラメータ	説明
Fdlptr	FDLバッファのポインターです
Field	フィールド名です
idx	フィールド順です
iData	データです

- 戻り値

戻り値	説明
-1以外の値	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 例

```
Dim lsendbuf As Long
Dim lret As Long
Dim intdata As Integer

lsendbuf = tpalloc("FIELD ", "", 0)
...
lret = PUTINT(ByVal lsendbuf, "INTDATA", 0, intdata)
...
```

- 関連関数

PUTLONG(), PUTDOUBLE(), PUTVAR(), PUTCAR(), PUTFLOAT(), PUTSTR(), PUTCHR()

2.2.18. PUTLONG

FIELDバッファにLong型データをフィールド順で入力する関数です。FIELD(FDL)バッファにFieldに指定されたフィールド名でLong型データのIDataをフィールド順idxで入力します。

- プロトタイプ

```
Function PUTLONG(ByVal Fdlptr&, Field As String, idx As Long, lData As Long)
As Long
```

- パラメータ

パラメータ	説明
Fdlptr	FDLバッファのポインターです
Field	フィールド名です
idx	フィールド順です
IData	データです

- 戻り値

戻り値	説明
-1以外の値	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 例

```
Dim lsendbuf As Long
Dim lret As Long
```

```

Dim longdata As Long

lsendbuf = tmalloc("FIELD ", "", 0)
...
lret = PUTLONG(ByVal lsendbuf, "LONGDATA", 0, longdata)
...

```

- 関連関数

PUTINT()、PUTDOUBLE()、PUTVAR()、PUTCAR()、PUTFLOAT()、PUTSTR()、PUTCHR()

2.2.19. PUTSTR

STRINGバッファのデータをtextに保存する関数です。

- プロトタイプ

```

Function PUTSTR(ByVal strptr&, text As String)
As Long

```

- パラメータ

パラメータ	説明
strptr	STRINGバッファのポインターです
text	データです

- 戻り値

戻り値	説明
-1以外の値	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 例

```

Dim lsendbuf As Long
Dim lret As Long
Dim text As String

lsendbuf = tmalloc("STRING", "", 1024)
...
lret = PUTSTR(lsendbuf, text)
...

```

- 関連関数

PUTINT()、PUTLONG()、PUTDOUBLE()、PUTVAR()、PUTCAR()、PUTFLOAT()、PUTCHR()

2.2.20. PUTVAR

FIELDバッファにString型データをフィールド順で入力する関数です。FIELD(FDL)バッファにFieldに指定されたフィールド名でString型データのtextをフィールド順idxで入力します。

- プロトタイプ

```
Function PUTVAR(ByVal Fdlptr&, Field As String, idx As Long, text As String)
As Long
```

- パラメータ

パラメータ	説明
Fdlptr	FDLバッファのポインターです
Field	フィールド名です
idx	フィールド順です
text	データです

- 戻り値

戻り値	説明
-1以外の値	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 例

```
Dim lsendbuf As Long
Dim lret As Long

lsendbuf = tpalloc("FIELD ", "", 0)
...
lret = PUTVAR(ByVal lsendbuf, "INPUT", 0, txtInput.text)
...
```

- 関連関数

PUTINT()、PUTLONG()、PUTDOUBLE()、PUTCAR()、PUTFLOAT()、PUTSTR()、PUTCHR()

2.2.21. TmaxError

与えられたStrErrメッセージと一緒にtperrnoに該当する文字列エラーがメッセージボックスで画面に出力される関数です。

- プロトタイプ

```
Sub TmaxError(StrErr As String)
```

- パラメータ

パラメータ	説明
StrErr	エラー・メッセージです

- 戻り値

メッセージ画面を出力します。



- 例

```
Dim lsendbuf As Long
Dim lrecvbuf As Long
Dim lret As Long
Dim lrbuflen As Long

lsendbuf = tpalloc("CARRAY", "", 1024)
lrecvbuf = tpalloc("CARRAY", "", 1024)
lret = tpcall(ByVal "SVC1", ByVal lsendbuf, ByVal 0, lrecvbuf, lrbuflen, ByVal 0)

If lret = -1 Then
    TmaxError ("tpcall (SVC1)")
End If
```

- 関連関数

FdlErrorMsg()

2.3. サンプル・プログラム

クライアント・プログラムはユーザーの要求を受けてサービスを要求し、サーバーはクライアントが要求したとおりOracleデータベースの社員情報(EMP)の表でデータを入力、修正、照会、削除します。

2.3.1. プログラムの構成

サーバーとクライアントは環境設定が行われている必要があります。TMAXDIR、ポート番号、サーバーIPなどの環境設定が既に完了されていると前提し、設定についての説明は省略します。

下記のサンプル・ファイルをモジュールとしてプロジェクトで追加します。

- <atmi.bas>
- <fdl.bas>
- <comm.bas>
- <winapi.bas>

それぞれのプログラムは、以下のファイルで構成されています。

- 共通プログラム

プログラムファイル	説明
demo.f	フィールド・キー・バッファを定義したファイルです
tmax	ライブラリーファイルです
tmconfig.m	Tmax環境設定ファイルです

- クライアント・プログラム

プログラムファイル	説明
EmployeeGrid.frm	照会結果を表示するプログラムです
EmployeeGrid.frx	照会結果を表示するプログラムです
EmployeeMgr.frm	メイン・プログラムです。照会、修正、削除、入力を行います
MSSCCPRJ.SCC	ソースコード制御ファイルです
QA_4GL.vbg	クライアント・プログラムのグループ・プロジェクト・ファイルです
QA_4GL_Sample.exe	クライアント・プログラムの実行ファイルです
QA_4GL_Sample.vbp	クライアント・プログラムのプロジェクト・ファイルです
QA_4GL_Sample.vbw	クライアント・プログラムの作業領域(workspace)ファイルです
Atmi.bas	atmi関数のプロトタイプを定義するファイルです

プログラムファイル	説明
comm.bas	ユーザーの便宜のためのマクロ定義ファイルです
fdl.bas	フィールド・キー関数のプロトタイプを定義するファイルです
winapi.bas	Windowsで提供する関数定義ファイルです

- サーバー・プログラム

プログラムファイル	説明
emp_c.mk	メイクファイルです
emp_c.pc	サービスを行うサーバー・プログラムです。AIXとOracle 9iを使用します

2.3.2. プログラムの特徴

下記はプログラムの特徴です。

- クライアント・プログラム

機能	説明
Tmax接続	NULL引数で接続します
バッファ・タイプ	フィールド・キー・バッファ、構造体ファイルをfdlcユーティリティでコンパイルして「fdl」ファイルを作成する必要があります
通信タイプ	tpcall()を利用した同期通信を送信するバッファと受信するバッファを別々に指定します
トランザクションの有無	TMSで自動トランザクション(AutoTransaction)を付与します

- サーバー・プログラム

機能	説明
サービス	FDLSELECT、FDLINSERT、FDLDELETE、FDLUPDATEを作成します
データベース指定	Oracleデータベースを使用します。システム構成ファイルのSVRGROUPにデータベース情報を指定します
通信タイプ	tpcall()を利用した同期通信を送信するバッファと受信するバッファを別々に指定します
トランザクションの有無	TMSで自動トランザクションを付与します

2.3.3. 共通プログラム

データベースのEMP表

下記は、Tmax EMPデータベース(Oracle)表の例です。

EMPNO	NUMBER	NOT NULL	P1
ENAME	VARCHAR(16)		
JOB	VARCHAR(16)		
MGR	NUMBER		
HIREDATE	DATE		
SAL	NUMBER(7,2)		
COMM	NUMBER(7,2)		
DEPTNO	NUMBER		

Tmax EMPフィールド表

下記は、Tmax EMPフィールド表の例です。

<demo.f>

#For tmax demo employee program				
EMPNO	7500	long	-	-
ENAME	7501	string	-	-
JOB	7502	string	-	-
MGR	7503	long	-	-
DATE	7504	string	-	-
SAL	7505	float	-	-
COMM	7506	float	-	-
DEPTNO	7507	long	-	-
E_TYPE	9009	long	-	-
E_CODE	9010	long	-	-
E_MSG	9011	string	-	-
E_TMP	9012	long	-	-

Tmax環境設定

下記は、Tmax環境設定ファイルの例です。

<tmconfig.m>

*DOMAIN	
dom1	SHMKEY = 70000, MAXUSER = 200, MINCLH = 1, MAXCLH = 5, TPORTNO = 8888, BLOCKTIME = 200, TXTIME = 200
*NODE	


```

tmax1      TMAXDIR = "/home/tmax",
           APPDIR = "/home/tmax/appbin",
           PATHDIR = "/home/tmax/path",
           TLOGDIR = "/home/tmax/log/tlog",
           SLOGDIR = "/home/tmax/log/slog"
           ULOGDIR = "/home/tmax/log/ulog"

*SVRGROUP
svg1       NODENAME = tmax1,
           DBNAME = ORACLE,
           OPENINFO = "ORACLE_XA+Acc=P/scott/tiger+SesTm=60",
           TMSNAME = svg1_tms

*SERVER
emp_c      SVGNAME = svg1, MIN = 1

*SERVICE
FDLSELECT  VRNAME = emp_c
FDLUPDATE  SVRNAME = emp_c
FDLDELETE  SVRNAME = emp_c
FDLINSERT  SVRNAME = emp_c

```

2.3.4. クライアント・プログラム

メイン画面フォーム

下記は、メイン画面フォームのデザイン画面です。

下記は、例で使用するメイン画面フォームのデザイン構成表です。

制御名	制御タイプ	備考
EmployeeMgr	フォーム	Caption="社員管理プログラム"
LabelErr	ラベル	Caption="エラー"
BtnExit	コマンドボタン	Caption="終了"
BtnIns	コマンドボタン	Caption="入力"
BtnDel	コマンドボタン	Caption="修正"
BtnUdt	コマンドボタン	Caption="照会"
BtnSel	コマンドボタン	
EditName	テキストボックス	
EditEmpNo	テキストボックス	
EditDept	テキストボックス	
EditComm	テキストボックス	
EditSal	テキストボックス	
EditDate	テキストボックス	
EditMgr	テキストボックス	
EditJob	テキストボックス	

下記は、メイン画面フォームのデザインを実行するための例です。

```
EmployeeMgr.frm Source
Option Explicit

Private Sub BtnDel_Click()

    Dim Isendbuf As Long
    Dim Irecvbuf As Long
    Dim Irbuflen As Long
    Dim Iret As Long
    Dim text As String
    Dim value As Long
    Dim dvalue As Double
    Dim svalue As Single
    Dim tx_b As Integer
    Dim txbool As Integer ' トランザクションが開始されると1,されなければ0 '

    ' tpstart '
    tmaxStart

    ' トランザクションの開始有無を初期化 '
    txbool = 0
```

```

If EditEmpNo.text = "" Then
    MsgBox "社員番号を入力してください。"
    tmaxEnd
    Exit Sub
End If

value = Trim(EditEmpNo.text)

' 送信するデータのためのバッファ割り当て '
Isendbuf = fmalloc(100, 1024)
If Isendbuf = Null Then
    TmaxError ("fmalloc")
    Call fbfree(Isendbuf)
    tmaxEnd
    Exit Sub
End If

' 受信するデータのためのバッファ割り当て '
Irecvbuf = fmalloc(100, 1024)
If Irecvbuf = Null Then
    TmaxError ("fmalloc")
    Call fbfree(Irecvbuf)
    tmaxEnd
    Exit Sub
End If

Iret = PUTLONG(ByVal Isendbuf, "EMPNO", 0, value)
If Iret = -1 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

' トランザクション開始 '
Iret = tx_begin
txbool = 1
If Iret < 0 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

' 照会サービスを要求 '
Iret = tpcall(ByVal "FDLDELETE", ByVal Isendbuf, ByVal 0, Irecvbuf,
              Irbuflen, ByVal 0)

LabelErr.Caption = "sbuf = " & Isendbuf & ", rbuf = " & Irecvbuf

If Iret = -1 Then

```

```

        TmaxError ("tpcall(SVC1)")
        ViewErr (Irecvbuf)
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    End If

    ' トランザクション保存 '
    Iret = tx_commit
    If Iret < 0 Then
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    Else
        txbool = 1
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    End If

End Sub

Private Sub BtnExit_Click()
    End
End Sub
作成
Private Sub BtnIns_Click()

    Dim Isendbuf As Long
    Dim Irecvbuf As Long
    Dim Irbuflen As Long
    Dim Iret As Long
    Dim text As String
    Dim value As Long
    Dim dvalue As Double
    Dim svalue As Single
    Dim lenL As Long
    Dim txbool As Integer ' トランザクションが開始されると1,されなければ0 '

    ' tpstart '
    tmaxStart

    ' トランザクションの開始有無を初期化 '
    txbool = 0

    If EditEmpNo.text = "" Then
        MsgBox "社員番号を入力してください。"
        tmaxEnd
        Exit Sub
    End If

    ' 送信するデータのためのバッファ割り当て '
    Isendbuf = fmalloc(100, 1024)

```

```

If Isendbuf = Null Then
    TmaxError ("fballoc")
    Call fbfree(Isendbuf)
    tmaxEnd
    Exit Sub
End If

' 受信するデータのためのバッファ割り当て '
Irecvbuf = fballoc(100, 1024)
If Irecvbuf = Null Then
    TmaxError ("fballoc")
    Call fbfree(Irecvbuf)
    tmaxEnd
    Exit Sub
End If

Iret = PUTLONG(ByVal Isendbuf, "EMPNO", 0, Trim(EditEmpNo.text))
If Iret = -1 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

Iret = PUTVAR(ByVal Isendbuf, "ENAME", 0, Trim(EditName.text))
If Iret = -1 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

Iret = PUTVAR(ByVal Isendbuf, "JOB", 0, Trim(EditJob.text))
If Iret = -1 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

Iret = PUTLONG(ByVal Isendbuf, "MGR", 0, Trim(EditMgr.text))
If Iret = -1 Then      ' トランザクション開始 '
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

Iret = PUTVAR(ByVal Isendbuf, "DATE", 0, Trim(EditDate.text))
If Iret = -1 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

svalue = Trim(EditSal.text)
' fbput, fbget_fldkey関数はcomm.basファイルに定義されています。 '
' fbget_fldkey関数はフィールド名をキー値に変換します。 '
Iret = fbput(ByVal Isendbuf, ByVal fbget_fldkey("SAL"), svalue, ByVal lenL)
If Iret = -1 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)

```

```

End If

' 以下のコメント部分は、コメントの下記文と同様な機能ができます。 '
'svalue = Trim(EditComm.text)
'Iret = fbput(ByVal Isendbuf, ByVal fbget_fldkey("COMM"), svalue, ByVal lenL)

'If Iret = -1 Then
'    FdlErrorMsg ("fbput")
'    Exit Sub
'End If

' 上記のコメントと同様な機能をします。 '
Iret = PUTFLOAT(ByVal Isendbuf, "COMM", 0, Trim(EditComm.text))
If Iret = -1 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

Iret = PUTLONG(ByVal Isendbuf, "DEPTNO", 0, Trim(EditDept.text))
If Iret = -1 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

' トランザクション開始 '
Iret = tx_begin
txbool = 1
If Iret < 0 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

' 修正サービスを要求 '
Iret = tpcall(ByVal "FDLINSERT", ByVal Isendbuf, ByVal 0, Irecvbuf,
              Irbuflen, ByVal 0)

LabelErr.Caption = "sbuf = " & Isendbuf & ", rbuf = " & Irecvbuf

If Iret = -1 Then
    TmaxError ("tpcall(SVC1)")
    ViewErr (Irecvbuf)
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

' トランザクション保存 '
Iret = tx_commit
If Iret < 0 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
Else

```

```

        txbool = 1
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    End If

End Sub

' Oracleのエラーを出力する関数 '
' 照会ボタン・クリック時のイベント・ハンドラー '
Private Sub BtnSel_Click()

    Hide
    ' 照会結果画面を表示 '
    EmployeeGrid.Show

End Sub

' 修正ボタン・クリック時のイベント・ハンドラー '
Private Sub BtnUdt_Click()

    Dim Isendbuf As Long
    Dim Irecvbuf As Long
    Dim Irbuflen As Long
    Dim Iret As Long
    Dim text As String
    Dim value As Long
    Dim dvalue As Double
    Dim svalue As Single
    Dim lenL As Long
    Dim txbool As Integer ' トランザクションが開始されると1、されなければ0 '

    ' tpstart '
    tmaxStart

    ' トランザクションの開始有無を初期化 '
    txbool = 0

    If EditEmpNo.text = "" Then
        MsgBox "社員番号を入力してください。"
        tmaxEnd
        Exit Sub
    End If

    ' 送信するデータのためのバッファ割り当て '
    Isendbuf = fmalloc(100, 1024)
    If Isendbuf = Null Then
        TmaxError ("fmalloc")
        Call fbfree(Isendbuf)
        tmaxEnd
        Exit Sub
    End If

```

```

End If

' 受信するデータのためのバッファ割り当て '
Irecvbuf = fmalloc(100, 1024)
If Irecvbuf = Null Then
    TmaxError ("fmalloc")
    Call fbfree(Irecvbuf)
    tmaxEnd
    Exit Sub
End If

Iret = PUTLONG(ByVal Isendbuf, "EMPNO", 0, Trim(EditEmpNo.text))
If Iret = -1 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

Iret = PUTVAR(ByVal Isendbuf, "ENAME", 0, Trim(EditName.text))
If Iret = -1 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

Iret = PUTVAR(ByVal Isendbuf, "JOB", 0, Trim(EditJob.text))
If Iret = -1 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

Iret = PUTLONG(ByVal Isendbuf, "MGR", 0, Trim(EditMgr.text))
If Iret = -1 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

Iret = PUTVAR(ByVal Isendbuf, "DATE", 0, Trim(EditDate.text))
If Iret = -1 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

svalue = Trim(EditSal.text)
' Declare Function fbput Lib "TMAX4GL.DLL"
    (ByVal pFBUF As Long, ByVal fieldid As Long, pBuffer As Any,
    ByVal Fieldlen As Long) As Long
Iret = fbput(ByVal Isendbuf, ByVal fbget_fldkey("SAL"), svalue, ByVal lenL)
If Iret = -1 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

svalue = Trim(EditComm.text)

```



```

Iret = fbput(ByVal Isendbuf, ByVal fbget_fldkey("COMM"), svalue, ByVal lenL)
If Iret = -1 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

Iret = PUTLONG(ByVal Isendbuf, "DEPTNO", 0, Trim(EditDept.text))
If Iret = -1 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

' トランザクション開始 '
Iret = tx_begin
txbool = 1
If Iret < 0 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

' 修正サービスを要求 '
Iret = tpcall(ByVal "FDLUPDATE", ByVal Isendbuf, ByVal 0, Irecvbuf, Irbuflen,

                ByVal 0)

LabelErr.Caption = "sbuf = " & Isendbuf & ", rbuf = " & Irecvbuf

If Iret = -1 Then
    TmaxError ("tpcall(SVC1)")
    ViewErr (Irecvbuf)
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

' トランザクション保存 '
Iret = tx_commit
If Iret < 0 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
Else
    txbool = 1
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

End Sub

Private Sub ExitSub(txbool As Integer, Isendbuf As Long, Irecvbuf As Long)
    If txbool = 1 Then
        tx_rollback
    End If

    ' 割り当てられたバッファを解除 '

```

```

    Call fbfree(ByVal Isendbuf&)
    Call fbfree(ByVal Irecvbuf&)

    ' tpend実行関数 '
    tmaxEnd

Exit Sub

End Sub

' tpstart '
Private Sub tmaxStart()
    Dim ret As Integer

    ' 環境設定ファイルを呼び出します。 '
    ' atmi.basファイルに定義されています。 '
    ' Declare Function tmaxreadenv Lib "TMAX4GL.DLL" (ByVal envfile As String,
                                                ByVal label As String)
                                                As Long '
    ' 詳細説明については、Tmaxリファレンスガイドを参照してください。 '
    ret = tmaxreadenv("C:\tmax.env", "aix51389")
    If ret < 0 Then
        TmaxError ("tmaxreadenv")
        Exit Sub
    End If

    ret = tpstart(ByVal 0&)
    If ret = -1 Then
        LabelErr.Caption = "tpstart error = " & gettperrno()
        TmaxError ("tpstart")
        Exit Sub
    Else
        LabelErr.Caption = "tpstart return value = " & ret & " tpstart success"
    End If

End Sub

' tpend '
Private Sub tmaxEnd()
    Dim ret As Integer
    ret = tpend()
    If ret = -1 Then
        LabelErr.Caption = "tpend error = " & gettperrno()
        TmaxError ("tpend")
        Exit Sub
    Else
        LabelErr.Caption = "tpreturn return value = " & ret
    End If

```

```
End Sub
```

Oracleのエラー出力

下記は、Oracleのエラー出力画面です。



下記は、Oracleのエラー出力画面を実行するために必要な環境設定の例です。

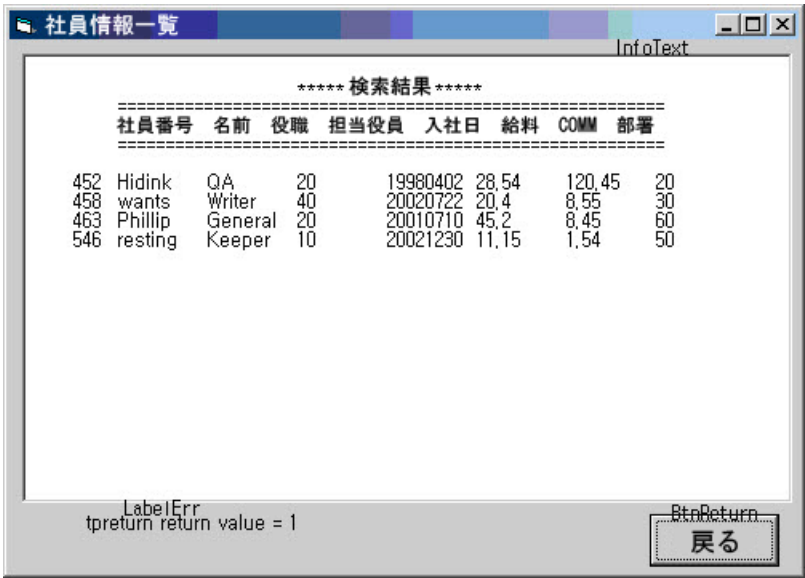
```
' Oracleのエラーを出力する関数 '  
Private Sub ViewErr(ByVal a As Long)  
  
    'E_TYPE          9009    long    -    -  
    'E_CODE          9010    long    -    -  
    'E_MSG            9011    string  -    -  
    'E_TMP            9012    long    -    -  
  
    Dim typeL, codeL, tmpL As Long  
    Dim msgS As String  
    Dim Iret As Integer  
  
    ' Tmax FDLリファレンスガイドを参照 '  
    ' comm.basに関数が定義されている '  
    Iret = fbget_tu(ByVal a, ByVal fbget_fldkey("E_TYPE"), 0, typeL, 0)  
    If Iret = -1 Then  
        FdlErrorMsg ("fbget_tu")  
        Exit Sub  
    End If  
  
    Iret = fbget_tu(ByVal a, ByVal fbget_fldkey("E_CODE"), 0, codeL, 0)  
    If Iret = -1 Then  
        FdlErrorMsg ("fbget_tu")  
        Exit Sub  
    End If  
  
    Iret = GETVAR(ByVal a, "E_MSG", 0, msgS)  
    If Iret = -1 Then  
        TmaxError ("ViewErr, GETVAR, E_MSG")  
        Exit Sub  
    End If作成
```

```
Iret = fbget_tu(ByVal a, ByVal fbget_fldkey("E_TMP"), 0, tmpL, 0)
If Iret = -1 Then
    FdlErrorMsg ("fbget_tu")
    Exit Sub
End If
' エラー・メッセージをボックスで表示する '
MsgBox ("Error Type : " & typeL & " ,Code : " & codeL & " ,Message :
        " & msgS & " ,Tmp : " & tmpL)

End Sub
```

社員照会画面

下記は、社員照会画面フォームのデザイン画面です。



下記は、照会画面フォームのデザイン構成表です。

制御名	制御タイプ	修正が必要なプロパティ
EmployeeGrid	フォーム	Caption="社員情報一覧"
InfoText	テキストボックス	MultiLine=True
BtnReturn	コマンドボタン	Caption="戻る"

下記は、照会画面フォームのデザイン画面を実行するために必要な環境設定の例です。

```
EmployeeGrid.frm Source LabelErr

Option Explicit
' 照会画面を閉じる '
```

```

Private Sub BtnReturn_Click()
    Hide
    EmployeeMgr.Show

End Sub
' 照会ボタン・クリック時に照会一覧を表示する関数 '
Private Sub Form_Activate()

Dim Isendbuf As Long
    Dim Irecvbuf As Long
    Dim Irbuflen As Long
    Dim Iret As Integer
    Dim text As String
    Dim value As Long
    Dim dvalue As Double
    Dim svalue As Single
    Dim lenL As Long
    Dim initS, outputS As String
    Dim empnoS, enameS, jobS, mgrS, dateS, sals, commS, deptnoS As String
    Dim cntL As Long
    Dim eNo As Long
    Dim txbool As Integer ' トランザクションが開始されると1, されなければ0 '

    ' tpstart '
    tmaxStart

    ' トランザクションの開始有無を初期化 '
    txbool = 0

    initS = vbCrLf & vbTab & vbTab & vbTab & "***** 検索結果 *****"
        & vbCrLf & vbTab & "===== "

        & vbCrLf & vbTab & "社員番号 名前 役職 担当役員 入社日 給料 COMM 部署"
        & vbCrLf & vbTab & "===== "

        & vbCrLf & vbCrLf

    If EmployeeMgr.EditEmpNo.text = "" Then
        MsgBox "社員番号を入力してください。"
        Hide
        EmployeeMgr.Show
        Exit Sub
    End If

    ' 送信するデータのためのバッファ割り当て '
    Isendbuf = fmalloc(100, 1024)
    If Isendbuf = Null Then

```

```

        TmaxError ("fballoc")
        Call fbfree(Isendbuf)
        tmaxEnd
    Exit Sub
End If

' 受信するデータのためのバッファ割り当て '
Irecvbuf = fballoc(100, 1024)
If Irecvbuf = Null Then
    TmaxError ("fballoc")
    Call fbfree(Irecvbuf)
    tmaxEnd
    Exit Sub
End If

' 社員管理フォームから社員番号を取得します。
eNo = Trim(EmployeeMgr.EditEmpNo.text)
Iret = fbput(ByVal Isendbuf, ByVal fbget_fldkey("EMPNO"), eNo, ByVal lenL)
If Iret = -1 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

' 照会サービスを要求 '
Iret = tpcall(ByVal "FDLSELECT", ByVal Isendbuf, ByVal 0, Irecvbuf, Irbuflen,

                ByVal 0)
If Iret = -1 Then
    ViewErr (Irecvbuf)
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

cntL = fbkeyoccur(ByVal Irecvbuf, ByVal fbget_fldkey("EMPNO"))

Dim i As Long

For i = 0 To cntL - 1

    Iret = GETLONG(ByVal Irecvbuf, "EMPNO", i, value)
    If Iret = -1 Then
        TmaxError ("GETLONG(EMPNO)")
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    End If
    empnoS = value

    Iret = GETVAR(ByVal Irecvbuf, "ENAME", i, text)
    If Iret = -1 Then
        TmaxError ("GETVAR(ENAME)")
    End If

```

```

        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    End If
    enameS = text

    Iret = GETVAR(ByVal Irecvbuf, "JOB", i, text)
    If Iret = -1 Then
        TmaxError ("GETVAR(JOB)")
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    End If
    jobS = text

    Iret = GETLONG(ByVal Irecvbuf, "MGR", i, value)
    If Iret = -1 Then
        TmaxError ("GETLONG(MGR)")
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    End If
    mgrS = value

    Iret = GETVAR(ByVal Irecvbuf, "DATE", i, text)
    If Iret = -1 Then
        TmaxError ("GETVAR(DATE)")
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    End If
    dateS = text

    ' SALデータフィールドからfloatデータを取得します。 '
    ' comm.basに関数が定義されています。 '
    Iret = fbget_tu(ByVal Irecvbuf, ByVal fbget_fldkey("SAL"), i, svalue, 0)
    If Iret = -1 Then
        FdlErrorMsg ("fbget_tu")
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    End If
    salS = svalue

    Iret = GETFLOAT(ByVal Irecvbuf, "COMM", i, svalue)
    If Iret = -1 Then
        TmaxError ("GETFLOAT(COMM)")
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    End If
    commS = svalue

    Iret = GETLONG(ByVal Irecvbuf, "DEPTNO", i, value)
    If Iret = -1 Then
        TmaxError ("GETLONG(DEPTNO)")
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    End If
    deptnoS = value & vbCrLf

```

```

        outputS = outputS & "          " & empnoS & vbTab & enameS & vbTab & jobS
                                & vbTab & mgrS & vbTab & dateS & vbTab
                                & salS & vbTab & commS & vbTab & deptnoS

    Next i

    InfoText.text = initS & outputS
    ' 返されたデータをテキストボックスに出力 '
    LabelErr.Caption = "SAL = " & dvalue

    Call ExitSub(txbool, Isendbuf, Irecvbuf)

End Sub

Private Sub ExitSub(txbool As Integer, Isendbuf As Long, Irecvbuf As Long)
    If txbool = 1 Then
        tx_rollback
    End If

    ' 割り当てられたバッファを解除 '
    Call fbfree(ByVal Isendbuf&)
    Call fbfree(ByVal Irecvbuf&)

    ' tpend実行関数 '
    tmaxEnd

    Exit Sub

End Sub

' tpstart '
Private Sub tmaxStart()
    Dim ret As Integer

    ' 環境設定ファイルを呼び出します。 '
    ' atmi.basファイルに定義されています。 '
    ' Declare Function tmaxreadenv Lib "TMAX4GL.DLL"(ByVal envfile As String,
                                                    ByVal label As String) As Long
    '
    ' 詳細説明については、Tmax リファレンスガイドを参照してください。 '
    ret = tmaxreadenv("C:\tmax.env", "aix51389")
    If ret < 0 Then
        TmaxError ("tmaxreadenv")
        Exit Sub
    End If

```



```

    ret = tpstart(ByVal 0&)
    If ret = -1 Then
        LabelErr.Caption = "tpstart error = " & gettperrno()
        TmaxError ("tpstart")
        Exit Sub
    Else
        LabelErr.Caption = "tpstart return value = " & ret & " tpstart success"
    End If

End Sub

' tpend '
Private Sub tmaxEnd()
    Dim ret As Integer
    ret = tpend()
    If ret = -1 Then
        LabelErr.Caption = "tpend error = " & gettperrno()
        TmaxError ("tpend")
        Exit Sub
    Else
        LabelErr.Caption = "tpreturn return value = " & ret
    End If
End Sub

' Oracleのエラーを出力する関数 '
Private Sub ViewErr(ByVal a As Long)

'E_TYPE          9009    long    -    -
'E_CODE          9010    long    -    -
'E_MSG           9011    string  -    -
'E_TMP           9012    long    -    -

    Dim typeL As Long
    Dim codeL As Long
    Dim msgS As String
    Dim tmpL As Long
    Dim Iret As Integer

    Iret = fbget_tu(ByVal a, ByVal fbget_fldkey("E_TYPE"), 0, typeL, 0)

    If Iret = -1 Then
        FdlErrorMsg ("fbget_tu")
        Exit Sub
    End If
    Iret = fbget_tu(ByVal a, ByVal fbget_fldkey("E_CODE"), 0, codeL, 0)
    If Iret = -1 Then
        FdlErrorMsg ("fbget_tu")
        Exit Sub
    End If

```

```

Iret = GETVAR(ByVal a, "E_MSG", 0, msgS)
Iret = fbget_tu(ByVal a, ByVal fbget_fldkey("E_TMP"), 0, tmpL, 0)
If Iret = -1 Then
    FdlErrorMsg ("fbget_tu")
    Exit Sub
End If
MsgBox ("Error Type : " & typeL & " ,Code : " & codeL & " ,Message : "&msgS&"

        ,Tmp : " & tmpL)
End Sub

```

2.3.5. サーバー・プログラム

サービス・プログラム

下記は、サービス・プログラムの例です。

<emp_c.pc>

```

#include <stdio.h>
#include <ctype.h>
#include <tuxinc/macro.h>
#include "../fdl/demo_fdl.h"

EXEC SQL include sqlca.h;
EXEC SQL INCLUDE ORACA;
EXEC ORACLE  OPTION (ORACA=YES);
EXEC ORACLE  OPTION (RELEASE_CURSOR=YES);

#define INP    1
#define ORA    2
#define TMX    3
#define APP    4

EXEC SQL begin declare section;
int    h_empno;
char   h_ename[11];
char   h_job[10];
int    h_mgr;
char   h_date[11];
float  h_sal;
float  h_comm;
int    h_deptno;
EXEC SQL end declare section;

void svc_error(long type, long err_code, char *msg, long tmp);

```

```

FDLINSERT( TPSVCINFO *msg )
{
    FBUF *rcvbuf;
    int i, occurrence;
    rcvbuf = (FBUF *)msg->data;
    h_empno = h_mgr = h_sal = h_comm = h_deptno = 0;

    memset(h_ename, 0x00, sizeof(h_ename));
    memset(h_job, 0x00, sizeof(h_job));
    memset(h_date, 0x00, sizeof(h_date));

    occurrence = fbkeyoccur(rcvbuf, EMPNO);

    for (i=0; i< occurrence; i++){
        fbget_tu(rcvbuf, EMPNO, i, (char *)&h_empno, 0);
        fbget_tu(rcvbuf, MGR, i, (char *)&h_mgr, 0);
        fbget_tu(rcvbuf, SAL, i, (char *)&h_sal, 0);
        fbget_tu(rcvbuf, COMM, i, (char *)&h_comm, 0);
        fbget_tu(rcvbuf, DEPTNO, i, (char *)&h_deptno, 0);
        fbget_tu(rcvbuf, ENAME, i, (char *)h_ename, 0);
        fbget_tu(rcvbuf, JOB, i, (char *)h_job, 0);
        fbget_tu(rcvbuf, DATE, i, (char *)h_date, 0);

        EXEC SQL INSERT
        INTO emp(empno, ename, job, mgr, hiredate, sal, comm, deptno)
        VALUES (:h_empno, :h_ename, :h_job, :h_mgr,
                to_date(:h_date, 'yyyymmdd'), :h_sal, :h_comm, :h_deptno );
    }

    if(sqlca.sqlcode != 0)
        goto LB_DBERROR;

    EXEC SQL WHENEVER SQLERROR
        goto LB_DBERROR;

    tpreturn(TPSUCCESS, 0L, (char *)rcvbuf, 0L, 0L);

    LB_DBERROR :
        EXEC SQL WHENEVER SQLERROR CONTINUE;
        svc_error(ORA, sqlca.sqlcode, sqlca.sqlerrm.sqlerrmc, 0);
}

FDLDELETE( TPSVCINFO *msg )
{
    FBUF *rcvbuf;
    int i, occurrence;

    rcvbuf = ( FBUF *)msg->data;
    occurrence = fbkeyoccur(rcvbuf, EMPNO);

```

```

    for (i=0; i< occurrence; i++){
        fbget_tu (rcvbuf, EMPNO, i, (char *)&h_empno , 0);

        EXEC SQL DELETE
        FROM emp
        WHERE empno = :h_empno;
    }
    if(sqlca.sqlcode != 0)
        goto LB_DBERROR;

    EXEC SQL WHENEVER SQLERROR
        goto LB_DBERROR ;

    EXEC SQL WHENEVER NOT FOUND
        goto LB_DBERROR ;

    tpreturn(TPSUCCESS, 0L, (char *)rcvbuf, 0L, 0L);

    LB_DBERROR :
        EXEC SQL WHENEVER SQLERROR CONTINUE;
        svc_error(ORA, sqlca.sqlcode, sqlca.sqlerrm.sqlerrmc, 0);
}

FDLSELECT( TPSVCINFO *msg )
{
    FBUF *rcvbuf;
    FLDLEN fldlen;
    int i=0, ROWMEM=200, CHKROW=500;

    rcvbuf = (FBUF *)msg->data;

    if((rcvbuf=(FBUF *)tprealloc((char *)rcvbuf,ROWMEM*CHKROW))==NULL) {
        rcvbuf=(FBUF *)msg->data;
        goto LB_TMAXERROR ;
    }

    h_empno = h_mgr = h_sal = h_comm = h_deptno = 0;

    memset( h_ename, 0x00, sizeof( h_ename ) );
    memset( h_job, 0x00, sizeof( h_job ) );
    memset( h_date, 0x00, sizeof( h_date ) );

    fbget_tu( rcvbuf, EMPNO, 0, (char *)&h_empno, &fldlen);

    EXEC SQL DECLARE DB_CUR CURSOR FOR
    SELECT  nvl(empno,0), nvl(ename,' '), nvl(job,' '),
            nvl(to_char(hiredate,'yyyymmdd'),' '),  nvl(mgr,0),
            nvl(sal,0), nvl(comm,0), nvl(deptno,0)
    FROM    emp
    WHERE   empno >= :h_empno-50 AND empno <= :h_empno+50;
    EXEC SQL OPEN DB_CUR;

```

```

if(sqlca.sqlcode != 0)
    goto LB_DBERROR;

EXEC SQL WHENEVER SQLERROR
    goto LB_DBERROR ;
EXEC SQL WHENEVER NOT FOUND
    Do break ;

while(1) {
    EXEC SQL FETCH DB_CUR
    INTO      :h_empno,
              :h_ename,
              :h_job,
              :h_date,
              :h_mgr,
              :h_sal,
              :h_comm,
              :h_deptno;

    fbchg_tu(rcvbuf, EMPNO, i, (char *)&h_empno, 0);
    fbchg_tu(rcvbuf, MGR, i, (char *)&h_mgr, 0);
    fbchg_tu(rcvbuf, SAL, i, (char *)&h_sal, 0);
    fbchg_tu(rcvbuf, DEPTNO, i, (char *)&h_deptno, 0);
    fbchg_tu(rcvbuf, COMM, i, (char *)&h_comm, 0);
    fbchg_tu(rcvbuf, ENAME, i, (char *)&h_ename, 0);
    fbchg_tu(rcvbuf, JOB, i, (char *)&h_job, 0);
    fbchg_tu(rcvbuf, DATE, i, (char *)&h_date, 0);

    i++;
}

if (i < 1) goto LB_DBERROR;

EXEC SQL CLOSE DB_CUR;

tpreturn(TPSUCCESS, 0L, (char *)rcvbuf, 0L, 0L);

LB_DBERROR :
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    EXEC SQL CLOSE DB_CUR ;
    svc_error(ORA, sqlca.sqlcode, sqlca.sqlerrm.sqlerrmc, 0L) ;

LB_TMAXERROR :
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    EXEC SQL CLOSE DB_CUR;
    svc_error(TMX, tperrno, "", 0L);
}

FDLUPDATE( TPSVCINFO *msg )
{

```

```

    FBUF *rcvbuf ;
    int i, occurrence;
    rcvbuf = (FBUF *)msg->data;
    h_empno = h_mgr = h_sal = h_comm = h_deptno = 0;

    memset( h_ename, 0x00, sizeof( h_ename ) );
    memset( h_job, 0x00, sizeof( h_job ) );
    memset( h_date, 0x00, sizeof( h_date ) );

    occurrence = fbkeyoccur(rcvbuf, EMPNO);

    for (i=0; i< occurrence; i++){
        fbget_tu ( rcvbuf, EMPNO, i, (char *)&h_empno, 0);
        fbget_tu ( rcvbuf, ENAME, i, (char *)h_ename, 0);
        fbget_tu ( rcvbuf, JOB, i, (char *)h_job, 0);
        fbget_tu ( rcvbuf, MGR, i, (char *)&h_mgr, 0);
        fbget_tu ( rcvbuf, SAL, i, (char *)&h_sal, 0);
        fbget_tu ( rcvbuf, COMM, i, (char *)&h_comm, 0);
        fbget_tu ( rcvbuf, DEPTNO, i, (char *)&h_deptno, 0);
        fbget_tu ( rcvbuf, DATE, i, (char *)h_date, 0);

        EXEC SQL UPDATE emp
        SET ename = nvl(:h_ename, ename),
            job   = nvl(:h_job, job),
            mgr    = :h_mgr,
            hiredate = nvl(to_date(:h_date,
                                'yyyymmdd'),hiredate),
            sal    = :h_sal,
            comm   = :h_comm,
            deptno = :h_deptno
        WHERE empno = :h_empno;

        if(sqlca.sqlcode != 0)
            goto LB_DBERROR;

        EXEC SQL WHENEVER SQLERROR
            goto LB_DBERROR;

        EXEC SQL WHENEVER NOT FOUND
            goto LB_DBERROR;
    }
    tpreturn(TPSUCCESS, 0L, (char *)rcvbuf, 0L, 0L);

    LB_DBERROR :
        EXEC SQL WHENEVER SQLERROR CONTINUE;
        svc_error(ORA, sqlca.sqlcode, sqlca.sqlerrm.sqlerrmc, 0L) ;
}

/*****
* エラー処理:サービスでエラーの発生時にそのエラーをバッファに入れてクライアントに転送します。
*****/

```

```

void svc_error(long type, long err_code, char *msg, long tmp) {
    FBUFF *transf;
    char *svcname;
    char err_msg[100];
    char temp[100];
    int i = 0;

    printf("type      ==>[%ld]\n", type);
    printf("err_code ==>[%ld]\n", err_code);
    printf("msg        ==>[%s]\n",  msg);
    strcpy(err_msg, msg);

    if ((transf = (FBFR *)tpalloc("FML", NULL, 0)) == NULL) {
        printf("tpalloc failed! errno = %d\n", tperrno);
    }

    switch(type) {
        case INP:
            switch(err_code) {
                case -1000:
                    strcpy(err_msg, "当該ユーザーに権限がありません。");
                    break;
                default:
                    strcpy(err_msg,
                        "Inputエラー・メッセージが登録されていません。");
            }
            break;
        case ORA:
            if (strlen(err_msg)== 0)
                strcpy(err_msg, sqlca.sqlerrm.sqlerrmc);
            break;
        case TMX:
            if (strlen(err_msg)== 0)
                strcpy(err_msg, tpstrerror(tperrno));
            break;
        case APP:
            if (strlen(err_msg)== 0) {
                /* err_mssg=""の場合はエラー・メッセージを設定します。 *****/
                switch(err_code) {
                    case -500: /* システム関連エラー */
                        strcpy(err_msg, "ファイル作成エラーです。");
                        break;
                    case -502:
                        strcpy(err_msg, "内部サービスが呼び出せませんでした。");
                        break;
                    case -504:
                        strcpy(err_msg, "ソケット通信エラーです。");
                        break;
                    case -505: /* 他のトランザクションで修正された場合はMSG処理*/
                        /* "照会の後、他の場所で[%s]データが変更されました。
                        \n\nを再照会した後処理してください。": クライアントで処理*/

```

```

        strcpy(err_msg, "がありません。");
        break;
    case -5002:
        strcpy(err_msg, "電算室にお問い合わせください。");
        break;
    default:
        strcpy(err_msg,
            "アプリケーション・エラー・メッセージが登録されていません。");
    }
}
break;
}

/* ROLLBACK */
EXEC SQL WHENEVER SQLERROR CONTINUE;
EXEC SQL ROLLBACK;

fbchg_tu ( transf, E_TYPE, i, (char *)&type,0);
fbchg_tu ( transf, E_CODE, i, (char *)&err_code,0);
fbchg_tu ( transf, E_MSG, i, (char *)&err_msg,0);
fbchg_tu ( transf, E_TMP, i, (char *)&tmp,0);

tpreturn(TPFAIL, 0, (char *)transf, 0L, 0L);
}

```

メイクファイル(Makefile)

下記は、emp_c.pcソースをTmaxアプリケーションに作成するメイクファイルの例です。

<emp_c.mk>

```

include $(ORACLE_HOME)/precomp/lib/env32.mk
ORALIBDIR = $(LIBHOME)
ORALIB = -L/home/oracle/OraHome/lib32/ -lclntsh -lld -lm `cat
        /home/oracle/OraHome/lib32/sysliblist` -lm -lc_r -lpthreads
TARGET = emp_c
APOBJS = emp_c.o
NSDLOBJ = $(TMAXDIR)/lib/sdl.o
#CC
CC=cc

#Oracle
LIBS = -lsvr -loras

OBS = $(APOBJS) $(SVCTOBJ)
SVCTOBJ = $(TARGET)_svctab.o

CFLAGS = -q32 -O -I$(TMAXDIR)

```



```

LDFLAGS = -brtl
APPDIR  = $(TMAXDIR)/appbin
SVCTDIR = $(TMAXDIR)/svct
TMAXLIBDIR = $(TMAXDIR)/lib
#
.SUFFIXES : .c

.c.o:
    $(CC) $(CFLAGS) $(LD_FLAGS) -c $<

all: $(TARGET)

$(TARGET): $(OBJS)
    $(CC) $(CFLAGS) $(LD_FLAGS) -L$(TMAXLIBDIR) -o $(TARGET)
    -L$(ORALIBDIR) $(ORALIB) $(OBJS) $(LIBS) $(NSDLOBJ)
    mv $(TARGET) $(TMAXDIR)/appbin

$(APOBJS): $(TARGET).pc
    proc iname=emp_c include=$(TMAXDIR)
    define=__LINUX_ORACLE_PROC__
    $(CC) $(CFLAGS) $(LD_FLAGS) -c $(TARGET).c

$(SVCTOBJ):
    touch $(SVCTDIR)/$(TARGET)_svctab.c
    $(CC) $(CFLAGS) $(LD_FLAGS) -c $(SVCTDIR)/$(TARGET)_svctab.c
#
clean:
    -rm -f *.o core $(TARGET) $(TARGET).lis

```


第3章 Delphiインターフェース

本章では、Delphiインターフェース・モジュールと例について説明します。

3.1. 概要

Delphiインターフェースには、Visual Basicと類似した形式で、クライアント・ライブラリーで提供する関数を呼び出すためのインターフェースが存在します。開発者は、インターフェース・モジュールをインストールすればすべての関数を使用できます。DelphiはPowerBuilderやVisual Basicとは違ってポインターの概念がありません。値の転送方法がC言語と類似しており、Pascalの文法に従っているため、既存の言語に慣れているユーザーなら問題なく使用できます。そのため、別途のマクロやコンポーネントはありません。

Delphiインターフェース・コンポーネントは下記のとおりです。

モジュール	説明
atmi.pas	atmi関数のプロトタイプ定義ファイルです
fdl.pas	フィールド・キー関数のプロトタイプ定義ファイルです

参考

atmi関数とフィールド・キー関数のプロトタイプおよび機能についての説明は、『Tmaxリファレンスガイド』および『Tmax アプリケーション開発ガイド』を参照してください。Delphiインターフェースの使用方法はサンプル・プログラムを利用して説明します。

3.2. サンプル・プログラム

社員番号を入力すると、Oracleデータベースから名前、所属、部署名などをルックアップして読み込むプログラムです。

3.2.1. プログラムの構成

atmi.pas、fdl.pas、TuxSvc.pasをモジュールとしてプロジェクトで追加します。

- 共通プログラム

プログラムファイル	説明
demo.f	フィールド・キー・バッファーを定義したファイルです

プログラムファイル	説明
tmax	ライブラリー・ファイルです

- クライアント・プログラム

プログラムファイル	説明
Atmi.dcu	atmiソースファイルをコンパイルして作成されるオブジェクト・ファイルです
Atmi.pas	atmi関数のプロトタイプを定義したファイルです
EmployeeMgr.bpg	プロジェクトのグループ・ファイルです
EmployeeMgr.cfg	プロジェクトの環境設定ファイルです
EmployeeMgr.dof	Delphiオプション・ファイルです
EmployeeMgr.dpr	複数のpasファイルとdfmファイルの情報を有するプロジェクト・ファイルです
EmployeeMgr.exe	オブジェクト・ファイルを実行可能なファイルに作成します
EmployeeMgr.res	コンパイルされた2進数リソースファイルです
Fdl.dcu	Fdlソースファイルをコンパイルして作成されるオブジェクト・ファイルです
Fdl.pas	フィールド・キー関数のプロトタイプを定義したファイルです
TuxSvc.pas	Tuxedo変換用のソースファイルです
main.dcu	メイン・ソースファイルをコンパイルして作成されるオブジェクト・ファイルです
main.dfm	メイン・フォーム・ファイルです
main.pas	クライアント・プログラムです

- サーバー・プログラム

プログラムファイル	説明
emp_c.mk	メイクファイルです
emp_c.pc	サーバー・プログラムとしてAIXとOracle 9iを使用します
employee.m	Tmax環境設定ファイルです

3.2.2. プログラムの特徴

- クライアント・プログラム

機能	説明
Tmax接続	Tmaxユーザー・アカウントとアプリケーション定義ユーザー名を引数として接続します

機能	説明
バッファ・タイプ	フィールド・キー・バッファ、フィールド・キー・ファイルをfdlcユーティリティでコンパイルして「fdl」ファイルを作成する必要があります
通信タイプ	tpcall()を使用する同期通信です。送受信バッファが同じです
トランザクションの有無	TMSで自動トランザクション(AutoTransaction)を付与します

- サーバー・プログラム

機能	説明
サービス	FDLSELECT、FDLDELETE、FDLUPDATE、FDLINSERTを作成します
データベース指定	Oracleデータベースを使用します。システム構成ファイルのSVRGROUPにデータベース情報を指定します

3.2.3. 共通プログラム

データベースのEMP表

下記は、データベースを操作するための基本表の例です。

EMPNO	NUMBER	NOT NULL	P1
ENAME	VARCHAR(16)		
JOB	VARCHAR(16)		
MGR	NUMBER		
HIREDATE	DATE		
SAL	NUMBER(7,2)		
COMM	NUMBER(7,2)		
DEPTNO	NUMBER		

フィールド・キー・バッファの定義

下記は、フィールド・キー・バッファを定義したファイルの例です。

<demo.f>

#For tmax demo employee program				
EMPNO	7500	long	-	-
ENAME	7501	string	-	-
JOB	7502	string	-	-
MGR	7503	long	-	-
DATE	7504	string	-	-
SAL	7505	float	-	-

COMM	7506	float	-	-
DEPTNO	7507	long	-	-
E_TYPE	9009	long	-	-
E_CODE	9010	long	-	-
E_MSG	9011	string	-	-
E_TMP	9012	long	-	-

Tmax環境設定

下記は、Tmax環境設定ファイルの例です。

<tmconfig.m>

```
*DOMAIN
dom1      SHMKEY = 70000,  MAXUSER = 200, MINCLH = 1, MAXCLH = 5,
          TPORTNO = 8888, BLOCKTIME = 200, TXTIME = 200

*NODE
tmax1     TMAXDIR = "/home/tmax",
          APPDIR = "/home/tmax/appbin",
          PATHDIR = "/home/tmax/path",
          TLOGDIR = "/home/tmax/log/tlog",
          SLOGDIR = "/home/tmax/log/slog"
          ULOGDIR = "/home/tmax/log/ulog"

*SVRGROUP
svg1      NODENAME = tmax1, DBNAME = ORACLE,
          OPENINFO = "ORACLE_XA+Acc=P/scott/tiger+SesTm=60",
          TMSNAME = svg1_tms

*SERVER
emp_c     SVGNAME = svg1, MIN = 1

*SERVICE
FDLSELECT SVRNAME = emp_c
FDLUPDATE SVRNAME = emp_c
FDLDELETE SVRNAME = emp_c
FDLINSERT SVRNAME = emp_c
```

3.2.4. クライアント・プログラム

メイン画面フォームのデザイン

下記は、メイン画面フォームのデザイン画面です。

下記は、例で使用するメイン画面フォームのデザイン構成表です。

制御名	制御タイプ	備考
EmployeeMgrForm	Form	Caption="社員管理プログラム"
LabelErr	TLabel	Caption="エラー"
BtnExit	TButton	Caption="終了"
BtnIns	TButton	Caption="入力"
BtnDel	TButton	Caption="削除"
BtnUdt	TButton	Caption="修正"
BtnSel	TButton	Caption="照会"
EditName	TEdit	
EditEmpNo	TEdit	
EditDept	TEdit	
EditComm	TEdit	
EditSal	TEdit	
EditDate	TEdit	
EditMgr	TEdit	

制御名	制御タイプ	備考
EditJob	TEdit	
MList	TEdit	
BtnReturn	TEdit	

下記は、メイン画面フォームのデザイン画面を実行する例です。

<main.pas>

```
unit main;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;

type
  TEmployeeMgrForm = class(TForm)
    LabelEmpNo: TLabel;
    LabelName: TLabel;
    EditEmpNo: TEdit;
    EditName: TEdit;
    GroupBoxInfo: TGroupBox;
    LabelJob: TLabel;
    LabelMgr: TLabel;
    LabelDate: TLabel;
    LabelSal: TLabel;
    EditJob: TEdit;
    EditMgr: TEdit;
    EditDate: TEdit;
    EditSal: TEdit;
    EditComm: TEdit;
    EditDept: TEdit;
    LabelComm: TLabel;
    LabelDept: TLabel;
    BtnSel: TButton;
    BtnUdt: TButton;
    BtnDel: TButton;
    BtnIns: TButton;
    BtnExit: TButton;
    LabelErr: TLabel;
    MList: TMemo;
    BtnReturn: TButton;
    procedure BtnExitClick(Sender: TObject);
  end;
end;
```



```

        procedure BtnSelClick(Sender: TObject);
        procedure BtnUdtClick(Sender: TObject);
        procedure BtnDelClick(Sender: TObject);
        procedure BtnInsClick(Sender: TObject);
        procedure tmaxStart();
        procedure BtnReturnClick(Sender: TObject);
        procedure ViewErr(a:Pointer);

        { Private declarations }
    public
        { Public declarations }
    end;

var
    EmployeeMgrForm: TEmployeeMgrForm;

implementation

// Atmi, Fdl 使用
uses Atmi, Fdl;
const BufferSize = 1024;

{$R *.DFM}

// Stringに長さの値を省略
{$H+}

procedure TEmployeeMgrForm.tmaxStart();

var
    tpinfo: pTPSTART;
    ret: integer;

begin
    // atmi.pasファイルに定義されています。
    // Function tmaxreadenv(a:PChar; b:PChar):Integer; cdecl; external TmaxDLL;
    // 詳細説明については、Tmax リファレンスガイドを参照してください。
    ret := tmaxreadenv('C:\tmax.env', 'aix51389');
    if ret < 0 then begin
        ShowMessage('tmaxreadenv Error');
        Exit;
    end;
    // tpstart時にユーザー情報を送信するためのバッファを割り当て
    tpinfo := tpalloc('TPSTART', NIL, 0);

    if tpinfo = Nil then begin
        ShowMessage('tpinfo tpalloc failed,' + StrPas(tpstrerror(gettperrno)));
    end;
end;

```

```

        tpfree(tpinfo);
        Exit;
    end;
    //
    // 下記のようにオプションを指定することができます。
    //
    // ユーザー認証セキュリティのためのユーザー・アカウント
    // tpinfo.usrname := 'tmax';
    // 自主的なメッセージ受信時に使用される識別名
    // tpinfo.cltname := 'tmax';
    // 自主的なメッセージを許可
    // tpinfo.flags := TPUNSOL_POLL;

    // Tmax接続
    ret := tpstart(tpinfo);
    if ret < 0 then begin
        ShowMessage('tpstart failed' + StrPas(tpstrerror(gettperrno)));
        tpfree(tpinfo);
        tpend();
        Exit;
    end;
    // ユーザー情報を送信するためのバッファを解除
    tpfree(tpinfo);
end;

// 終了ボタンを押したとき
procedure TEmployeeMgrForm.BtnExitClick(Sender: TObject);
begin
    tpend();
    close;
end;

procedure TEmployeeMgrForm.BtnUdtClick(Sender: TObject);

var sndbuf, revbuf: Pointer;
    ret, empno_l, mgr_l, deptno_l: longint;
    sal_f, comm_f: single;
    job_s, ename_s: pointer;
    rlen: integer;
    date_s: string[100];

begin

    tmaxStart();

    // 入力するバッファを割り当て

```

```

sndbuf := fmalloc(1000, 10000);
if sndbuf = Nil then begin
    ShowMessage('sndbuf tmalloc failed, ' + StrPas(tpstrerror(gettperrno)));
    fbfree(sndbuf);
    tpend();
    Exit;
end;

// 受信するバッファを割り当て
revbuf := fmalloc(1000, 10000);
if revbuf = Nil then begin
    ShowMessage('revbuf tmalloc failed, ' + StrPas(tpstrerror(gettperrno)));
    fbfree(revbuf);
    tpend();
    Exit;
end;

empno_l := StrToInt(EditEmpNo.text);
fbput(sndbuf, fbget_fldkey('EMPNO'), @empno_l, 0);

ename_s := PChar(EditName.text);
fbput(sndbuf, fbget_fldkey('ENAME'), ename_s, 0);

job_s := PChar(EditJob.text);
fbput(sndbuf, fbget_fldkey('JOB'), job_s, 0);

mgr_l := StrToInt(EditMgr.text);
fbput(sndbuf, fbget_fldkey('MGR'), @mgr_l, 0);

// date_s := PChar(EditDate.text);
// fbput(sndbuf, fbget_fldkey('DATE'), @date_s, 0);
date_s := EditDate.text;
rlen := length(EditDate.text);
ret := fbchg_tu(sndbuf, fbget_fldkey('DATE'), 0, @date_s[1], 0);
if ret = -1 then begin
    LabelErr.Caption := LabelErr.Caption + 'DATE error!!!!';
    Exit;
end;

sal_f := StrToFloat(EditSal.text);
fbput(sndbuf, fbget_fldkey('SAL'), @sal_f, 0);

comm_f := StrToFloat(EditComm.text);
fbput(sndbuf, fbget_fldkey('COMM'), @comm_f, 0);

deptno_l := StrToInt(EditDept.text);
fbput(sndbuf, fbget_fldkey('DEPTNO'), @deptno_l, 0);

```

```

// トランザクション開始
ret := tx_begin();
if ret = -1 then begin
    LabelErr.Caption := LabelErr.Caption + 'tx_begin error';
    Exit;
end;

// サービスを呼び出す
ret := tpcall('FDLUPDATE', sndbuf, 0, @revbuf, @rlen, 0);
if ret < 0 then begin
    ViewErr(revbuf);
    tx_rollback();
    fbfree(sndbuf);
    fbfree(revbuf);
    tpend();
    Exit;
end;

ret := tx_commit();
if ret < 0 then begin
    ShowMessage('tx_commit failed! ' + StrPas(tpstrerror(gettperrno)));
    tx_rollback();
    fbfree(sndbuf);
    fbfree(revbuf);
    tpend();
    Exit;
end;

// 割り当てられたバッファを解除
fbfree(sndbuf);
fbfree(revbuf);
tpend();
end;

procedure TEmployeeMgrForm.BtnDelClick(Sender: TObject);
var sndbuf, revbuf: Pointer;
    empno_l, ret: longint;
    rlen: integer;

begin
    tmaxStart();

    // サーバーに送信するバッファを割り当て
    sndbuf := fballoc(1000, 10000);

```

```

if sndbuf = Nil then begin
    ShowMessage('sndbuf tmalloc failed, ' + StrPas(tpstrerror(gettperrno)));
    tpend();
    Exit;
end;

// サーバーで受信するバッファを割り当て
revbuf := fballoc(1000, 10000);
if sndbuf = Nil then begin
    ShowMessage('sndbuf tmalloc failed, ' + StrPas(tpstrerror(gettperrno)));
    tpend();
    Exit;
end;

empno_1 := StrToInt(EditEmpNo.text);
fbput(sndbuf, fbget_fldkey('EMPNO'), @empno_1, 0);

// トランザクション開始
ret := tx_begin();
if ret = -1 then begin
    LabelErr.Caption := LabelErr.Caption + 'tx_begin error';
    Exit;
end;

// サーバーにサービス呼び出す
ret := tpcall('FDLDELETE', sndbuf, 0, @revbuf, @rlen, 0);
if ret < 0 then begin
    ViewErr(revbuf);
    tx_rollback();
    fbfree(sndbuf);
    fbfree(revbuf);
    tpend();
    Exit;
end;

ret := tx_commit();
if ret < 0 then begin
    ViewErr(revbuf);
    tx_rollback();
    fbfree(sndbuf);
    fbfree(revbuf);
    tpend();
    Exit;
end;

// 割り当てられたバッファを解除
fbfree(sndbuf);

```

```

    fbfree(revbuf);

    tpend();
end;

procedure TEmployeeMgrForm.BtnInsClick(Sender: TObject);
var sndbuf, revbuf: Pointer;
    empno_l, mgr_l, deptno_l: longint;
    sal_f, comm_f: single;
    job_s, ename_s, date_s: string[100];
    ret, rlen: integer;

begin

    // tpstart実行関数
    tmaxStart();

    // 入力するバッファを割り当て
    LabelErr.Caption := '';
    sndbuf := fballoc(1000,10000);
    if sndbuf = Nil then begin
        LabelErr.Caption := 'sndbuf tmalloc failed, '
            + StrPas(tpstrerror(gettperrno));

        fbfree(sndbuf);
        tpend();
        Exit;
    end;
    revbuf := fballoc(100,1000);
    if revbuf = Nil then begin
        LabelErr.Caption := 'revbuf tmalloc failed, '
            + StrPas(tpstrerror(gettperrno));

        fbfree(revbuf);
        tpend();
        Exit;
    end;

    empno_l := StrToInt(EditEmpNo.text);
    ret := fbput(sndbuf, fbget_fldkey('EMPNO'), @empno_l, 0);
    if ret = -1 then begin
        LabelErr.Caption := 'EMPNO error!!!';
        Exit;
    end;

    ename_s := EditName.text;
    rlen := length(EditName.text);
    ret := fbchg_tu(sndbuf, fbget_fldkey('ENAME'), 0, @ename_s[1], 0);

```

```

if ret = -1 then begin
    LabelErr.Caption := LabelErr.Caption + 'ENAME error!!!';
    Exit;
end;

job_s := EditJob.text;
rlen := length(EditJob.text);
ret := fbchg_tu(sndbuf, fbget_fldkey('JOB'),0 ,@job_s[1], 0);
if ret = -1 then begin
    LabelErr.Caption := LabelErr.Caption + 'JOB error!!!';
    Exit;
end;

mgr_l := StrToInt(EditMgr.text);
ret := fbput(sndbuf, fbget_fldkey('MGR'), @mgr_l, 0);
if ret = -1 then begin
    LabelErr.Caption := LabelErr.Caption + 'MGR error!!!';
    Exit;
end;

date_s := EditDate.text;
rlen := length(EditDate.text);
ret := fbchg_tu(sndbuf, fbget_fldkey('DATE'),0 ,@date_s[1], 0);
if ret = -1 then begin
    LabelErr.Caption := LabelErr.Caption + 'DATE error!!!';
    Exit;
end;

sal_f := StrToFloat(EditSal.text);
ret := fbput(sndbuf, fbget_fldkey('SAL'), @sal_f, 0);
if ret = -1 then begin
    LabelErr.Caption := LabelErr.Caption + 'SAL error!!!';
    Exit;
end;

comm_f := StrToFloat(EditComm.text);
ret := fbput(sndbuf, fbget_fldkey('COMM'), @comm_f, 0);
if ret = -1 then begin
    LabelErr.Caption := LabelErr.Caption + 'COMM error!!!';
    Exit;
end;

deptno_l := StrToInt(EditDept.text);
ret := fbput(sndbuf, fbget_fldkey('DEPTNO'), @deptno_l, 0);
if ret = -1 then begin
    LabelErr.Caption := LabelErr.Caption + 'DEPTNO error!!!';
    Exit;
end;

```

```

end;

// トランザクション開始
ret := tx_begin();
if ret = -1 then begin
    LabelErr.Caption := LabelErr.Caption + 'tx_begin error';
    Exit;
end;

// サービスを呼び出す
ret := tpcall('FDLINSERT', sndbuf, 0, @revbuf, @rlen, 0);
if ret < 0 then begin
    ViewErr(revbuf);
    tx_rollback();
    fbfree(sndbuf);
    fbfree(revbuf);
    tpend();
    Exit;
end;

ret := tx_commit();
if ret < 0 then begin
    ViewErr(revbuf);
    tx_rollback();
    fbfree(sndbuf);
    fbfree(revbuf);
    tpend();
    Exit;
end;

// 割り当てられたバッファを解除
fbfree(revbuf);
fbfree(sndbuf);
tpend();
end;

procedure TEmployeeMgrForm.BtnReturnClick(Sender: TObject);
begin
    BtnReturn.Visible := false;
    MList.Visible := false;
end;

procedure TEmployeeMgrForm.ViewErr( a:Pointer );
var   typeS: PChar;
      codeS: PChar;
      msgS: PChar;
      tmpS: PChar;

```



```

        // Iret: Integer;
begin
    // E_TYPE      9009    long    -    -
    // E_CODE      9010    long    -    -
    // E_MSG       9011    string   -    -
    // E_TMP       9012    long     -    -

    typeS := fbgetvals(a, fbget_fldkey('E_TYPE'), 0);
    codeS := fbgetvals(a, fbget_fldkey('E_CODE'), 0);
    msgS := fbgetvals(a, fbget_fldkey('E_MSG'), 0);
    tmpS := fbgetvals(a, fbget_fldkey('E_TMP'), 0);
    ShowMessage('Error Type : ' + typeS + ' ,Code : ' + codeS + ' ,Message :
                ' + msgS + ' ,Tmp : ' + tmpS);
end;

```

照会画面

下記は、照会画面フォームのデザイン画面です。

社員番号	名前	役職	担当役員	入社日	給料	COMM	部署
452	James QA	20		20090909	1000	100	10

下記は、照会画面フォームのデザイン画面を実行する例です。

```

unit main;

interface

uses
    Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
    StdCtrls;

type
    TEmployeeMgrForm = class(TForm)

```

```

LabelEmpNo: TLabel;
LabelName: TLabel;
EditEmpNo: TEdit;
EditName: TEdit;
GroupBoxInfo: TGroupBox;
LabelJob: TLabel;
LabelMgr: TLabel;
LabelDate: TLabel;
LabelSal: TLabel;
EditJob: TEdit;
EditMgr: TEdit;
EditDate: TEdit;
EditSal: TEdit;
EditComm: TEdit;
EditDept: TEdit;
LabelComm: TLabel;
LabelDept: TLabel;
BtnSel: TButton;
BtnUdt: TButton;
BtnDel: TButton;
BtnIns: TButton;
BtnExit: TButton;
LabelErr: TLabel;
MList: TMemo;
BtnReturn: TButton;
procedure BtnExitClick(Sender: TObject);
procedure BtnSelClick(Sender: TObject);
procedure BtnUdtClick(Sender: TObject);
procedure BtnDelClick(Sender: TObject);
procedure BtnInsClick(Sender: TObject);
procedure tmaxStart();
procedure BtnReturnClick(Sender: TObject);
procedure ViewErr( a:Pointer );

{ Private declarations }
public
{ Public declarations }
end;

var
    EmployeeMgrForm: TEmployeeMgrForm;

implementation

// Atmi, Fdl 使用
uses Atmi, Fdl;
const BufferSize = 1024;

```

```

{$R *.DFM}

// Stringに長さの値を省略
{$H+}

procedure TEmployeeMgrForm.tmaxStart();

var
    tpinfo: PTPSTART;
    ret: integer;

begin
    // atmi.pasファイルに定義されています。
    // Function tmaxreadenv(a:PChar; b:PChar):Integer; cdecl; external TmaxDLL;
    // 詳細説明については、Tmax リファレンスガイドを参照してください。
    ret := tmaxreadenv('C:\tmax.env', 'aix51389');
    if ret < 0 then begin
        ShowMessage('tmaxreadenv Error');
        Exit;
    end;
    // tpstart時にユーザー情報を送信するためのバッファを割り当て
    tpinfo := tpalloc('TPSTART', NIL, 0);

    if tpinfo = Nil then begin
        ShowMessage('tpinfo tpalloc failed,' + StrPas(tpstrerror(gettperrno)));
        tpfree(tpinfo);
        Exit;
    end;
    //
    // 下記のようにオプションを指定することができます。
    //
    // ユーザー認証セキュリティーのためのユーザー・アカウント
    // tpinfo.usrname := 'tmax';
    // 自主的なメッセージ受信時に使用される識別名
    // tpinfo.cltname := 'tmax';
    // 自主的なメッセージを許可
    // tpinfo.flags := TPUNSOL_POLL;

    // Tmax接続
    ret := tpstart(tpinfo);
    if ret < 0 then begin
        ShowMessage('tpstart failed' + StrPas(tpstrerror(gettperrno)));
        tpfree(tpinfo);
        tpend();
        Exit;
    end;
end;

```

```

        // ユーザー情報を送信するためのバッファを解除
        tpfree(tpinfo);
    end;

// 終了ボタンをクリックしたとき
procedure TEmployeeMgrForm.BtnExitClick(Sender: TObject);
begin
    tpend();
    close;
end;

procedure TEmployeeMgrForm.BtnUdtClick(Sender: TObject);

var sndbuf, revbuf: Pointer;
    ret, empno_l, mgr_l, deptno_l: longint;
    sal_f, comm_f: single;
    job_s, ename_s: pointer;
    rlen: integer;
    date_s: string[100];

begin
    tmaxStart();

    // 入力するバッファを割り当て
    sndbuf := fballoc(1000, 10000);
    if sndbuf = Nil then begin
        ShowMessage('sndbuf tpalloc failed, ' + StrPas(tpstrerror(gettperrno)));
        fbfree(sndbuf);
        tpend();
        Exit;
    end;

    // 受信するバッファを割り当て
    revbuf := fballoc(1000, 10000);
    if revbuf = Nil then begin
        ShowMessage('revbuf tpalloc failed, ' + StrPas(tpstrerror(gettperrno)));
        fbfree(revbuf);
        tpend();
        Exit;
    end;

    empno_l := StrToInt(EditEmpNo.text);
    fbput(sndbuf, fbget_fldkey('EMPNO'), @empno_l, 0);

    ename_s := PChar(EditName.text);

```

```

fbput(sndbuf, fbget_fldkey('ENAME'), ename_s, 0);

job_s := PChar(EditJob.text);
fbput(sndbuf, fbget_fldkey('JOB'), job_s, 0);

mgr_l := StrToInt(EditMgr.text);
fbput(sndbuf, fbget_fldkey('MGR'), @mgr_l, 0);

// date_s := PChar(EditDate.text);
// fbput(sndbuf, fbget_fldkey('DATE'), @date_s, 0);
date_s := EditDate.text;
rlen := length(EditDate.text);
ret := fbchg_tu(sndbuf, fbget_fldkey('DATE'), 0, @date_s[1], 0);
if ret = -1 then begin
    LabelErr.Caption := LabelErr.Caption + 'DATE error!!!';
    Exit;
end;

sal_f := StrToFloat(EditSal.text);
fbput(sndbuf, fbget_fldkey('SAL'), @sal_f, 0);

comm_f := StrToFloat(EditComm.text);
fbput(sndbuf, fbget_fldkey('COMM'), @comm_f, 0);

deptno_l := StrToInt(EditDept.text);
fbput(sndbuf, fbget_fldkey('DEPTNO'), @deptno_l, 0);

// トランザクション開始
ret := tx_begin();
if ret = -1 then begin
    LabelErr.Caption := LabelErr.Caption + 'tx_begin error';
    Exit;
end;

// サービスを呼び出す
ret := tpcall('FDLUPDATE', sndbuf, 0, @revbuf, @rlen, 0);
if ret < 0 then begin
    ViewErr(revbuf);
    tx_rollback();
    fbfree(sndbuf);
    fbfree(revbuf);
    tpend();
    Exit;
end;

ret := tx_commit();

```

```

    if ret < 0 then begin
        ShowMessage('tx_commit failed! ' + StrPas(tpstrerror(gettperrno)));
        tx_rollback();
        fbfree(sndbuf);
        fbfree(revbuf);
        tpend();
        Exit;
    end;

    // 割り当てられたバッファを解除
    fbfree(sndbuf);
    fbfree(revbuf);
    tpend();
end;

procedure TEmployeeMgrForm.BtnDelClick(Sender: TObject);
var sndbuf, revbuf: Pointer;
    empno_1, ret: longint;
    rlen: integer;

begin
    tmaxStart();

    // サーバーに送信するバッファを割り当て
    sndbuf := fballoc(1000, 10000);
    if sndbuf = Nil then begin
        ShowMessage('sndbuf tmalloc failed, ' + StrPas(tpstrerror(gettperrno)));
        tpend();
        Exit;
    end;

    // サーバーで受信するバッファを割り当て
    revbuf := fballoc(1000, 10000);
    if revbuf = Nil then begin
        ShowMessage('sndbuf tmalloc failed, ' + StrPas(tpstrerror(gettperrno)));
        tpend();
        Exit;
    end;

    empno_1 := StrToInt(EditEmpNo.text);
    fbput(sndbuf, fbget_fldkey('EMPNO'), @empno_1, 0);

    // トランザクション開始
    ret := tx_begin();
    if ret = -1 then begin
        LabelErr.Caption := LabelErr.Caption + 'tx_begin error';
    end;
end;

```

```

        Exit;
    end;

    // サーバーにサービス呼び出す
    ret := tpcall('FDLDELETE', sndbuf, 0, @revbuf, @rlen, 0);
    if ret < 0 then begin
        ViewErr(revbuf);
        tx_rollback();
        fbfree(sndbuf);
        fbfree(revbuf);
        tpend();
        Exit;
    end;

    ret := tx_commit();
    if ret < 0 then begin
        ViewErr(revbuf);
        tx_rollback();
        fbfree(sndbuf);
        fbfree(revbuf);
        tpend();
        Exit;
    end;

    // 割り当てられたバッファを解除
    fbfree(sndbuf);
    fbfree(revbuf);

    tpend();
end;

procedure TEmployeeMgrForm.BtnInsClick(Sender: TObject);
var sndbuf, revbuf: Pointer;
    empno_l, mgr_l, deptno_l: longint;
    sal_f, comm_f: single;
    job_s, ename_s, date_s: string[100];
    ret, rlen: integer;

begin

    // tpstart実行関数
    tmaxStart();

    // 入力するバッファを割り当て
    LabelErr.Caption := '';
    sndbuf := fballoc(1000,10000);
    if sndbuf = Nil then begin

```

```

        LabelErr.Caption := 'sndbuf tmalloc failed, '
                        + StrPas(tpstrerror(gettperrno));

        fbfree(sndbuf);
        tpend();
        Exit;
    end;
    revbuf := fballoc(100,1000);
    if revbuf = Nil then begin
        LabelErr.Caption := 'revbuf tmalloc failed, '
                        + StrPas(tpstrerror(gettperrno));

        fbfree(revbuf);
        tpend();
        Exit;
    end;

    empno_1 := StrToInt(EditEmpNo.text);
    ret := fbput(sndbuf, fbget_fldkey('EMPNO'), @empno_1, 0);
    if ret = -1 then begin
        LabelErr.Caption := 'EMPNO error!!!';
        Exit;
    end;

    ename_s := EditName.text;
    rlen := length(EditName.text);
    ret := fbchg_tu(sndbuf, fbget_fldkey('ENAME'), 0, @ename_s[1], 0);
    if ret = -1 then begin
        LabelErr.Caption := LabelErr.Caption + 'ENAME error!!!';
        Exit;
    end;

    job_s := EditJob.text;
    rlen := length(EditJob.text);
    ret := fbchg_tu(sndbuf, fbget_fldkey('JOB'), 0, @job_s[1], 0);
    if ret = -1 then begin
        LabelErr.Caption := LabelErr.Caption + 'JOB error!!!';
        Exit;
    end;

    mgr_1 := StrToInt(EditMgr.text);
    ret := fbput(sndbuf, fbget_fldkey('MGR'), @mgr_1, 0);
    if ret = -1 then begin
        LabelErr.Caption := LabelErr.Caption + 'MGR error!!!';
        Exit;
    end;

    date_s := EditDate.text;

```



```

rlen := length(EditDate.text);
ret := fbchg_tu(sndbuf, fbget_fldkey('DATE'), 0, @date_s[1], 0);
if ret = -1 then begin
    LabelErr.Caption := LabelErr.Caption + 'DATE error!!!';
    Exit;
end;

sal_f := StrToFloat(EditSal.text);
ret := fbput(sndbuf, fbget_fldkey('SAL'), @sal_f, 0);
if ret = -1 then begin
    LabelErr.Caption := LabelErr.Caption + 'SAL error!!!';
    Exit;
end;

comm_f := StrToFloat(EditComm.text);
ret := fbput(sndbuf, fbget_fldkey('COMM'), @comm_f, 0);
if ret = -1 then begin
    LabelErr.Caption := LabelErr.Caption + 'COMM error!!!';
    Exit;
end;

deptno_1 := StrToInt(EditDept.text);
ret := fbput(sndbuf, fbget_fldkey('DEPTNO'), @deptno_1, 0);
if ret = -1 then begin
    LabelErr.Caption := LabelErr.Caption + 'DEPTNO error!!!';
    Exit;
end;

// トランザクション開始
ret := tx_begin();
if ret = -1 then begin
    LabelErr.Caption := LabelErr.Caption + 'tx_begin error';
    Exit;
end;

// サービスを呼び出す
ret := tpcall('FDLINSERT', sndbuf, 0, @revbuf, @rlen, 0);
if ret < 0 then begin
    ViewErr(revbuf);
    tx_rollback();
    fbfree(sndbuf);
    fbfree(revbuf);
    tpend();
    Exit;
end;

ret := tx_commit();

```

```

    if ret < 0 then begin
        ViewErr(revbuf);
        tx_rollback();
        fbfree(sndbuf);
        fbfree(revbuf);
        tpend();
        Exit;
    end;

    // 割り当てられたバッファを解除
    fbfree(revbuf);
    fbfree(sndbuf);
    tpend();
end;

procedure TEmployeeMgrForm.BtnReturnClick(Sender: TObject);
begin
    BtnReturn.Visible := false;
    MList.Visible := false;
end;

procedure TEmployeeMgrForm.ViewErr( a:Pointer );
var  typeS: PChar;
     codeS: PChar;
     msgS: PChar;
     tmpS: PChar;
     // Iret: Integer;
begin
    // E_TYPE          9009    long    -    -
    // E_CODE          9010    long    -    -
    // E_MSG           9011    string  -    -
    // E_TMP           9012    long    -    -

    typeS := fbgetvals(a, fbget_fldkey('E_TYPE'), 0);
    codeS := fbgetvals(a, fbget_fldkey('E_CODE'), 0);
    msgS := fbgetvals(a, fbget_fldkey('E_MSG'), 0);
    tmpS := fbgetvals(a, fbget_fldkey('E_TMP'), 0);
    ShowMessage('Error Type : ' + typeS + ' ,Code : ' + codeS + ' ,Message : '
                + msgS + ' ,Tmp : ' + tmpS);
end;

```

Oracleのエラー出力画面

下記は、Oracleのエラー出力画面です。



下記は、Oracleのエラー出力画面を実行する例です。

```
procedure TEmployeeMgrForm.ViewErr( a:Pointer );
var
    typeS: PChar;
    codeS: PChar;
    msgS: PChar;
    tmpS: PChar;
    // Iret: Integer;
begin
    // E_TYPE      9009    long    -    -
    // E_CODE      9010    long    -    -
    // E_MSG       9011    string   -    -
    // E_TMP       9012    long     -    -

    typeS := fbgetvals(a, fbget_fldkey('E_TYPE'), 0);
    codeS := fbgetvals(a, fbget_fldkey('E_CODE'), 0);
    msgS := fbgetvals(a, fbget_fldkey('E_MSG'), 0);
    tmpS := fbgetvals(a, fbget_fldkey('E_TMP'), 0);
    ShowMessage('Error Type : ' + typeS + ' ,Code : ' + codeS + ' ,Message : '
                + msgS + ' ,Tmp : ' + tmpS);
end;
```

3.2.5. サーバー・プログラム

サービス・プログラム

下記は、サービス・プログラムの例です。

<em4p_c.pc>

```
#include <stdio.h>
#include <ctype.h>
#include <tuxinc/macro.h>
#include "../fdl/demo_fdl.h"
```

```

EXEC SQL include sqlca.h;
EXEC SQL INCLUDE ORACA;
EXEC ORACLE    OPTION (ORACA=YES);
EXEC ORACLE    OPTION (RELEASE_CURSOR=YES);

#define INP     1
#define ORA     2
#define TMX     3
#define APP     4

EXEC SQL begin declare section;
int  h_empno;
char h_ename[11];
char h_job[10];
int  h_mgr;
char h_date[11];
float h_sal;
float h_comm;
int  h_deptno;
EXEC SQL end declare section;

void svc_error(long type, long err_code, char *msg, long tmp);

FDLINSERT( TPSVCINFO *msg )
{
    FBUF *rcvbuf;
    int i, occurrence;
    rcvbuf = (FBUF *)msg->data;
    h_empno = h_mgr = h_sal = h_comm = h_deptno = 0;

    memset( h_ename, 0x00, sizeof( h_ename ) );
    memset( h_job, 0x00, sizeof( h_job ) );
    memset( h_date, 0x00, sizeof( h_date ) );

    occurrence = fbkeyoccur(rcvbuf, EMPNO);

    for (i=0; i< occurrence; i++){
        fbget_tu ( rcvbuf, EMPNO, i, (char *)&h_empno, 0 );
        fbget_tu ( rcvbuf, MGR,   i, (char *)&h_mgr, 0 );
        fbget_tu ( rcvbuf, SAL,   i, (char *)&h_sal, 0 );
        fbget_tu ( rcvbuf, COMM,  i, (char *)&h_comm, 0 );
        fbget_tu ( rcvbuf, DEPTNO,i, (char *)&h_deptno, 0 );
        fbget_tu ( rcvbuf, ENAME, i, (char *)h_ename, 0 );
        fbget_tu ( rcvbuf, JOB   , i, (char *)h_job, 0 );
        fbget_tu ( rcvbuf, DATE  , i, (char *)h_date, 0 );

        EXEC SQL INSERT
        INTO emp(empno, ename, job, mgr, hiredate, sal,comm, deptno)
        VALUES (:h_empno, :h_ename, :h_job, :h_mgr,
        to_date(:h_date,'yyyymmdd'), :h_sal, :h_comm, :h_deptno );
    }
}

```

```

        if(sqlca.sqlcode != 0)
            goto LB_DBERROR;

EXEC SQL WHENEVER SQLERROR
    goto LB_DBERROR;

tpreturn(TPSUCCESS, 0L, (char *)rcvbuf, 0L, 0L);

LB_DBERROR :
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    svc_error(ORA, sqlca.sqlcode, sqlca.sqlerrm.sqlerrmc, 0);
}

FDLDELETE( TPSVCINFO *msg )
{
    FBUF *rcvbuf;
    int i, occurrence;

    rcvbuf = ( FBUF *)msg->data;
    occurrence = fbkeyoccur(rcvbuf, EMPNO);

    for (i=0; i< occurrence; i++){
        fbget_tu (rcvbuf, EMPNO, i, (char *)&h_empno , 0);

        EXEC SQL DELETE
            FROM emp
            WHERE empno = :h_empno;
    }
    if(sqlca.sqlcode != 0)
        goto LB_DBERROR;

EXEC SQL WHENEVER SQLERROR
    goto LB_DBERROR ;

EXEC SQL WHENEVER NOT FOUND
    goto LB_DBERROR ;

tpreturn(TPSUCCESS, 0L, (char *)rcvbuf, 0L, 0L);

LB_DBERROR :
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    svc_error(ORA, sqlca.sqlcode, sqlca.sqlerrm.sqlerrmc, 0) ;
}

FDLSELECT( TPSVCINFO *msg )
{
    FBUF *rcvbuf;
    FLDLEN fldlen;
    int i=0, ROWMEM=200, CHKROW=500;

```

```

rcvbuf = (FBUF *)msg->data;

if((rcvbuf=(FBUF *)tprealloc((char *)rcvbuf,ROWMEM*CHKROW))==NULL){
    rcvbuf=(FBUF *)msg->data;
    goto LB_TMAXERROR ;
}

h_empno = h_mgr = h_sal = h_comm = h_deptno = 0;

memset( h_ename, 0x00, sizeof( h_ename ) );
memset( h_job, 0x00, sizeof( h_job ) );
memset( h_date, 0x00, sizeof( h_date ) );

fbget_tu( rcvbuf, EMPNO, 0, (char *)&h_empno, &fldlen);

EXEC SQL DECLARE DB_CUR CURSOR FOR
SELECT  nvl(empno,0), nvl(ename,' '), nvl(job,' '),
        nvl(to_char(hiredate,'yyyymmdd'),' '), nvl(mgr,0),
        nvl(sal,0), nvl(comm,0), nvl(deptno,0)
FROM    emp
WHERE empno >= :h_empno-50 AND empno <= :h_empno+50;
EXEC SQL OPEN DB_CUR;

if(sqlca.sqlcode != 0)
    goto LB_DBERROR;
EXEC SQL WHENEVER SQLERROR
    goto LB_DBERROR ;
EXEC SQL WHENEVER NOT FOUND
    Do break ;

while(1) {
    EXEC SQL FETCH DB_CUR
    INTO      :h_empno,
              :h_ename,
              :h_job,
              :h_date,
              :h_mgr,
              :h_sal,
              :h_comm,
              :h_deptno;

    fbchg_tu(rcvbuf, EMPNO, i, (char *)&h_empno, 0);
    fbchg_tu(rcvbuf, MGR, i, (char *)&h_mgr, 0);
    fbchg_tu(rcvbuf, SAL, i, (char *)&h_sal, 0);
    fbchg_tu(rcvbuf, DEPTNO, i, (char *)&h_deptno, 0);
    fbchg_tu(rcvbuf, COMM, i, (char *)&h_comm, 0);
    fbchg_tu(rcvbuf, ENAME, i, (char *)&h_ename, 0);
    fbchg_tu(rcvbuf, JOB, i, (char *)&h_job, 0);
    fbchg_tu(rcvbuf, DATE, i, (char *)&h_date, 0);

```

```

        i++;
    }

    if (i < 1)
        goto LB_DBERROR;

    EXEC SQL CLOSE DB_CUR;

    tpretturn(TPSUCCESS, 0L, (char *)rcvbuf, 0L, 0L);

LB_DBERROR :
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    EXEC SQL CLOSE DB_CUR ;
    svc_error(ORA, sqlca.sqlcode, sqlca.sqlerrm.sqlerrmc, 0L);

LB_TMAXERROR :
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    EXEC SQL CLOSE DB_CUR ;
    svc_error(TMX, tperrno, "", 0L) ;
}

FDLUPDATE( TPSVCINFO *msg )
{

    FBUF *rcvbuf ;
    int i, occurrence;
    rcvbuf = (FBUF *)msg->data;
    h_empno = h_mgr = h_sal = h_comm = h_deptno = 0;

    memset( h_ename, 0x00, sizeof( h_ename ) );
    memset( h_job, 0x00, sizeof( h_job ) );
    memset( h_date, 0x00, sizeof( h_date ) );

    occurrence = fbkeyoccur(rcvbuf, EMPNO);

    for (i=0; i< occurrence; i++){
        fbget_tu ( rcvbuf, EMPNO, i, (char *)&h_empno, 0);
        fbget_tu ( rcvbuf, ENAME, i, (char *)&h_ename, 0);
        fbget_tu ( rcvbuf, JOB, i, (char *)&h_job, 0);
        fbget_tu ( rcvbuf, MGR, i, (char *)&h_mgr, 0);
        fbget_tu ( rcvbuf, SAL, i, (char *)&h_sal, 0);
        fbget_tu ( rcvbuf, COMM, i, (char *)&h_comm, 0);
        fbget_tu ( rcvbuf, DEPTNO, i, (char *)&h_deptno, 0);
        fbget_tu ( rcvbuf, DATE , i, (char *)&h_date, 0 );

        EXEC SQL UPDATE emp
        SET ename = nvl(:h_ename, ename),
            job   = nvl(:h_job, job),
            mgr   = :h_mgr,
            hiredate = nvl(to_date(:h_date, 'yyyymmdd'),hiredate),
            sal   = :h_sal,

```

```

        comm = :h_comm,
        deptno = :h_deptno
WHERE empno = :h_empno;

if(sqlca.sqlcode != 0)
    goto LB_DBERROR;

EXEC SQL WHENEVER SQLERROR
    goto LB_DBERROR;

EXEC SQL WHENEVER NOT FOUND
    goto LB_DBERROR;
}

tpreturn(TPSUCCESS, 0L, (char *)rcvbuf, 0L, 0L);

LB_DBERROR :
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    svc_error(ORA, sqlca.sqlcode, sqlca.sqlerrm.sqlerrmc, 0L);
}

/*****
 * エラー処理:サービスでエラーの発生時にそのエラーをバッファに入れてクライアントに転送します。
 *****/
void svc_error(long type, long err_code, char *msg, long tmp) {
    FBUF *transf;
    char *svcname;
    char err_msg[100];
    char temp[100];
    int i = 0;

    printf("type      ==>[%ld]\n", type);
    printf("err_code ==>[%ld]\n", err_code);
    printf("msg        ==>[%s]\n", msg);
    strcpy(err_msg, msg);

    if ((transf = (FBUF *)tpalloc("FML", NULL, 0)) == NULL) {
        printf("tpalloc failed! errno = %d\n", tperrno);
    }

    switch(type) {
        case INP:
            switch(err_code) {
                case -1000:
                    strcpy(err_msg, "当該ユーザーに権限がありません。");
                    break;
                default:
                    strcpy(err_msg,
                        "Inputエラー・メッセージが登録されていません。");
            }
            break;
        case ORA:

```



```

        if (strlen(err_msg)== 0)
            strcpy(err_msg, sqlca.sqlerrm.sqlerrmc);
        break;
    case TMX:
        if (strlen(err_msg)== 0)
            strcpy(err_msg, tpstrerror(tperrno));
        break;
    case APP:
        if (strlen(err_msg)== 0) {
            /* err_mssg=""の場合はエラー・メッセージを設定します。 *****/
            switch(err_code) {
                case -500:          /* システム関連エラー */
                    strcpy(err_msg, "ファイル作成エラーです。");
                    break;
                case -502:
                    strcpy(err_msg, "内部サービスが呼び出せませんでした。");
                    break;
                case -504:
                    strcpy(err_msg, "ソケット通信エラーです。");
                    break;
                case -505: /* 他のトランザクションで修正された場合はMSG処理*/
                    /* "照会の後、他の場所で[%s]データが変更されました。
                    \n\nを再照会した後処理してください。": クライアントで処理*/
                    strcpy(err_msg, "がありません。");
                    break;
                case -5002:
                    strcpy(err_msg, "電算室にお問い合わせください。");
                    break;
                default:
                    strcpy(err_msg,
                        "アプリケーション・エラー・メッセージが登録されていません。");
            }
        }
        break;
    }

    /* ROLLBACK          */
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    EXEC SQL ROLLBACK;

    fbchg_tu ( transf, E_TYPE, i, (char *)&type,0);
    fbchg_tu ( transf, E_CODE, i, (char *)&err_code,0);
    fbchg_tu ( transf, E_MSG, i, (char *)&err_msg,0);
    fbchg_tu ( transf, E_TMP, i, (char *)&tmp,0);

    tpreturn(TPFFAIL, 0, (char *)transf, 0L, 0L);
}

```

メイクファイル(Makefile)

下記は、emp_c.pcソースをTmaxアプリケーションに作成するメイクファイルの例です。

<emp_c.mk>

```
include $(ORACLE_HOME)/precomp/lib/env32.mk
ORALIBDIR = $(LIBHOME)
ORALIB = -L/home/oracle/OraHome/lib32/ -lclntsh -lld -lm
        `cat /home/oracle/OraHome/lib32/sysliblist` -lm -lc_r -lpthreads
TARGET = emp_c
APOBJS = emp_c.o
NSDLOBJ = $(TMAXDIR)/lib/sdl.o
#CC
CC=cc

#Oracle
LIBS = -lsvr -loras

OBS = $(APOBJS) $(SVCTOBJ)
SVCTOBJ = $(TARGET)_svctab.o

CFLAGS = -q32 -O -I$(TMAXDIR)
LDFLAGS = -brtl
APPDIR = $(TMAXDIR)/appbin
SVCTDIR = $(TMAXDIR)/svct
TMAXLIBDIR = $(TMAXDIR)/lib
#
.SUFFIXES : .c

.c.o:
        $(CC) $(CFLAGS) $(LDFLAGS) -c $<

all: $(TARGET)

$(TARGET): $(OBS)
        $(CC) $(CFLAGS) $(LDFLAGS) -L$(TMAXLIBDIR) -o $(TARGET)
        -L$(ORALIBDIR) $(ORALIB) $(OBS) $(LIBS) $(NSDLOBJ)
        mv $(TARGET) $(TMAXDIR)/appbin

$(APOBJS): $(TARGET).pc
        proc iname=emp_c include=$(TMAXDIR) define=__LINUX_ORACLE_PROC__
        $(CC) $(CFLAGS) $(LDFLAGS) -c $(TARGET).c

$(SVCTOBJ):
        touch $(SVCTDIR)/$(TARGET)_svctab.c
```

```
$(CC) $(CFLAGS) $(LDFLAGS) -c $(SVCTDIR)/$(TARGET)_svctab.c
#
clean:
    -rm -f *.o core $(TARGET) $(TARGET).lis
```


第4章 Visual Basic .netインターフェース (Unicode)

本章では、UnicodeをサポートするVisual Basic .netインターフェースで使用する関数と例について説明します。

参考

Unicodeをサポートするために使用されるIntPtrオブジェクトは、Visual Basicの場合intタイプが割り当てられるため、一般Visual Basic .netインターフェースとの互換が可能です。一般Visual Basic .netインターフェースの詳細内容については、「[第5章 Visual Basic .netインターフェース\(一般\)](#)」を参照してください。

4.1. 概要

Visual Basic .netインターフェースには、クライアント・ライブラリーの呼び出しができるように関数のプロトタイプを定義したインターフェース・モジュールが存在します。

開発者は、下記のインターフェース・モジュールをインストールすれば、Tmaxクライアント・ライブラリーで提供する関数を呼び出して使用することができます。

モジュール	説明
atmi.vb	atmi関数のプロトタイプ定義モジュールです
fdl.vb	フィールド・キー関数のプロトタイプ定義モジュールです
TmaxMacros.vb	ユーザーの便宜のためのマクロ定義クラスです
WinApi.vb	Windowsで提供する関数呼び出しのためのプロトタイプ定義モジュールです

下記は、Visual Basic .netインターフェースで提供する関数の一覧です。

関数	説明
ErrorMsg	フィールド・キー「STATLIN」の内容とtpurcodeのエラー・メッセージを出力する関数です
FdlErrorMsg	FDL関数のエラーを出力する関数です
FilltpstartBuf	Tmaxシステムと接続するために設定するバッファーであるtpstart_t構造体を構成する関数です

関数	説明
GETCAR	tpalloc()を利用してメモリーに割り当てられたバッファからString型データを読み込む関数です
GETCAR2	tpalloc()を利用してメモリーに割り当てられたバッファからBinary型データを読み込む関数です
GETCAR3	Tmax FDL関数のfbget_tu()を利用して指定したfldNameに指定されたフィールド名のnth番に該当するフィールド・データをvalue値(バイト配列)に返す関数です
GETCHR	Tmax FDL関数のfbget_tu()を利用して指定したfldNameに指定されたフィールド名のnth番に該当するフィールド・データをvalue値(CARRAY型)に返す関数です
GETDOUBLE	Tmax FDL関数のfbget_tu()を利用して指定したfldNameに指定されたフィールド名のnth番に該当するフィールド・データをvalue値(Double型)に返す関数です
GETFLOAT	Tmax FDL関数のfbget_tu()を利用して指定したfldNameに指定されたフィールド名のnth番に該当するフィールド・データをvalue値(Single型)に返す関数です
GETINT	Tmax FDL関数のfbget_tu()を利用して指定したfldNameに指定されたフィールド名のnth番に該当するフィールド・データをvalue値(Integer型)に返す関数です
GETLONG	Tmax FDL関数のfbget_tu()を利用して指定したfldNameに指定されたフィールド名のnth番に該当するフィールド・データをvalue値(Long型)に返す関数です
GETSHORT	Tmax FDL関数のfbget_tu()を利用して指定したfldNameに指定されたフィールド名のnth番に該当するフィールド・データをvalue値(Short型)に返す関数です
GETVAR	Tmax FDL関数のfbget_tu()を利用して指定したfldNameに指定されたフィールド名のnth番に該当するフィールド・データをvalue値(String型)に返す関数です
PUTCAR	tpalloc()を利用してメモリーに割り当てられたバッファにCARRAYデータを保存する関数です
PUTCAR2	tpalloc()を利用してメモリーに割り当てられたバッファにBinary型データを保存する関数です
PUTCAR3	Tmax FDL関数のfbchg_tu()を利用して指定したnthに該当する順番でfldNameに指定されたフィールド名のvalue値(バイト配列)を保存する関数です
PUTCHR	Tmax FDL関数のfbchg_tu()を利用して指定したnthに該当する順番でfldNameに指定されたフィールド名のvalue値(Char型)を保存する関数です
PUTDOUBLE	Tmax FDL関数のfbchg_tu()を利用して指定したnthに該当する順番でfldNameに指定されたフィールド名のvalue値(Double型)を保存する関数です
PUTFLOAT	Tmax FDL関数のfbchg_tu()を利用して指定したnthに該当する順番でfldNameに指定されたフィールド名のvalue値(Single型)を保存する関数です
PUTINT	Tmax FDL関数のfbchg_tu()を利用して指定したnthに該当する順番でfldNameに指定されたフィールド名のvalue値(Integer型)を保存する関数です
PUTLONG	Tmax FDL関数のfbchg_tu()を利用して指定したnthに該当する順番でfldNameに指定されたフィールド名のvalue値(Long型)を保存する関数です

関数	説明
PUTSHORT	Tmax FDL関数のfbchg_tu()を利用して指定したnthに該当する順番でfldNameに指定されたフィールド名のvalue値(Short型)を保存する関数です
PUTVAR	Tmax FDL関数のfbchg_tu()を利用して指定したnthに該当する順番でfldNameに指定されたフィールド名のvalue値(String型)を保存する関数です

参考

atmi関数とフィールド・キー関数のプロトタイプおよび機能については、『Tmax リファレンスガイド』および『Tmax アプリケーション開発ガイド』を参照してください。

4.2. 関数

4.2.1. ErrorMessage

フィールド・キー「STATLIN」の内容とtpurcodeのエラー・メッセージを出力する関数です。TmaxMacros.vb ファイルに定義されています。

マクロの内容は、フィールド・キー「STATLIN」とtpurcodeの内容をメッセージボックスで表示します。サーバーからエラー・メッセージが送信された場合は、ユーザーが出力するmsgと一緒にサーバーのメッセージをメッセージボックスで表示します。

● プロトタイプ

```
Public Shared Function ErrorMessage (ByVal pBuffer As IntPtr, ByVal msg As String)
    As Integer
```

● パラメータ

パラメータ	説明
pBuffer	バッファのポインターです
msg	出力するデータです

● 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

● 例

```

Dim lsndbuf, lrcvbuf As IntPtr
Dim rlen, lret As Integer

lret = tpcall("FDLTOUPPER", lsndbuf, 0, lrcvbuf, rlen, TPNOFLAGS)
If lret < 0 Then
    ErrorMsg(lrcvbuf, "FDLTOUPPER")
End If

```

- 関連関数

FdlErrorMsg()

4.2.2. FdlErrorMsg

FDL関数のエラーを出力する関数です。この関数を呼び出すとmsgとエラー値をメッセージボックスで表示します。

- プロトタイプ

```

Public Shared Function FdlErrorMsg(ByVal msg As String)
    As Integer

```

- パラメータ

パラメータ	説明
msg	出力するデータです

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 例

```

Dim lsndbuf As Integer
Dim lret As Integer

lret = fbput(lsndbuf, fbget_fldkey("INPUT"), txtSmallF.Text, txtSmallF.TextLength)
If lret < 0 Then
    FdlErrorMsg("fbput")
End If

```


- 関連関数

ErrorMsg()

4.2.3. FilltpstartBuf

Tmaxシステムと接続するために設定するバッファであるtpstart_t構造体を構成する関数です。

- プロトタイプ

```
Public Shared Function FilltpstartBuf(ByVal pBuffer As IntPtr,
                                     _ ByVal startInfo As tpstart_t)
    As Integer
```

- パラメータ

パラメータ	説明
pBuffer	tpalloc()を利用してメモリに割り当てられたバッファです
startInfo	tpstart_t型で宣言された変数です

- 戻り値

戻り値	説明
0 (primary host)	関数呼び出しに成功した場合です
1 (backup host)	
-1	関数呼び出しに失敗した場合です

- 例

```
Dim lsndbuf As IntPtr
Dim lret As Integer
Dim cinfo As tpstart_t

lsndbuf = tpalloc("TPSTART", "", 0)
If lsndbuf = 0 Then
    error processing
End If

cinfo.cltname = "tmax" + Chr(0)
cinfo.usrname = "tmax" + Chr(0)
cinfo.dompwd = "xamt" + Chr(0)
cinfo.usrpwd = "batman" + Chr(0)
```

```

cinfo.flags = TPUNSOL_HND

lret = FilltpstartBuf(lsndbuf, cinfo)
If ret < 0 Then
    error processing
End If

ret = tpstart(lsndbuf)
If ret < 0 Then
    error processing
End

```

4.2.4. GETCAR

tpalloc()を利用してメモリーに割り当てられたバッファーからString型データを読み込む関数です。

● プロトタイプ

```

Public Shared Function GETCAR(ByVal pBuffer As IntPtr, _ByRef value As String,
                                _ByVal len As Integer)
    As Integer

```

● パラメータ

パラメータ	説明
pBuffer	tpalloc()を利用してメモリーに割り当てられたバッファーです
value	読み込むデータです
len	読み込むデータ長です

● 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

● 例

```

Dim lsndbuf As IntPtr
Dim lret As Integer
Dim tempS As String

lsndbuf = tpalloc("STRING", "", 0)

```

```

lret = GETCAR(lrcvbuf, tempS, 10)
If lret < 0 Then
    error processing
End If

```

- 関連関数

PUTCAR()

4.2.5. GETCAR2

tpalloc()を利用してメモリーに割り当てられたバッファからBinary型データを読み込む関数です。

- プロトタイプ

```

Public Shared Function GETCAR2(ByVal pBuffer As IntPtr,
                                _ByRef value() As Byte, _ByVal len As Integer)
    As Integer

```

- パラメータ

パラメータ	説明
pBuffer	tpalloc()を利用してメモリーに割り当てられたバッファです
value	読み込むデータです
len	読み込むデータ長です

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 例

```

Dim lsndbuf As IntPtr
Dim lret As Integer
Dim tempB() As Byte

lsndbuf = tpalloc("STRING", "", 0)

lret = GETCAR2(lrcvbuf, tempB, 1024)
If lret < 0 Then

```

```

        error processing
    End If

```

- 関連関数

PUTCAR2()

4.2.6. GETCAR3

Tmax FDL関数のfbget_tu()を利用して指定したfldNameに指定されたフィールド名のnth番に該当するフィールド・データをvalue値(バイト配列)に返す関数です。

- プロトタイプ

```

Public Shared Function GETCAR3(ByVal pFBUF As IntPtr, _ByRef fldName As String,
                                _ByVal nth As Integer, _ByRef value() As Byte,
                                _ByRef len As Integer)
    As Integer

```

- パラメータ

パラメータ	説明
pFBUF	tpalloc()またはfballoc()を利用してメモリーに割り当てられたバッファです
fldName	フィールド・バッファ(pFBUF)から読み込むフィールド名です
nth	フィールド順です
value	読み込むフィールド・データです
len	読み込むフィールド・データ長です

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 例

```

Dim lsndbuf As IntPtr
Dim lret As Integer
Dim tempB() As Byte
Dim len As Integer

```

```

lsndbuf = fballot(10, 100)

lret = GETCAR3(lsndbuf, "TP_BITMAP", 0, tempB, len)
If lret < 0 Then
    error processing
End If

```

- 関連関数

GETINT()、GETLONG()、GETDOUBLE()、GETFLOAT()、GETSHORT()、GETVAR()、GETCHR()

4.2.7. GETCHR

Tmax FDL関数のfbget_tu()を利用して指定したfldNameに指定されたフィールド名のnth番に該当するフィールド・データをvalue値(CARRAY型)に返す関数です。

GETVARはString型ですが、GETCHRはCARRAY型です。CARRAY型は長さを明示的に表現する必要があるため、関数に文字列の長さを表すlen値をパラメータとして取得します。参考までに、String型は文字列の後尾にNULL値があります。

- プロトタイプ

```

Public Shared Function GETCHR(ByVal pFBUF As IntPtr, _ByRef fldName As String,

                                _ByVal nth As Integer, _ByRef value As Char,
                                _ByRef len As Integer)

    As Integer

```

- パラメータ

パラメータ	説明
pFBUF	tpalloc()またはfballoc()を利用してメモリーに割り当てられたバッファです
fldName	フィールド・バッファ(pFBUF)から読み込むフィールド名です
nth	フィールド順です
value	読み込むフィールド・データです
len	読み込むフィールド・データ長です

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 例

```
Dim lsndbuf As IntPtr
Dim lret As Integer
Dim tempC As Char
Dim len As Integer

lsndbuf = fmalloc(10, 100)

lret = GETCHR(lsndbuf, "CHR", 0, tempC, len)
If lret < 0 Then
    error processing
End If
```

- 関連関数

GETINT()、GETLONG()、GETDOUBLE()、GETFLOAT()、GETSHORT()、GETCAR3()、GETVAR()

4.2.8. GETDOUBLE

Tmax FDL関数のfbget_tu()を利用して指定したfldNameに指定されたフィールド名のnth番に該当するフィールド・データをvalue値(Double型)に返す関数です。

- プロトタイプ

```
Public Shared Function GETINT(ByVal pFBUF As IntPtr, _ByRef fldName As String,
                               _ByVal nth As Integer, _ByRef value As Double)
    As Integer
```

- パラメータ

パラメータ	説明
pFBUF	tpalloc()またfmalloc()を利用してメモリーに割り当てられたバッファです
fldName	フィールド・バッファ(pFBUF)から読み込むフィールド名です
nth	フィールド順です
value	読み込むデータです

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 例

```
Dim lsndbuf As IntPtr
Dim lret As Integer
Dim tempID As Double

lsndbuf = fballocc(10, 100)
lret = GETDOUBLE(lsndbuf, "DOUBLEDATA", 0, tempD)
If lret < 0 Then
    error processing
End If
```

- 関連関数

GETINT()、GETLONG()、GETFLOAT()、GETSHORT()、GETCAR3()、GETVAR()、GETCHR()

4.2.9. GETFLOAT

Tmax FDL関数のfbget_tu()を利用して指定したfldNameに指定されたフィールド名のnth番に該当するフィールド・データをvalue値(Single型)に返す関数です。

- プロトタイプ

```
Public Shared Function GETFLOAT(ByVal pFBUF As IntPtr, _ByRef fldName As String,
                                _ByVal nth As Integer, _ByRef value As Single)
    As Integer
```

- パラメータ

パラメータ	説明
pFBUF	tpalloc()またはfballoc()を利用してメモリーに割り当てられたバッファです
fldName	フィールド・バッファ(pFBUF)から読み込むフィールド名です
nth	フィールド順です
value	読み込むフィールド・データです

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 例

```
Dim lsndbuf As IntPtr
Dim lret As Integer
Dim tempS As Single
lsndbuf = fballoc(10, 100)
lret = GETFLOAT(lsndbuf, "TAPE_SENT", 0, tempS)
If lret < 0 Then
    error processing
End If
```

- 関連関数

GETINT()、GETLONG()、GETDOUBLE()、GETSHORT()、GETCAR3()、GETVAR()、GETCHR()

4.2.10. GETINT

Tmax FDL関数のfbget_tu()を利用して指定したfldNameに指定されたフィールド名のnth番に該当するフィールド・データをvalue値(Integer型)に返す関数です。

- プロトタイプ

```
Public Shared Function GETINT(ByVal pFBUF As IntPtr, _ByRef fldName As String,
                               _ByVal nth As Integer, _ByRef value As Integer)
    As Integer
```

- パラメータ

パラメータ	説明
pFBUF	tpalloc()またはfballoc()を利用してメモリーに割り当てられたバッファです
fldName	フィールド・バッファ(pFBUF)から読み込むフィールド名です
nth	フィールド順です
value	読み込むフィールド・データです

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合は
-1	関数呼び出しに失敗した場合は

- 例

```
Dim lsndbuf As IntPtr
Dim lret As Integer
Dim tempI As Integer

lsndbuf = fmalloc(10, 100)

lret = GETINT(lsndbuf, "INTDATA", 0, tempI)
If lret < 0 Then
    error processing
End If
```

- 関連関数

GETLONG()、GETDOUBLE()、GETFLOAT()、GETSHORT()、GETCAR3()、GETVAR()、GETCHR()

4.2.11. GETLONG

Tmax FDL関数のfbget_tu()を利用して指定したfldNameに指定されたフィールド名のnth番に該当するフィールド・データをvalue値(Long型)に返す関数です。

- プロトタイプ

```
Public Shared Function GETLONG(ByVal pFBUF As IntPtr, _ByRef fldName As String,
                                _ByVal nth As Integer, _ByRef value As Long)
    As Integer
```

- パラメータ

パラメータ	説明
pFBUF	tpalloc()またはfmalloc()を利用してメモリーに割り当てられたバッファです
fldName	フィールド・バッファ(pFBUF)から読み込むフィールド名です
nth	フィールド順です
value	読み込むフィールド・データです

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 例

```
Dim lsndbuf As IntPtr
Dim lret As Integer
Dim tempL As Long

lsndbuf = fmalloc(10, 100)

lret = GETINT(lsndbuf, "LONGDATA", 0, tempL)
If lret < 0 Then
    error processing
End If
```

- 関連関数

GETINT()、GETDOUBLE()、GETFLOAT()、GETSHORT()、GETCAR3()、GETVAR()、GETCHR()

4.2.12. GETSHORT

Tmax FDL関数のfbget_tu()を利用して指定したfldNameに指定されたフィールド名のnth番に該当するフィールド・データをvalue値(Short型)に返す関数です。

- プロトタイプ

```
Public Shared Function GETSHORT(ByVal pFBUF As IntPtr, _ByRef fldName As String,
                                _ByVal nth As Integer, _ByRef value As Short)
    As Integer
```

- パラメータ

パラメータ	説明
pFBUF	tpalloc()またはfmalloc()を利用してメモリーに割り当てられたバッファです
fldName	フィールド・バッファ(pFBUF)から読み込むフィールド名です
nth	フィールド順です
value	読み込むフィールド・データです

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 例

```
Dim lsndbuf As IntPtr
Dim lret As Integer
Dim tempS As Short

lsndbuf = fmalloc(10, 100)

lret = GETSHORT(lsndbuf, "SUPER_NUM", 0, tempS)
If lret < 0 Then
    error processing
End If
```

- 関連関数

GETINT()、GETLONG()、GETDOUBLE()、GETFLOAT()、GETCAR3()、GETVAR()、GETCHR()

4.2.13. GETVAR

Tmax FDL関数のfbget_tu()を利用して指定したfldNameに指定されたフィールド名のnth番に該当するフィールド・データをvalue値(String型)に返す関数です。

- プロトタイプ

```
Public Shared Function GETVAR(ByVal pFBUF As IntPtr, _ByRef fldName As String,
                               _ByVal nth As Integer, _ByRef value As String)
    As Integer
```

- パラメータ

パラメータ	説明
pFBUF	tpalloc()またはfmalloc()を利用してメモリーに割り当てられたバッファです
fldName	フィールド・バッファ(pFBUF)から読み込むフィールド名です
nth	フィールド順です
value	読み込むフィールド・データです

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 例

```
Dim lsndbuf As IntPtr
Dim lret As Integer
Dim tempS As String

lsndbuf = fmalloc(10, 100)

lret = GETVAR(lsndbuf, "FILENAM", 0, tempS)
If lret < 0 Then
    error processing
End If
```

- 関連関数

GETINT()、GETLONG()、GETDOUBLE()、GETFLOAT()、GETSHORT()、GETCAR3()、GETCHR()

4.2.14. PUTCAR

tpalloc()を利用してメモリーに割り当てられたバッファにCARRAY型データを保存する関数です。

- プロトタイプ

```
Public Shared Function PUTCAR(ByVal pBuffer As IntPtr, _ByVal value As String,
                               _ByVal len As Integer)
    As Integer
```

- パラメータ

パラメータ	説明
pBuffer	tpalloc()を利用してメモリーに割り当てられたバッファです
value	保存するString型データです
len	保存するString型データの長さです

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 例

```

Dim lsndbuf As IntPtr
Dim rlet As Integer
Dim tempS As String

lsndbuf = tpalloc("CARRAY", "", 0)
tempS = txtString.Text

lret = PUTCAR(lsndbuf, tempS, tempS.Length)
If lret < 0 Then
    MsgBox("PUTCAR fail..." & tpstrerror(gettperrno()) & "]")
End If

```

- 関連関数

GETCAR()

4.2.15. PUTCAR2

tpalloc()を利用してメモリーに割り当てられたバッファーにBinary型データを保存する関数です。

- プロトタイプ

```

Public Shared Function PUTCAR2(ByVal pBuffer As IntPtr, _ByRef value() As Byte,
                                _ByVal len As Integer)
    As Integer

```

- パラメータ

パラメータ	説明
pBuffer	tpalloc()を利用してメモリーに割り当てられたバッファーです
value	保存するBinary型データです
len	保存するBinary型データの長さです

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 例

```

Dim lsndbuf As IntPtr
Dim lret As Integer
Dim tempB() As Short

Dim fs As FileStream
Dim br As BinaryReader
Dim FilePath As String

lsndbuf = tpalloc("CARRAY", "", 0)

fs = New FileStream(FilePath, FileMode.Open, FileAccess.Read, FileShare.None)
br = New BinaryReader(fs)
ReDim tempB(CInt(fs.Length))
    br.Read(tempB, 0, CInt(fs.Length))
    br.Close()
    fs.Close()

lret = PUTCAR2(lsndbuf, tempB, CInt(fs.Length))
If lret < 0 Then
    MsgBox("PUTCAR2 fail...[" & tpstrerror(gettperrno()) & "]")
End If

```

- 関連関数

GETCAR2()

4.2.16. PUTCAR3

Tmax FDL関数のfbchg_tu()を利用して指定したnthに該当する順番でfldNameに指定されたフィールド名のvalue値(バイト配列)を保存する関数です。この関数は、Binary型データの処理時に使用されます。

fbchg_tu()関数を使用するため、フィールド順(nth)に既存のデータがある場合は与えられたデータ(value)に変更され、フィールド順に既存のデータがない場合は、与えられたデータが自動でフィールドに追加されます。

- プロトタイプ

```

Public Shared Function PUTCAR3(ByVal pFBUF As IntPtr, _ByVal fldName As String,

                                _ByVal nth As Integer, _ByRef value() As Byte,
                                _ByVal len As Integer)

    As Integer

```

- パラメータ

パラメータ	説明
pFBUF	tpalloc()またはfballoc()を利用してメモリーに割り当てられたバッファです
fldName	フィールド・バッファ(pFBUF)に保存するフィールド名です
nth	フィールド順です
value	保存するBinary型データです
len	保存するBinary型データの長さです

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合は
-1	関数呼び出しに失敗した場合は

- 例

```

Dim lsndbuf As IntPtr
Dim lret As Integer
Dim tempB() As Short

Dim fs As FileStream
Dim br As BinaryReader
Dim FilePath As String

lsndbuf = fballoc(10, 100)

fs = New FileStream(FilePath, FileMode.Open, FileAccess.Read, FileShare.None)
br = New BinaryReader(fs)
ReDim tempB(CInt(fs.Length))
    br.Read(tempB, 0, CInt(fs.Length))
    br.Close()
    fs.Close()

lret = PUTCAR3(lsndbuf, "TP_BITMAP", 0, tempB, CInt(fs.Length))
If lret < 0 Then
    error processing
End If

```

- 関連関数

PUTINT()、PUTLONG()、PUTSHORT()、PUTDOUBLE()、PUTFLOAT()、PUTVAR()、PUTCHR()

4.2.17. PUTCHR

Tmax FDL関数のfbchg_tu()を利用して指定したnthに該当する順番でfldNameに指定されたフィールド名のvalue(Char型)を保存する関数です。

fbchg_tu()関数を使用するため、フィールド順(nth)に既存のデータがある場合は与えられたデータ(value)に変更され、フィールド順に既存のデータがない場合は、与えられたデータが自動でフィールドに追加されます。

● プロトタイプ

```
Public Shared Function PUTCHR(ByVal pFBUF As IntPtr, _ByVal fldName As String,  
  
                                _ByVal nth As Integer, _ByRef value As Char,  
                                _ByVal len As Integer)  
  
    As Integer
```

● パラメータ

パラメータ	説明
pFBUF	tpalloc()またはfballoc()を利用してメモリーに割り当てられたバッファです
fldName	フィールド・バッファ(pFBUF)に保存するフィールド名です
nth	フィールド順です
value	保存するChar型データです
len	保存するChar型データの長さです

● 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

● 例

```
Dim lsndbuf As IntPtr  
Dim lret As Integer  
Dim tempC As Char  
  
lsndbuf = fballoc(10, 100)  
tempC = New String("a").ToCharArray()  
  
lret = PUTCHR(lsndbuf, "CHR", 0, tempC(0), 1)  
If lret < 0 Then
```



```

        error processing
    End If

```

- 関連関数

PUTINT()、PUTLONG()、PUTSHORT()、PUTDOUBLE()、PUTFLOAT()、PUTCAR3()、PUTVAR()

4.2.18. PUTDOUBLE

Tmax FDL関数のfbchg_tu()を利用して指定したnthに該当する順番でfldNameに指定されたフィールド名のvalue値(Double型)を保存する関数です。

fbchg_tu()関数を使用するため、フィールド順(nth)に既存のデータがある場合は与えられたデータ(value)に変更され、フィールド順に既存のデータがない場合は、与えられたデータが自動でフィールドに追加されます。

- プロトタイプ

```

Public Shared Function PUTDOUBLE(ByVal pFBUF As IntPtr,
                                _ByVal fldName As String, _ByVal nth As Integer,
                                _ByRef value As Double)
    As Integer

```

- パラメータ

パラメータ	説明
pFBUF	tpalloc()またはfballoc()を利用してメモリーに割り当てられたバッファです
fldName	フィールド・バッファ(pFBUF)に保存するフィールド名です
nth	フィールド順です
value	保存するDouble型データです

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 例

```

Dim lsndbuf As IntPtr
Dim lret As Integer
Dim tempD As Double

```

```

lsndbuf = fmalloc(10, 100)
tempD = CDb1(txtDouble.Text)

lret = PUTDOUBLE(lsndbuf, "DOUBLEDATA", 2, tempD)
If lret < 0 Then
    error processing
End If

```

- 関連関数

PUTINT()、PUTLONG()、PUTFLOAT()、PUTSHORT()、PUTCAR3()、PUTVAR()、PUTCHR()

4.2.19. PUTFLOAT

Tmax FDL関数のfbchg_tu()を利用して指定したnthに該当する順番でfldNameに指定されたフィールド名のvalue値(Single型)を保存する関数です。

fbchg_tu()関数を使用するため、フィールド順(nth)に既存のデータがある場合は与えられたデータ(value)に変更され、フィールド順に既存のデータがない場合は、与えられたデータが自動でフィールドに追加されます。

- プロトタイプ

```

Public Shared Function PUTFLOAT(ByVal pFBUF As IntPtr,
                                _ByVal fldName As String, _ByVal nth As Integer,
                                _ByRef value As Single)
    As Integer

```

- パラメータ

パラメータ	説明
pFBUF	tpalloc()またはfmalloc()を利用してメモリーに割り当てられたバッファです
fldName	フィールド・バッファ(pFBUF)に保存するフィールド名です
nth	フィールド順です
value	保存するString型データです

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 例

```
Dim lsndbuf As IntPtr
Dim lret As Integer
Dim tempS As Single

lsndbuf = fballocc(10, 100)
tempS = CSng(txtSingle.Text)

lret = PUTFLOAT(lsndbuf, "TAPE_SENT", 2, tempS)
If lret < 0 Then
    error processing
End If
```

- 関連関数

PUTINT(), PUTLONG(), PUTSHORT(), PUTDOUBLE(), PUTCAR3(), PUTVAR(), PUTCHR()

4.2.20. PUTINT

Tmax FDL関数のfbchg_tu()を利用して指定したnthに該当する順番でfldNameに指定されたフィールド名のvalue値(Integer型)を保存する関数です。

fbchg_tu()関数を使用するため、フィールド順(nth)に既存のデータがある場合は与えられたデータ(value)に変更され、フィールド順に既存のデータがない場合は、与えられたデータが自動でフィールドに追加されます。

- プロトタイプ

```
Public Shared Function PUTINT(ByVal pFBUF As IntPtr, _ByVal fldName As String,
                               _ByVal nth As Integer, _ByRef value As Integer)
    As Integer
```

- パラメータ

パラメータ	説明
pFBUF	tpalloc()またはfballoc()を利用してメモリーに割り当てられたバッファです
fldName	フィールド・バッファ(pFBUF)に保存するフィールド名です
nth	フィールド順です
value	保存するInteger型データです

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 例

```
Dim lsndbuf As IntPtr
Dim lret As Integer
Dim tempI As Integer

lsndbuf = fballoc(10, 100)
tempI = CInt(txtInt.Text)

lret = PUTINT(lsndbuf, "INTDATA", 2, tempI)
If lret < 0 Then
    error processing
End If
```

- 関連関数

PUTLONG()、PUTDOUBLE()、PUTFLOAT()、PUTSHORT()、PUTCAR3()、PUTVAR()、PUTCHR()

4.2.21. PUTLONG

Tmax FDL関数のfbchg_tu()を利用して指定したnthに該当する順番でfldNameに指定されたフィールド名のvalue値(Long型)を保存する関数です。

fbchg_tu()関数を使用するため、フィールド順(nth)に既存のデータがある場合は与えられたデータ(value)に変更され、フィールド順に既存のデータがない場合は、与えられたデータが自動でフィールドに追加されます。

- プロトタイプ

```
Public Shared Function PUTLONG(ByVal pFBUF As IntPtr,
                                _ByVal fldName As String, _ByVal nth As Integer,
                                _ByRef value As Long)
    As Integer
```

- パラメータ

パラメータ	説明
pFBUF	tpalloc()またはfballoc()を利用してメモリーに割り当てられたバッファです
fldName	フィールド・バッファ(pFBUF)に保存するフィールド名です

パラメータ	説明
nth	フィールド順です
value	保存するLong型データです

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 例

```
Dim lsndbuf As IntPtr
Dim lret As Integer
Dim tempL As Long

lsndbuf = fmalloc(10, 100)
tempL = CLong(txtLong.Text)

lret = PUTLONG(lsndbuf, "LONGDATA", 2, tempL)
If lret < 0 Then
    error processing
End If
```

- 関連関数

PUTINT()、PUTDOUBLE()、PUTFLOAT()、PUTSHORT()、PUTCAR3()、PUTVAR()、PUTCHR()

4.2.22. PUTSHORT

Tmax FDL関数のfbchg_tu()を利用して指定したnthに該当する順番でfldNameに指定されたフィールド名のvalue値(Short型)を保存する関数です。

fbchg_tu()関数を使用するため、フィールド順(nth)に既存のデータがある場合は与えられたデータ(value)に変更され、フィールド順に既存のデータがない場合は、与えられたデータが自動でフィールドに追加されます。

- プロトタイプ

```
Public Shared Function PUTSHORT(ByVal pFBUF As IntPtr,
                                _ByVal fldName As String, _ByVal nth As Integer,
                                _ByRef value As Short)
    As Integer
```

- パラメータ

パラメータ	説明
pFBUF	tpalloc()またはfballoc()を利用してメモリーに割り当てられたバッファです
fldName	フィールド・バッファ(pFBUF)に保存するフィールド名です
nth	フィールド順です
value	保存するShort型データです

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 例

```
Dim lsndbuf As IntPtr
Dim lret As Integer
Dim tempS As Short

lsndbuf = fballoc(10, 100)
tempS = CShort(txtShort.Text)

lret = PUTSHORT(lsndbuf, "SUPER_NUM", 2, tempS)
If lret < 0 Then
    error processing
End If
```

- 関連関数

PUTINT(), PUTLONG(), PUTFLOAT(), PUTDOUBLE(), PUTCAR3(), PUTVAR(), PUTCHR()

4.2.23. PUTVAR

Tmax FDL関数のfbchg_tu()を利用して指定したnthに該当する順番でfldNameに指定されたフィールド名のvalue値(String型)を保存する関数です。

fbchg_tu()関数を使用するため、フィールド順(nth)に既存のデータがある場合は与えられたデータ(value)に変更され、フィールド順に既存のデータがない場合は、与えられたデータが自動でフィールドに追加されます。

- プロトタイプ

```
Public Shared Function PUTVAR(ByVal pFBUF As IntPtr, _ByVal fldName As String,
                             _ByVal nth As Integer, _ByVal value As String)
    As Integer
```

- パラメータ

パラメータ	説明
pFBUF	tpalloc()またはfballoc()を利用してメモリーに割り当てられたバッファです
fldName	フィールド・バッファ(pFBUF)に保存するフィールド名です
nth	フィールド順です
value	保存するString型データです

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 例

```
Dim lsndbuf As IntPtr
Dim lret As Integer
Dim tempS As String

lsndbuf = fballoc(10, 100)
tempS = txtString.Text

lret = PUTVAR(lsndbuf, " FILENAME ", 2, tempS)
If lret < 0 Then
    error processing
End If
```

- 関連関数

PUTINT()、PUTLONG()、PUTSHORT()、PUTDOUBLE()、PUTFLOAT()、PUTCAR3()、PUTCHR()

4.3. サンプル・プログラム

account idをキー値として照会、修正、削除、入力するプログラムです。バッファはフィールド・キー・バッファを使用します。

4.3.1. プログラムの構成

atmi.vb、fdl.vb、TmaxMacros.vb、WinAPI.vbをモジュールとしてプロジェクトで追加します。

- 共通プログラム

プログラムファイル	説明
demo.f	フィールド・キー・バッファを定義したファイルです
tmconfig.m	Tmax環境設定ファイルです

- クライアント・プログラム

プログラムファイル	説明
AssemblyInfo.vb	アセンブリを行うときに使用されるデフォルト値を設定するファイルです
CodeFile1.vb	Visual Basic .netインターフェースで自動的に作成されるファイルです
EmployeeGrid.resx	Visual Basic .netインターフェースで自動的に作成されるファイルです
EmployeeGrid.vb	照会結果を表示するプログラムです
EmployeeMgr.resx	Visual Basic .netインターフェースで自動的に作成されるファイルです
EmployeeMgr.sln	Visual Basic .netインターフェースで自動的に作成されるファイルです
EmployeeMgr.suo	Visual Basic .netインターフェースで自動的に作成されるファイルです
EmployeeMgr.vb	メイン・プログラムです。照会、修正、削除、入力を行います
EmployeeMgr.vbproj	Visual Basic .netインターフェースで自動的に作成されるファイルです
EmployeeMgr.vbproj.user	Visual Basic .netインターフェースで自動的に作成されるファイルです
TmaxMacros.vb	ユーザーの便宜のためのマクロ定義クラスです
WinAPI.vb	Windowsで提供する関数を呼び出すためのプロトタイプを定義したモジュールです
atmi.vb	atmi関数のプロトタイプを定義したファイルです
fdl.vb	フィールド・キー関数のプロトタイプを定義したファイルです

- サーバー・プログラム

プログラムファイル	説明
emp_c.mk	メイクファイルです
emp_c.pc	サービスを行うサーバー・プログラムです。AIXとOracle 9iを使用します

4.3.2. プログラムの特徴

下記はプログラムの特徴です。

- クライアント・プログラム

機能	説明
Tmax接続	tpstart_t引数で接続します
バッファータ입	フィールド・キー・バッファ、構造体ファイルをfdlcユーティリティーでコンパイルして「fdl」ファイルを作成する必要があります
通信타입	tpcall()を利用した同期通信を送信するバッファと受信するバッファを別々に指定します
トランザクションの有無	TMSで自動トランザクションを付与します

- サーバー・プログラム

機能	説明
サービス	FDLSELECT、FDLINSERTを作成します
データベース指定	Oracleデータベースを使用します。システム構成ファイルのSVRGROUPにデータベース情報を指定します

4.3.3. 共通プログラム

データベースのEMP表

下記は、データベースに作成する表領域構造の例です。

EMPNO	NUMBER	NOT NULL	P1
ENAME	VARCHAR(16)		
JOB	VARCHAR(16)		
MGR	NUMBER		
HIREDATE	DATE		
SAL	NUMBER(7,2)		
COMM	NUMBER(7,2)		
DEPTNO	NUMBER		

フィールド・キー・バッファの定義

下記は、フィールド・キー・バッファを定義したファイルの例です。

```
<demo.f>
```

```
#For tmax demo employee program
EMPNO          7500          long          -          -
ENAME          7501          string         -          -
JOB            7502          string         -          -
MGR            7503          long           -          -
DATE           7504          string         -          -
SAL            7505          float          -          -
COMM           7506          float          -          -
DEPTNO         7507          long           -          -

E_TYPE         9009          long           -          -
E_CODE         9010          long           -          -
E_MSG          9011          string         -          -
E_TMP          9012          long           -          -
```

Tmax環境設定

下記は、Tmax環境設定ファイルの例です。

<tmconfig.m>

```
*DOMAIN
dom1          SHMKEY = 70000, MAXUSER = 200, MINCLH = 1, MAXCLH = 5,
              TPORTNO = 8888, BLOCKTIME = 200, TXTIME = 200

*NODE
tmax1         TMAXDIR = "/home/tmax",
              APPDIR  = "/home/tmax/appbin",
              PATHDIR = "/home/tmax/path",
              TLOGDIR = "/home/tmax/log/tlog",
              SLOGDIR = "/home/tmax/log/slog",
              ULOGDIR = "/home/tmax/log/ulog"

*SVRGROUP
svg1          NODENAME = tmax1, DBNAME = ORACLE,
              OPENINFO = "ORACLE_XA+Acc=P/scott/tiger+SesTm=60",
              TMSNAME  = svg1_tms

*SERVER
emp_c         SVGNAME = svg1, MIN = 1

*SERVICE
FDLSELECT     SVRNAME = emp_c
FDLUPDATE     SVRNAME = emp_c
FDLDELETE     SVRNAME = emp_c
FDLINSERT     SVRNAME = emp_c
```

4.3.4. クライアント・プログラム

メイン画面

下記は、メイン画面フォームのデザイン画面です。

下記は、例で使用するメイン画面フォームのデザイン構成表です。

制御名	制御タイプ	備考
EmployeeMgrForm	フォーム	Caption="社員管理プログラム"
LabelErr	ラベル	Caption="エラー"
BtnExit	コマンド・ボタン	Caption="終了"
BtnIns	コマンド・ボタン	Caption="入力"
BtnDel	コマンド・ボタン	Caption="削除"
BtnUdt	コマンド・ボタン	Caption="修正"
BtnSel	コマンド・ボタン	Caption="照会"
EditName	テキストボックス	
EditEmpNo	テキストボックス	
EditDept	テキストボックス	
EditComm	テキストボックス	
EditSal	テキストボックス	
EditDate	テキストボックス	
EditMgr	テキストボックス	
EditJob	テキストボックス	

制御名	制御タイプ	備考
MList	テキストボックス	
BtnReturn	テキストボックス	

下記は、社員管理のメイン画面デザイン・プログラムの例です。

```
Option Explicit
' 照会画面を閉じる '
Private Sub BtnReturn_Click()
    Hide
    EmployeeMgr.Show
End Sub
' 照会ボタンのクリック時に照会一覧の結果を表示する関数 '
Private Sub Form_Activate()

    ' tpstart 実行関数 '
    tmaxStart

    Dim Isendbuf As IntPtr
    Dim Irecvbuf As IntPtr
    Dim Irbuflen As Long
    Dim Iret As Integer
    Dim text As String
    Dim value As Long
    Dim dvalue As Double
    Dim svalue As Single
    Dim lenL As Long
    Dim initS, outputS As String
    Dim empnoS, enameS, jobS, mgrS, dateS, salS, commS, deptnoS As String
    Dim cntL As Long
    Dim eNo As Long
    Dim txbool As Integer ' トランザクションが開始されると 1、されなければ 0 '

    ' tpstart '
    tmaxStart

    ' トランザクションの開始有無を初期化 '
    txbool = 0

    initS = vbCrLf & vbTab & vbTab & vbTab & "***** 検索結果 *****"
            & vbCrLf & vbTab & "===== "

            & vbCrLf & vbTab & "社員番号 名前 役職 担当役員 入社日 給料 COMM 部署"
            & vbCrLf & vbTab & "===== "
```

```

        & vbCrLf & vbCrLf

If EmployeeMgr.EditEmpNo.text = "" Then
    MsgBox "社員番号を入力してください。"
    Hide
    EmployeeMgr.Show
    Exit Sub
End If

' 送信するデータのためのバッファ割り当て '
Isendbuf = fmalloc(100, 1024)
If Isendbuf = Null Then
    TmaxError ("fmalloc")
    Call fbfree(Isendbuf)
    tmaxEnd
    Exit Sub
End If

' 受信するデータのためのバッファ割り当て '
Irecvbuf = fmalloc(100, 1024)
If Irecvbuf = Null Then
    TmaxError ("fmalloc")
    Call fbfree(Irecvbuf)
    tmaxEnd
    Exit Sub
End If

' 社員管理フォームから社員番号を読み込みます。
eNo = Trim(EmployeeMgr.EditEmpNo.text)
Iret = fbput(ByVal Isendbuf, ByVal fbget_fldkey("EMPNO"), eNo, ByVal lenL)
If Iret = -1 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

' 照会サービスを要求 '
Iret = tpcall(ByVal "FDLSELECT", ByVal Isendbuf, ByVal 0, Irecvbuf, Irbuflen,

                ByVal 0)
If Iret = -1 Then
    ViewErr (Irecvbuf)
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

cntL = fbkeyoccur(ByVal Irecvbuf, ByVal fbget_fldkey("EMPNO"))

Dim i As Long

```

```

For i = 0 To cntL - 1

    Iret = GETLONG(ByVal Irecvbuf, "EMPNO", i, value)
    If Iret = -1 Then
        TmaxError ("GETLONG(EMPNO)")
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    End If
    empnoS = value

    Iret = GETVAR(ByVal Irecvbuf, "ENAME", i, text)
    If Iret = -1 Then
        TmaxError ("GETVAR(ENAME)")
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    End If
    enameS = text

    Iret = GETVAR(ByVal Irecvbuf, "JOB", i, text)
    If Iret = -1 Then
        TmaxError ("GETVAR(JOB)")
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    End If
    jobS = text

    Iret = GETLONG(ByVal Irecvbuf, "MGR", i, value)
    If Iret = -1 Then
        TmaxError ("GETLONG(MGR)")
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    End If
    mgrS = value

    Iret = GETVAR(ByVal Irecvbuf, "DATE", i, text)
    If Iret = -1 Then
        TmaxError ("GETVAR(DATE)")
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    End If
    dateS = text

    ' SAL データフィールドからfloatデータを読み込みます。 '
    ' comm.basに関数が定義されています。 '
    Iret = fbget_tu(ByVal Irecvbuf, ByVal fbget_fldkey("SAL"), i, svalue, 0)
    If Iret = -1 Then
        FdlErrorMsg ("fbget_tu")
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    End If
    salS = svalue

    Iret = GETFLOAT(ByVal Irecvbuf, "COMM", i, svalue)

```

```

        If Iret = -1 Then
            TmaxError ("GETFLOAT(COMM)")
            Call ExitSub(txbool, Isendbuf, Irecvbuf)
        End If
        commS = svalue

        Iret = GETLONG(ByVal Irecvbuf, "DEPTNO", i, value)
        If Iret = -1 Then
            TmaxError ("GETLONG(DEPTNO)")
            Call ExitSub(txbool, Isendbuf, Irecvbuf)
        End If
        deptnoS = value & vbCrLf

        outputS = outputS & "          " & empnoS & vbTab & enameS & vbTab &
            jobS & vbTab & mgrS & vbTab & dateS & vbTab &
            salS & vbTab & commS & vbTab & deptnoS

    Next i

    InfoText.text = initS & outputS
    ' 返されたデータをテキストボックスに出力 '
    LabelErr.Caption = "SAL = " & dvalue

    Call ExitSub(txbool, Isendbuf, Irecvbuf)

End Sub

Private Sub ExitSub(txbool As Integer, Isendbuf As IntPtr, Irecvbuf As IntPtr)
    If txbool = 1 Then
        tx_rollback
    End If

    ' 割り当てられたバッファを解除 '
    Call fbfree(ByVal Isendbuf&)
    Call fbfree(ByVal Irecvbuf&)

    ' tpend 実行関数 '
    tmaxEnd

    Exit Sub
End Sub

' tpstart '
Private Sub tmaxStart()
    Dim ret As Integer

```

```

' 環境設定ファイルを読み込みます。 '
' atmi.bas ファイルに定義されています。 '
' Declare Function tmaxreadenv Lib "TMAX4GL.DLL"(ByVal envfile As String,
                                                ByVal label As String) As Long'

' 詳細説明については、Tmax リファレンスガイドを参照してください。 '
ret = tmaxreadenv("C:\tmax.env", "aix51389")
If ret < 0 Then
    TmaxError ("tmaxreadenv")
    Exit Sub
End If

ret = tpstart(ByVal 0&)
If ret = -1 Then
    LabelErr.Caption = "tpstart error = " & gettperrno()
    TmaxError ("tpstart")
    Exit Sub
Else
    LabelErr.Caption = "tpstart return value = " & ret & " tpstart success"
End If

End Sub

' tpend '
Private Sub tmaxEnd()
    Dim ret As Integer
    ret = tpend()
    If ret = -1 Then
        LabelErr.Caption = "tpend error = " & gettperrno()
        TmaxError ("tpend")
        Exit Sub
    Else
        LabelErr.Caption = "tpreturn return value = " & ret
    End If
End Sub

End Sub

' Oracleのエラーを出力する関数 '
Private Sub ViewErr(ByVal a As IntPtr)

'E_TYPE      9009      long      -      -
'E_CODE      9010      long      -      -
'E_MSG       9011      string    -      -
'E_TMP       9012      long      -      -

    Dim typeL As Long
    Dim codeL As Long
    Dim msgS As String

```



```

Dim tmpL As Long
Dim Iret As Integer

Iret = fbget_tu(ByVal a, ByVal fbget_fldkey("E_TYPE"), 0, typeL, 0)

If Iret = -1 Then
    FdlErrorMsg ("fbget_tu")
    Exit Sub
End If

Iret = fbget_tu(ByVal a, ByVal fbget_fldkey("E_CODE"), 0, codeL, 0)
If Iret = -1 Then
    FdlErrorMsg ("fbget_tu")
    Exit Sub
End If

Iret = GETVAR(ByVal a, "E_MSG", 0, msgS)
Iret = fbget_tu(ByVal a, ByVal fbget_fldkey("E_TMP"), 0, tmpL, 0)
If Iret = -1 Then
    FdlErrorMsg ("fbget_tu")
    Exit Sub
End If

MsgBox ("Error Type : "&typeL&" ,Code : "&codeL&" ,Message : "&msgS&"
        ,Tmp : " & tmpL)

End Sub

```

下記は、社員管理を処理するプログラムの例です。

<EmployeeMgr.vb>

```

Option Strict Off
Option Explicit On

Imports WindowsApplication1.TmaxUtils.TmaxMacros

Public Class EmployeeMgr
    Inherits System.Windows.Forms.Form

    Public Shared EmpNo As Integer

    #Region " Windowsフォーム・デザイナーで作成したコード "
    #End Region

    Private Sub BtnExit_Click(ByVal sender As System.Object,
                              ByVal e As System.EventArgs) Handles BtnExit.Click
        End
    End

```

```

End Sub

Function getEmpNo()
    Return EmpNo
End Function

Protected Overrides Sub Finalize()
    MyBase.Finalize()
End Sub

Private Sub BtnSel_Click(ByVal sender As System.Object,
                        ByVal e As System.EventArgs) Handles BtnSel.Click
    Dim EmpGrid As EmployeeGrid
    EmpGrid = New EmployeeGrid()

    Hide()
    EmpGrid.EmpNo = EditEmpNo.Text()
    EmpGrid.ShowDialog()
    Show()
End Sub

Private Sub BtnUdt_Click(ByVal sender As System.Object,
                        ByVal e As System.EventArgs) Handles BtnUdt.Click

    Dim Isendbuf As IntPtr
    Dim Irecvbuf As IntPtr
    Dim Irbuflen As Long
    Dim Iret As Long
    Dim text As String
    Dim value As Long
    Dim dvalue As Double
    Dim svalue As Single
    Dim txbool As Integer ' トランザクションが開始されると 1、されなければ 0 '

    ' tpstart '
    tmaxStart()

    ' トランザクションの開始有無を初期化 '
    txbool = 0

    If EditEmpNo.Text = "" Then
        MsgBox("社員番号を入力してください。")
        tmaxEnd()
        Exit Sub
    End If

    ' 送信するデータのためのバッファ割り当て '

```

```

Isendbuf = fmalloc(100, 1024)

If Isendbuf = 0 Then
    Call fbfree(Isendbuf)
    tmaxEnd()
    Exit Sub
End If

' 受信するデータのためのバッファ割り当て '
Irecvbuf = fmalloc(100, 1024)

If Irecvbuf = 0 Then
    Call fbfree(Irecvbuf)
    tmaxEnd()
    Exit Sub
End If

Iret = PUTLONG(Isendbuf, "EMPNO", 0, Trim(EditEmpNo.Text))
If Iret = -1 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

Iret = PUTVAR(Isendbuf, "ENAME", 0, Trim(EditName.Text))
If Iret = -1 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

Iret = PUTVAR(Isendbuf, "JOB", 0, Trim(EditJob.Text))
If Iret = -1 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

Iret = PUTLONG(Isendbuf, "MGR", 0, Trim(EditMgr.Text))
If Iret = -1 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

Iret = PUTVAR(Isendbuf, "DATE", 0, Trim(EditDate.Text))
If Iret = -1 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

Iret = PUTFLOAT(Isendbuf, "SAL", 0, Trim(EditSal.Text))
If Iret = -1 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

```

```

Iret = PUTFLOAT(Isendbuf, "COMM", 0, Trim(EditComm.Text))
If Iret = -1 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

Iret = PUTLONG(Isendbuf, "DEPTNO", 0, Trim(EditDept.Text))
If Iret = -1 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

' トランザクション開始 '
Iret = tx_begin()
If Iret = -1 Then
    txbool = 1
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
Else
    txbool = 1
End If

' 修正サービスを要求 '
Iret = tpcall("FDLUPDATE", Isendbuf, 0, Irecvbuf, Irbuflen, 0)

LabelErr.Text = "sbuf = " & Isendbuf & ", rbuf = " & Irecvbuf

If Iret < 0 Then
    ViewErr(Irecvbuf)
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

' 割り当てられたバッファを解除

' トランザクション保存 '
Iret = tx_commit()
If Iret < 0 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
Else
    txbool = 0
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

End Sub

Private Sub BtnDel_Click(ByVal sender As System.Object,
                        ByVal e As System.EventArgs) Handles BtnDel.Click

    Dim Isendbuf As IntPtr
    Dim Irecvbuf As IntPtr

```

```

Dim Irbuflen As Long
Dim Iret As Long
Dim text As String
Dim value As Long
Dim dvalue As Double
Dim svalue As Single
Dim txbool As Integer ' トランザクションが開始されると 1、されなければ 0 '

' tpstart '
tmaxStart()

' トランザクションの開始有無を初期化 '
txbool = 0

If EditEmpNo.Text = "" Then
    MsgBox("社員番号を入力してください。")
    tmaxEnd()
    Exit Sub
End If

value = Trim(EditEmpNo.Text)

' 送信するデータのためのバッファ割り当て '
Isendbuf = fballoc(100, 1024)

If Isendbuf = 0 Then
    Call fbfree(Isendbuf)
    tmaxEnd()
    Exit Sub
End If

' 受信するデータのためのバッファ割り当て '
Irecvbuf = fballoc(100, 1024)

If Irecvbuf = 0 Then
    Call fbfree(Irecvbuf)
    tmaxEnd()
    Exit Sub
End If

Iret = PUTLONG(Isendbuf, "EMPNO", 0, value)
If Iret = -1 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

' トランザクション開始 '

```

```

Iret = tx_begin()
If Iret = -1 Then
    txbool = 1
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
Else
    txbool = 1
End If

' 照会サービスを要求 '
Iret = tpcall("FDLDELETE", Isendbuf, 0, Irecvbuf, Irbuflen, 0)
If Iret = -1 Then
    ViewErr(Irecvbuf)
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

' トランザクション保存 '
Iret = tx_commit()
If Iret < 0 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
Else
    txbool = 0
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

End Sub

Private Sub BtnIns_Click(ByVal sender As System.Object,
                        ByVal e As System.EventArgs) Handles BtnIns.Click

    Dim Isendbuf As IntPtr
    Dim Irecvbuf As IntPtr
    Dim Irbuflen As Long
    Dim Iret As Long
    Dim text As String
    Dim value As Long
    Dim dvalue As Double
    Dim svalue As Single
    Dim txbool As Integer ' トランザクションが開始されると 1、されなければ 0 '

    ' tpstart '
    tmaxStart()

    ' トランザクションの開始有無を初期化 '
    txbool = 0

```

```

If EditEmpNo.Text = "" Then
    MsgBox("社員番号を入力してください。")
    tmaxEnd()
    Exit Sub
End If

' 送信するデータのためのバッファ割り当て '
Isendbuf = fmalloc(100, 1024)

If Isendbuf = 0 Then
    Call fbfree(Isendbuf)
    tmaxEnd()
    Exit Sub
End If

' 受信するデータのためのバッファ割り当て '
Irecvbuf = fmalloc(100, 1024)

If Irecvbuf = 0 Then
    Call fbfree(Irecvbuf)
    tmaxEnd()
    Exit Sub
End If

Iret = PUTLONG(Isendbuf, "EMPNO", 0, Trim(EditEmpNo.Text))
If Iret = -1 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

Iret = PUTVAR(Isendbuf, "ENAME", 0, Trim(EditName.Text))
If Iret = -1 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

Iret = PUTVAR(Isendbuf, "JOB", 0, Trim(EditJob.Text))
If Iret = -1 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

Iret = PUTLONG(Isendbuf, "MGR", 0, Trim(EditMgr.Text))
If Iret = -1 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

Iret = PUTVAR(Isendbuf, "DATE", 0, Trim(EditDate.Text))
If Iret = -1 Then

```

```

        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    End If

    Iret = PUTFLOAT(Isendbuf, "SAL", 0, Trim(EditSal.Text))
    If Iret = -1 Then
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    End If

    Iret = PUTFLOAT(Isendbuf, "COMM", 0, Trim(EditComm.Text))
    If Iret = -1 Then
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    End If

    Iret = PUTLONG(Isendbuf, "DEPTNO", 0, Trim(EditDept.Text))
    If Iret = -1 Then
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    End If

    ' トランザクション開始 '
    Iret = tx_begin()
    If Iret = -1 Then
        txbool = 1
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    Else
        txbool = 1
    End If

    ' 修正サービスを要求 '
    Iret = tpcall("FDLINSERT", Isendbuf, 0, Irecvbuf, Irbuflen, 0)

    If Iret = -1 Then
        ViewErr(Irecvbuf)
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    End If

    ' トランザクション保存 '
    Iret = tx_commit()
    If Iret < 0 Then
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    Else
        txbool = 0
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    End If

End Sub

```



```

Private Sub ExitSub(ByVal txbool As Integer, ByVal Isendbuf As IntPtr,
                    ByVal Irecvbuf As IntPtr)

    If txbool = 1 Then
        tx_rollback()
    End If

    ' 割り当てられたバッファを解除 '
    Call fbfree(Isendbuf)
    Call fbfree(Irecvbuf)

    ' tpend '
    tmaxEnd()

Exit Sub
End Sub

Private Sub tmaxStart()
    Dim ret As Integer
    Dim envFile As String

    ' 環境設定ファイルを読み込みます。 '
    ' atmi.basファイルに定義されています。 '
    ' Declare Function tmaxreadenv Lib "TMAX4GL.DLL" (ByVal envfile As String,

                                                                    ByVal label As String)

                                                                    As Long '
    ' 詳細説明については、Tmax リファレンスガイドを参照してください。 '
    envFile = "C:\tmax.env"
    ret = tmaxreadenv(envFile, "aix51389")
    If ret < 0 Then
        MsgBox("Tmaxreadenv Error" & envFile)
        Exit Sub
    End If

    ret = tpstart(0&)
    If ret = -1 Then
        LabelErr.Text = "tpstart error = " & gettperrno()
    Else
        LabelErr.Text = "tpstart return value = " & ret & " tpstart success"
    End If

End Sub

Private Sub tmaxEnd()
    Dim ret As Integer

```

```

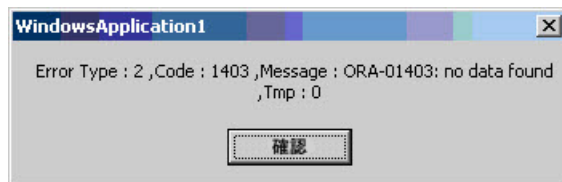
ret = tpend()
If ret = -1 Then
    LabelErr.Text = "tpend error = " & gettperrno()
Else
    LabelErr.Text = "tpreturn return value = " & ret
End If

End Sub

```

Oracleのエラー出力画面

下記は、Oracleのエラー出力画面です。



下記は、Oracleのエラー出力画面を実行するために必要な環境設定の例です。

```

' Oracleのエラーを出力する関数
Private Sub ViewErr(ByVal a As IntPtr)

    'E_TYPE      9009    long    -    -
    'E_CODE      9010    long    -    -
    'E_MSG       9011    string   -    -
    'E_TMP       9012    long     -    -

    Dim typeL As Long
    Dim codeL As Long
    Dim msgS As String
    Dim tmpL As Long
    Dim Iret As Integer

    GETLONG(a, "E_TYPE", 0, typeL)
    GETLONG(a, "E_CODE", 0, codeL)
    GETVAR(a, "E_MSG", 0, msgS)
    GETLONG(a, "E_TMP", 0, tmpL)
    MsgBox("Error Type : " & typeL & " ,Code : " & codeL & " ,Message : " & msgS & " ,Tmp : " & tmpL)

End Sub
End Class

```

照会画面フォームのデザイン画面

下記は、照会画面フォームのデザイン画面です。

下記は、照会画面フォームのデザイン構成表です。

制御名	制御タイプ	修正が必要なプロパティ
EmployeeGrid	フォーム	Caption="社員情報一覧"
ListTextBox	テキストボックス	MultiLine=True
BtnReturn	コマンド・ボタン	Caption="戻る"

下記は、社員管理照会画面のデザイン・プログラムの例です。

<EmployeeGrid.frm>

```
Option Strict Off
Option Explicit On

Imports WindowsApplication1.TmaxUtils.TmaxMacros

Imports System.IO

Public Class EmployeeGrid
    Inherits System.Windows.Forms.Form

    Public Shared EmpNo As Integer

    #Region " Windowsフォーム・デザイナーで作成したコード "
```

```

#End Region

Private Sub BtnReturn_Click(ByVal sender As System.Object,
                             ByVal e As System.EventArgs) Handles BtnReturn.Click

    Hide()
End Sub

Protected Overrides Sub OnLoad(ByVal e As System.EventArgs)

    Dim Isendbuf, Irecvbuf As IntPtr
    Dim Isendlen, Irecvlen As Integer
    Dim Irbuflen As Long
    Dim Iret As Integer
    Dim text As String
    Dim value As Long
    Dim dvalue As Double
    Dim svalue As Single
    Dim lenL As Integer
    Dim initS, outputS As String
    Dim empnoS, enameS, jobS, mgrS, dateS, salS, commS, deptnoS As String
    Dim cntL As Long
    Dim eNo As Long
    Dim i As Integer
    Dim txbool As Integer ' トランザクションが開始されると 1、されなければ 0 '

    ' tpstart '
    tmaxStart()

    ' トランザクションの開始有無を初期化 '
    txbool = 0

    ' 送信するデータのためのバッファ割り当て '
    Isendbuf = fballoc(100, 1024)
    If Isendbuf = 0 Then
        Call fbfree(Isendbuf)
        tmaxEnd()
        Exit Sub
    End If

    ' 受信するデータのためのバッファ割り当て '
    Irecvbuf = fballoc(100, 1024)
    If Irecvbuf = 0 Then
        Call fbfree(Irecvbuf)
        tmaxEnd()
        Exit Sub
    End If

```

```

Iret = fbput(Isendbuf, fbget_fldkey("EMPNO"), EmpNo, lenL)
If Iret = -1 Then
    FdlErrorMsg("fbput")
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

' 照会サービスを要求 '
Iret = tpcall("FDLSELECT", Isendbuf, 0, Irecvbuf, Irbuflen, 0)
If Iret = -1 Then
    ViewErr(Irecvbuf)
    initS = vbCrLf & vbTab & vbTab & vbTab & "***** 検索結果 *****"
        & vbCrLf & vbTab & "===== "

        & vbCrLf & vbTab & "社員番号 名前 役職 担当役員 入社日 給料 COMM 部
署"

        & vbCrLf & vbTab & "===== "

        & vbCrLf & vbCrLf
    ListTextBox.Text = initS & " "
    LabelErr.Text = "SAL = " & dvalue
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

Iret = fbkeyoccur(Irecvbuf, fbget_fldkey("EMPNO"))

For i = 0 To Iret - 1

    ' 返されたデータをテキストボックスに出力 '
    Iret = GETLONG(Irecvbuf, "EMPNO", i, value)
    If Iret = -1 Then
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    End If
    empnoS = value

    Iret = GETVAR(Irecvbuf, "ENAME", i, text)
    If Iret = -1 Then
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    End If
    enameS = text

    Iret = GETVAR(Irecvbuf, "JOB", i, text)
    If Iret = -1 Then
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    End If
    jobS = text

```

```

Iret = GETLONG(Irecvbuf, "MGR", i, value)
If Iret = -1 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If
mgrS = value

Iret = GETVAR(Irecvbuf, "DATE", i, text)
If Iret = -1 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If
dateS = text

Iret = fbget_tu(Irecvbuf, fbget_fldkey("SAL"), i, svalue, 0)
If Iret = -1 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If
salS = svalue

Iret = fbget_tu(Irecvbuf, fbget_fldkey("COMM"), i, svalue, 0)
If Iret = -1 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If
commS = svalue

Iret = GETLONG(Irecvbuf, "DEPTNO", i, value)
If Iret = -1 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If
deptnoS = value & vbCrLf

outputS = outputS & "          " & empnoS & vbTab & enameS & vbTab & jobs
          & vbTab & mgrS & vbTab & dateS & vbTab & salS & vbTab & commS
          & vbTab & deptnoS

Next i

initS = vbCrLf & vbTab & vbTab & vbTab & "***** 検索結果 *****"
        & vbCrLf & vbTab & "===== "
        & vbCrLf & vbTab & "社員番号 名前 役職 担当役員 入社日 給料 COMM 部署"
        & vbCrLf & vbTab & "===== "
        & vbCrLf & vbCrLf

ListTextBox.Text = initS & outputS
LabelErr.Text = "SAL = " & dvalue

' 割り当てられたバッファを解除 '

```

```

    Call fbfree(Isendbuf)
    Call fbfree(Irecvbuf)

    ' tpend 実行関数 '
    tmaxEnd()
End Sub

Private Sub ExitSub(ByVal txbool As Integer, ByVal Isendbuf As IntPtr,
                    ByVal Irecvbuf As IntPtr)

    If txbool = 1 Then
        tx_rollback()
    End If

    ' 割り当てられたバッファを解除 '
    Call fbfree(Isendbuf)
    Call fbfree(Irecvbuf)

    ' tpend '
    tmaxEnd()

End
End Sub

Public Sub tmaxStart()
    Dim ret As Integer
    Dim envFile As String

    ' 環境設定ファイルを呼び出します。 '
    ' atmi.basファイルに定義されています。 '
    ' Declare Function tmaxreadenv Lib "TMAX4GL.DLL" (ByVal envfile As String,
                                                    ByVal label As String) As Long
    ,

    ' 詳細説明については、『Tmax リファレンスガイド』を参照してください。 '
    envFile = "C:\tmax.env"
    ret = tmaxreadenv(envFile, "aix51389")
    If ret < 0 Then
        MsgBox("Tmaxreadenv Error : " & envFile)
    End
End If

    ret = tpstart(0&)
    If ret = -1 Then
        MsgBox("Tpstart Error")
    End
Else

```

```

        LabelErr.Text = "tpstart return value = " & ret & " tpstart success"
    End If

End Sub

Public Sub tmaxEnd()
    Dim ret As Integer
    ret = tpend()
    If ret = -1 Then
        MsgBox("Tpend Error")
    End
    Else
        LabelErr.Text = "tpreturn return value = " & ret
    End If
End Sub

Private Sub ViewErr(ByVal a As IntPtr)

    'E_TYPE      9009    long    -    -
    'E_CODE      9010    long    -    -
    'E_MSG       9011    string   -    -
    'E_TMP       9012    long     -    -

    Dim typeL As Long
    Dim codeL As Long
    Dim msgS As String
    Dim tmpL As Long
    Dim Iret As Integer

    GETLONG(a, "E_TYPE", 0, typeL)
    GETLONG(a, "E_CODE", 0, codeL)
    GETVAR(a, "E_MSG", 0, msgS)
    GETLONG(a, "E_TMP", 0, tmpL)
    MsgBox("Error Type : " & typeL & " ,Code : " & codeL & " ,Message :
        " & msgS & " ,Tmp : " & tmpL)

End Sub

End Class
Option Strict Off
Option Explicit On

Imports WindowsApplication1.TmaxUtils.TmaxMacros

Imports System.IO

Public Class EmployeeGrid
    Inherits System.Windows.Forms.Form

```



```

Public Shared EmpNo As Integer

#Region " Windowsフォームのデザイナーで作成したコード "

Public Sub New()
    MyBase.New()

    'この呼び出しはWindowsフォーム・デザイナーが必要です。
    InitializeComponent()

    'InitializeComponent()の呼び出し後、初期化作業を追加してください。

End Sub

'フォームはDisposeを再定義して構成要素の一覧を整理します。
Protected Overloads Overrides Sub Dispose(ByVal disposing As Boolean)
    If disposing Then
        If Not (components Is Nothing) Then
            components.Dispose()
        End If
    End If
    MyBase.Dispose(disposing)
End Sub

'Windowsフォーム・デザイナーが必要です。
Private components As System.ComponentModel.IContainer

'参考: 下記のプロシージャは、Windowsフォーム・デザイナーが必要です。
'Windowsフォーム・デザイナーを使用して修正できます。
'コード・エディターを使用して修正しないでください。
Friend WithEvents BtnReturn As System.Windows.Forms.Button
Friend WithEvents LabelErr As System.Windows.Forms.Label
Protected WithEvents ListTextBox As System.Windows.Forms.TextBox
<System.Diagnostics.DebuggerStepThrough> Private Sub InitializeComponent()
    Me.ListTextBox = New System.Windows.Forms.TextBox()
    Me.BtnReturn = New System.Windows.Forms.Button()
    Me.LabelErr = New System.Windows.Forms.Label()
    Me.SuspendLayout()
    '
    'ListTextBox
    '
    Me.ListTextBox.Location = New System.Drawing.Point(10, 12)
    Me.ListTextBox.Multiline = True
    Me.ListTextBox.Name = "ListTextBox"
    Me.ListTextBox.Size = New System.Drawing.Size(472, 272)
    Me.ListTextBox.TabIndex = 0
    Me.ListTextBox.TabStop = False

```

```

Me.ListTextBox.Text = "TextBox" & Microsoft.VisualBasic.ChrW(13)
                        & Microsoft.VisualBasic.ChrW(10) & "Hellow"
'
'BtnReturn
'
Me.BtnReturn.Location = New System.Drawing.Point(392, 288)
Me.BtnReturn.Name = "BtnReturn"
Me.BtnReturn.Size = New System.Drawing.Size(88, 32)
Me.BtnReturn.TabIndex = 1
Me.BtnReturn.Text = "戻る"
'
'LabelErr
'
Me.LabelErr.Location = New System.Drawing.Point(20, 294)
Me.LabelErr.Name = "LabelErr"
Me.LabelErr.Size = New System.Drawing.Size(350, 23)
Me.LabelErr.TabIndex = 2
Me.LabelErr.Text = "Label1"
'
'EmployeeGrid
'
Me.AutoScaleBaseSize = New System.Drawing.Size(6, 14)
Me.ClientSize = New System.Drawing.Size(492, 323)
Me.Controls.AddRange(New System.Windows.Forms.Control()
                        {Me.LabelErr, Me.BtnReturn, Me.ListTextBox})
Me.Name = "EmployeeGrid"
Me.Text = "社員情報一覧"
Me.ResumeLayout(False)

End Sub

#End Region

Private Sub BtnReturn_Click(ByVal sender As System.Object,
                            ByVal e As System.EventArgs) Handles BtnReturn.Click

    Hide()
End Sub

Protected Overrides Sub OnLoad(ByVal e As System.EventArgs)

    Dim Isendbuf, Irecvbuf As IntPtr
    Dim Isendlen, Irecvlen As Integer
    Dim Irbuflen As Long
    Dim Iret As Integer
    Dim text As String
    Dim value As Long

```

```

Dim dvalue As Double
Dim svalue As Single
Dim lenL As Integer
Dim initS, outputS As String
Dim empnoS, enameS, jobS, mgrS, dateS, salS, commS, deptnoS As String
Dim cntL As Long
Dim eNo As Long
Dim i As Integer
Dim txbool As Integer ' トランザクションが開始されると 1、されなければ 0 '

' tpstart '
tmaxStart()

' トランザクションの開始有無を初期化 '
txbool = 0

' 送信するデータのためのバッファ割り当て '
Isendbuf = fmalloc(100, 1024)
If Isendbuf = 0 Then
    Call fbfree(Isendbuf)
    tmaxEnd()
    Exit Sub
End If

' 受信するデータのためのバッファ割り当て '
Irecvbuf = fmalloc(100, 1024)
If Irecvbuf = 0 Then
    Call fbfree(Irecvbuf)
    tmaxEnd()
    Exit Sub
End If

Iret = fbput(Isendbuf, fbget_fldkey("EMPNO"), EmpNo, lenL)
If Iret = -1 Then
    FdlErrorMsg("fbput")
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

' 照会サービスを要求 '
Iret = tpcall("FDLSELECT", Isendbuf, 0, Irecvbuf, Irbuflen, 0)
If Iret = -1 Then
    ViewErr(Irecvbuf)
    initS = vbCrLf & vbTab & vbTab & vbTab & "***** 検索結果 *****"
        & vbCrLf & vbTab & "===== "

        & vbCrLf & vbTab & "社員番号 名前 役職 担当役員 入社日 給料 COMM 部署"

```

```

        & vbCrLf & vbTab & "===== "

        & vbCrLf & vbCrLf
ListTextBox.Text = initS & " "
LabelErr.Text = "SAL = " & dvalue
Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

Iret = fbkeyoccur(Irecvbuf, fbget_fldkey("EMPNO"))

For i = 0 To Iret - 1

    ' 返されたデータをテキストボックスに出力 '
    Iret = GETLONG(Irecvbuf, "EMPNO", i, value)
    If Iret = -1 Then
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    End If
    empnoS = value

    Iret = GETVAR(Irecvbuf, "ENAME", i, text)
    If Iret = -1 Then
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    End If
    enameS = text

    Iret = GETVAR(Irecvbuf, "JOB", i, text)
    If Iret = -1 Then
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    End If
    jobS = text

    Iret = GETLONG(Irecvbuf, "MGR", i, value)
    If Iret = -1 Then
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    End If
    mgrS = value

    Iret = GETVAR(Irecvbuf, "DATE", i, text)
    If Iret = -1 Then
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    End If
    dateS = text

    Iret = fbget_tu(Irecvbuf, fbget_fldkey("SAL"), i, svalue, 0)
    If Iret = -1 Then
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    End If

```

```

        sals = svalue

        Iret = fbget_tu(Irecvbuf, fbget_fldkey("COMM"), i, svalue, 0)
        If Iret = -1 Then
            Call ExitSub(txbool, Isendbuf, Irecvbuf)
        End If
        commS = svalue

        Iret = GETLONG(Irecvbuf, "DEPTNO", i, value)
        If Iret = -1 Then
            Call ExitSub(txbool, Isendbuf, Irecvbuf)
        End If
        deptnoS = value & vbCrLf

        outputS = outputS & "          " & empnoS & vbTab & enameS & vbTab & jobs
&
        vbTab & mgrS & vbTab & dateS & vbTab & sals & vbTab & commS &
        vbTab & deptnoS

    Next i

    initS = vbCrLf & vbTab & vbTab & vbTab & "***** 検索結果 *****"
        & vbCrLf & vbTab & "===== "
        & vbCrLf & vbTab & "社員番号 名前 役職 担当役員 入社日 給料 COMM 部署"
        & vbCrLf & vbTab & "===== "
        & vbCrLf & vbCrLf
    ListTextBox.Text = initS & outputS
    LabelErr.Text = "SAL = " & dvalue

    ' 割り当てられたバッファを解除 '
    Call fbfree(Isendbuf)
    Call fbfree(Irecvbuf)

    ' tpend 実行関数 '
    tmaxEnd()
End Sub

Private Sub ExitSub(ByVal txbool As Integer, ByVal Isendbuf As IntPtr,
    ByVal Irecvbuf As IntPtr)

    If txbool = 1 Then
        tx_rollback()
    End If

    ' 割り当てられたバッファを解除 '
    Call fbfree(Isendbuf)
    Call fbfree(Irecvbuf)

```

```

        ' tpend '
        tmaxEnd()

    End
End Sub

Public Sub tmaxStart()
    Dim ret As Integer
    Dim envFile As String

    ' 環境設定ファイルを呼び出します。 '
    ' atmi.basファイルに定義されています。 '
    ' Declare Function tmaxreadenv Lib "TMAX4GL.DLL" (ByVal envfile As String,
                                                                    ByVal label As String) As Long
    ,

    ' 詳細説明については、『Tmax リファレンスガイド』を参照してください。 '
    envFile = "C:\tmax.env"
    ret = tmaxreadenv(envFile, "aix51389")
    If ret < 0 Then
        MsgBox("Tmaxreadenv Error : " & envFile)
    End
End If

    ret = tpstart(0&)
    If ret = -1 Then
        MsgBox("Tpstart Error")
    End
    Else
        LabelErr.Text = "tpstart return value = " & ret & " tpstart success"
    End If

End Sub

Public Sub tmaxEnd()
    Dim ret As Integer
    ret = tpend()
    If ret = -1 Then
        MsgBox("Tpend Error")
    End
    Else
        LabelErr.Text = "tpreturn return value = " & ret
    End If
End Sub

Private Sub ViewErr(ByVal a As IntPtr)

```

```

'E_TYPE      9009    long    -    -
'E_CODE      9010    long    -    -
'E_MSG       9011    string   -    -
'E_TMP       9012    long     -    -

Dim typeL As Long
Dim codeL As Long
Dim msgS As String
Dim tmpL As Long
Dim Iret As Integer

GETLONG(a, "E_TYPE", 0, typeL)
GETLONG(a, "E_CODE", 0, codeL)
GETVAR(a, "E_MSG", 0, msgS)
GETLONG(a, "E_TMP", 0, tmpL)
MsgBox("Error Type : " & typeL & " ,Code : " & codeL & " ,Message :
      " & msgS & " ,Tmp : " & tmpL)

End Sub
End Class

```

4.3.5. サーバー・プログラム

サービス・プログラム

下記は、サービスを行うサーバー・プログラムの例です。

<emp_c.pc>

```

#include <stdio.h>
#include <ctype.h>
#include <tuxinc/macro.h>
#include "../fcl/demo_fcl.h"

EXEC SQL include sqlca.h;
EXEC SQL INCLUDE ORACA;
EXEC ORACLE  OPTION (ORACA=YES);
EXEC ORACLE  OPTION (RELEASE_CURSOR=YES);

#define INP    1
#define ORA    2
#define TMX    3
#define APP    4

EXEC SQL begin declare section;
int  h_empno;

```

```

char h_ename[11];
char h_job[10];
int h_mgr;
char h_date[11];
float h_sal;
float h_comm;
int h_deptno;
EXEC SQL end declare section;

void svc_error(long type, long err_code, char *msg, long tmp);

FDLINSERT( TPSVCINFO *msg )
{
    FBUF *rcvbuf;
    int i, occurrence;
    rcvbuf = (FBUF *)msg->data;
    h_empno = h_mgr = h_sal = h_comm = h_deptno = 0;

    memset( h_ename, 0x00, sizeof( h_ename ) );
    memset( h_job, 0x00, sizeof( h_job ) );
    memset( h_date, 0x00, sizeof( h_date ) );

    occurrence = fbkeyoccur(rcvbuf, EMPNO);

    for (i=0; i< occurrence; i++){
        fbget_tu ( rcvbuf, EMPNO, i, (char *)&h_empno, 0 );
        fbget_tu ( rcvbuf, MGR, i, (char *)&h_mgr, 0 );
        fbget_tu ( rcvbuf, SAL, i, (char *)&h_sal, 0 );
        fbget_tu ( rcvbuf, COMM, i, (char *)&h_comm, 0 );
        fbget_tu ( rcvbuf, DEPTNO,i, (char *)&h_deptno, 0 );
        fbget_tu ( rcvbuf, ENAME, i, (char *)h_ename, 0 );
        fbget_tu ( rcvbuf, JOB , i, (char *)h_job, 0 );
        fbget_tu ( rcvbuf, DATE , i, (char *)h_date, 0 );

        EXEC SQL INSERT
        INTO emp( empno, ename, job, mgr, hiredate, sal,comm, deptno)
        VALUES ( :h_empno, :h_ename, :h_job, :h_mgr,
                  to_date(:h_date,'yyyymmdd'), :h_sal, :h_comm, :h_deptno );
    }

    if(sqlca.sqlcode != 0)
        goto LB_DBERROR;

    EXEC SQL WHENEVER SQLERROR
        goto LB_DBERROR;

    tpreturn(TPSUCCESS, 0L, (char *)rcvbuf, 0L, 0L);

    LB_DBERROR :
        EXEC SQL WHENEVER SQLERROR CONTINUE;
        svc_error(ORA, sqlca.sqlcode, sqlca.sqlerrm.sqlerrmc, 0) ;
}

```



```

}

FDLDELETE( TPSVCINFO *msg )
{
    FBUF *rcvbuf;
    int i, occurrence;

    rcvbuf = ( FBUF *)msg->data;
    occurrence = fbkeyoccur(rcvbuf, EMPNO);

    for (i=0; i< occurrence; i++){
        fbget_tu (rcvbuf, EMPNO, i, (char *)&h_empno , 0);

        EXEC SQL DELETE
        FROM emp
        WHERE empno = :h_empno;
    }
    if(sqlca.sqlcode != 0)
        goto LB_DBERROR;

    EXEC SQL WHENEVER SQLERROR
        goto LB_DBERROR ;

    EXEC SQL WHENEVER NOT FOUND
        goto LB_DBERROR ;

    tpreturn(TPSUCCESS, 0L, (char *)rcvbuf, 0L, 0L);

    LB_DBERROR :
        EXEC SQL WHENEVER SQLERROR CONTINUE;
        svc_error(ORA, sqlca.sqlcode, sqlca.sqlerrm.sqlerrmc, 0) ;
}

FDLSELECT( TPSVCINFO *msg )
{
    FBUF *rcvbuf;
    FLDLEN fldlen;
    int i=0, ROWMEM=200, CHKROW=500;

    rcvbuf = (FBUF *)msg->data;

    if((rcvbuf=(FBUF *)tprealloc((char *)rcvbuf,ROWMEM*CHKROW))==NULL){
        rcvbuf=(FBUF *)msg->data;
        goto LB_TMAXERROR ;
    }

    h_empno = h_mgr = h_sal = h_comm = h_deptno = 0;

    memset( h_ename, 0x00, sizeof( h_ename ) );
    memset( h_job, 0x00, sizeof( h_job ) );

```

```

memset( h_date, 0x00, sizeof( h_date ) );

fbget_tu( rcvbuf, EMPNO, 0, (char *)&h_empno, &fldlen);

EXEC SQL DECLARE DB_CUR CURSOR FOR
SELECT  nvl(empno,0), nvl(ename,' '), nvl(job,' '),
        nvl(to_char(hiredate,'yyyymmdd'),' '), nvl(mgr,0),
        nvl(sal,0), nvl(comm,0), nvl(deptno,0)
FROM    emp
WHERE   empno >= :h_empno-50 AND empno <= :h_empno+50;
EXEC SQL OPEN DB_CUR;

if(sqlca.sqlcode != 0)
    goto LB_DBERROR;
EXEC SQL WHENEVER SQLERROR
    goto LB_DBERROR ;
EXEC SQL WHENEVER NOT FOUND
    Do break ;

while(1) {
    EXEC SQL FETCH DB_CUR
    INTO      :h_empno,
              :h_ename,
              :h_job,
              :h_date,
              :h_mgr,
              :h_sal,
              :h_comm,
              :h_deptno;

    fbchg_tu(rcvbuf, EMPNO,  i,(char *)&h_empno, 0);
    fbchg_tu(rcvbuf, MGR,    i,(char *)&h_mgr, 0);
    fbchg_tu(rcvbuf, SAL,    i,(char *)&h_sal, 0);
    fbchg_tu(rcvbuf, DEPTNO, i,(char *)&h_deptno, 0);
    fbchg_tu(rcvbuf, COMM,   i,(char *)&h_comm, 0);
    fbchg_tu(rcvbuf, ENAME,  i,(char *)h_ename, 0);
    fbchg_tu(rcvbuf, JOB,    i,(char *)h_job, 0);
    fbchg_tu(rcvbuf, DATE,   i,(char *)h_date, 0);

    i++;
}

if (i < 1) goto LB_DBERROR;

EXEC SQL CLOSE DB_CUR;

tpreturn(TPSUCCESS, 0L, (char *)rcvbuf, 0L, 0L);

LB_DBERROR :
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    EXEC SQL CLOSE DB_CUR ;

```

```

        svc_error(ORA, sqlca.sqlcode, sqlca.sqlerrm.sqlerrmc, 0L) ;

LB_TMAXERROR :
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    EXEC SQL CLOSE DB_CUR ;
    svc_error(TMX, tperrno, "", 0L) ;
}

FDLUPDATE( TPSVCINFO *msg )
{

    FBUF *rcvbuf ;
    int i, occurrence;
    rcvbuf = (FBUF *)msg->data;
    h_empno = h_mgr = h_sal = h_comm = h_deptno = 0;

    memset( h_ename, 0x00, sizeof( h_ename ) );
    memset( h_job, 0x00, sizeof( h_job ) );
    memset( h_date, 0x00, sizeof( h_date ) );

    occurrence = fbkeyoccur(rcvbuf, EMPNO);

    for (i=0; i< occurrence; i++){
        fbget_tu ( rcvbuf, EMPNO, i, (char *)&h_empno, 0);
        fbget_tu ( rcvbuf, ENAME, i, (char *)h_ename, 0);
        fbget_tu ( rcvbuf, JOB, i, (char *)h_job, 0);
        fbget_tu ( rcvbuf, MGR, i, (char *)&h_mgr, 0);
        fbget_tu ( rcvbuf, SAL, i, (char *)&h_sal, 0);
        fbget_tu ( rcvbuf, COMM, i, (char *)&h_comm, 0);
        fbget_tu ( rcvbuf, DEPTNO, i, (char *)&h_deptno, 0);
        fbget_tu ( rcvbuf, DATE, i, (char *)h_date, 0 );

        EXEC SQL UPDATE emp
        SET ename = nvl(:h_ename, ename),
            job = nvl(:h_job, job),
            mgr = :h_mgr,
            hiredate = nvl(to_date(:h_date, 'yyyymmdd'), hiredate),
            sal = :h_sal,
            comm = :h_comm,
            deptno = :h_deptno
        WHERE empno = :h_empno;

        if(sqlca.sqlcode != 0)
            goto LB_DBERROR;

        EXEC SQL WHENEVER SQLERROR
            goto LB_DBERROR;

        EXEC SQL WHENEVER NOT FOUND
            goto LB_DBERROR;
    }
}

```

```

        tpreturn(TPSUCCESS, 0L, (char *)rcvbuf, 0L, 0L);

    LB_DBERROR :
        EXEC SQL WHENEVER SQLERROR CONTINUE;
        svc_error(ORA, sqlca.sqlcode, sqlca.sqlerrm.sqlerrmc, 0L) ;
}

/*****
 * エラー処理:サービスでエラーの発生時にそのエラーをバッファに入れてクライアントに転送します。
 *****/
void svc_error(long type, long err_code, char *msg, long tmp) {
    FBUF *transf;
    char *svcname;
    char err_msg[100];
    char temp[100];
    int i = 0;

    printf("type      ==>[%ld]\n", type);
    printf("err_code ==>[%ld]\n", err_code);
    printf("msg        ==>[%s]\n",  msg);
    strcpy(err_msg, msg);

    if ((transf = (FBFR *)tpalloc("FML", NULL, 0)) == NULL) {
        printf("tpalloc failed! errno = %d\n", tperrno);
    }

    switch(type) {
        case INP:
            switch(err_code) {
                case -1000:
                    strcpy(err_msg, "当該ユーザーに権限がありません。");
                    break;
                default:
                    strcpy(err_msg,
                        "Inputエラー・メッセージが登録されていません。");
            }
            break;
        case ORA:
            if (strlen(err_msg)== 0) strcpy(err_msg, sqlca.sqlerrm.sqlerrmc);
            break;
        case TMX:
            if (strlen(err_msg)== 0) strcpy(err_msg, tpstrerror(tperrno));
            break;
        case APP:
            if (strlen(err_msg)== 0) {
                /* err_mssg=""の場合はエラー・メッセージを設定します。  */
                switch(err_code) {
                    case -500: /* SYSTEM関連エラー */
                        strcpy(err_msg, "ファイル作成エラーです。");
                        break;
                    case -502:

```

```

        strcpy(err_msg, "内部サービスが呼び出せませんでした。");
        break;
    case -504:
        strcpy(err_msg, "ソケット通信エラーです。");
        break;
    case -505: /* 他のトランザクションで修正された場合はMSG処理*/
        /* "照会の後、他の場所で[%s]データが変更されました。
        \n\nを再照会した後処理してください。": クライアントで処理*/
        strcpy(err_msg, "がありません。");
        break;
    case -5002:
        strcpy(err_msg, "電算室にお問い合わせください。");
        break;
    default:
        strcpy(err_msg,
            "アプリケーション・エラー・メッセージが登録されていません。");
    }
}
break;
}

/* ROLLBACK */
EXEC SQL WHENEVER SQLERROR CONTINUE;
EXEC SQL ROLLBACK;

fbchg_tu ( transf, E_TYPE, i, (char *)&type,0);
fbchg_tu ( transf, E_CODE, i, (char *)&err_code,0);
fbchg_tu ( transf, E_MSG, i, (char *)&err_msg,0);
fbchg_tu ( transf, E_TMP, i, (char *)&tmp,0);

tpreturn(TPFAIL, 0, (char *)transf, 0L, 0L);
}

```

メイクファイル(Makefile)

下記は、<emp_c.pc>ソースをTmaxアプリケーションに作成するメイクファイルの例です。

<emp_c.mk>

```

include $(ORACLE_HOME)/precomp/lib/env32.mk
ORALIBDIR = $(LIBHOME)
ORALIB = -L/home/oracle/OraHome/lib32/ -lclntsh -lld -lm
        `cat /home/oracle/OraHome/lib32/sysliblist` -lm -lc_r -lpthreads
TARGET = emp_c
APOBJS = emp_c.o
NSDLOBJ = $(TMAXDIR)/lib/sdl.o
#CC
CC=cc

```

```

#Oracle
LIBS = -lsvr -loras

OBS = $(APOBS) $(SVCTOBJ)
SVCTOBJ = $(TARGET)_svctab.o

CFLAGS = -q32 -O -I$(TMAXDIR)
LDFLAGS = -brtl
APPDIR = $(TMAXDIR)/appbin
SVCTDIR = $(TMAXDIR)/svct
TMAXLIBDIR = $(TMAXDIR)/lib
#
.SUFFIXES : .c

.c.o:
        $(CC) $(CFLAGS) $(LDFLAGS) -c $<

all: $(TARGET)

$(TARGET): $(OBS)
        $(CC) $(CFLAGS) $(LDFLAGS) -L$(TMAXLIBDIR) -o $(TARGET)
        -L$(ORALIBDIR) $(ORALIB) $(OBS) $(LIBS) $(NSDLOBJ)
        mv $(TARGET) $(TMAXDIR)/appbin

$(APOBS): $(TARGET).pc
        proc iname=emp_c include=$(TMAXDIR) define=__LINUX_ORACLE_PROC__
        $(CC) $(CFLAGS) $(LDFLAGS) -c $(TARGET).c

$(SVCTOBJ):
        touch $(SVCTDIR)/$(TARGET)_svctab.c
        $(CC) $(CFLAGS) $(LDFLAGS) -c $(SVCTDIR)/$(TARGET)_svctab.c
#
clean:
        -rm -f *.o core $(TARGET) $(TARGET).lis

```

第5章 Visual Basic .netインターフェース(一般)

本章では、一般Visual Basic .netインターフェースで使用する関数と例について説明します。

参考

UnicodeをサポートするVisual Basic .netインターフェースに関する説明は、「[第4章 Visual Basic .netインターフェース\(Unicode\)](#)」を参照してください。

5.1. 概要

Visual Basic .netインターフェースには、クライアント・ライブラリーの呼び出しができるように関数のプロトタイプを定義したインターフェース・モジュールが存在します。

開発者は、下記のインターフェース・モジュールをインストールすれば、Tmaxクライアント・ライブラリーで提供する関数を呼び出して使用することができます。

モジュール	説明
atmi.vb	atmi関数のプロトタイプ定義モジュールです
fdl.vb	フィールド・キー関数のプロトタイプ定義モジュールです
TmaxMacros.vb	ユーザーの便宜のためのマクロ定義クラスです
WinApi.vb	Windowsで提供する関数を呼び出すためのプロトタイプ定義モジュールです

下記は、Visual Basic .netインターフェースで提供する関数の一覧です。

関数	説明
ErrorMsg	フィールド・キー「STATLIN」の内容とtpurcodeのエラー・メッセージを出力する関数です
FdlErrorMsg	FDL関数のエラーを出力する関数です
FilltpstartBuf	Tmaxシステムと接続するために設定するバッファであるtpstart_t構造体を構成する関数です
GETCAR	tpalloc()を利用してメモリーに割り当てられたバッファからString型データを読み込む関数です
GETCAR2	tpalloc()を利用してメモリーに割り当てられたバッファからBinary型データを読み込む関数です

関数	説明
GETCAR3	Tmax FDL関数のfbget_tu()を利用して指定したfldNameに指定されたフィールド名のnth番に該当するフィールド・データをvalue値(バイト配列)に返す関数です
GETCHR	Tmax FDL関数のfbget_tu()を利用して指定したfldNameに指定されたフィールド名のnth番に該当するフィールド・データをvalue値(CARRAY型)に返す関数です
GETDOUBLE	Tmax FDL関数のfbget_tu()を利用して指定したfldNameに指定されたフィールド名のnth番に該当するフィールド・データをvalue値(Double型)に返す関数です
GETFLOAT	Tmax FDL関数のfbget_tu()を利用して指定したfldNameに指定されたフィールド名のnth番に該当するフィールド・データをvalue値(Single型)に返す関数です
GETINT	Tmax FDL関数のfbget_tu()を利用して指定したfldNameに指定されたフィールド名のnth番に該当するフィールド・データをvalue値(Integer型)に返す関数です
GETLONG	Tmax FDL関数のfbget_tu()を利用して指定したfldNameに指定されたフィールド名のnth番に該当するフィールド・データをvalue値(Long型)に返す関数です
GETSHORT	Tmax FDL関数のfbget_tu()を利用して指定したfldNameに指定されたフィールド名のnth番に該当するフィールド・データをvalue値(Short型)に返す関数です
GETVAR	Tmax FDL関数のfbget_tu()を利用して指定したfldNameに指定されたフィールド名のnth番に該当するフィールド・データをvalue値(String型)に返す関数です
PUTCAR	tpalloc()を利用してメモリーに割り当てられたバッファにCARRAY型データを保存する関数です
PUTCAR2	tpalloc()を利用してメモリーに割り当てられたバッファにBinary型データを保存する関数です
PUTCAR3	Tmax FDL関数のfbchg_tu()を利用して指定したnthに該当する順番でfldNameに指定されたフィールド名のvalue値(バイト配列)を保存する関数です
PUTCHR	Tmax FDL関数のfbchg_tu()を利用して指定したnthに該当する順番でfldNameに指定されたフィールド名のvalue値(Char型)を保存する関数です
PUTDOUBLE	Tmax FDL関数のfbchg_tu()を利用して指定したnthに該当する順番でfldNameに指定されたフィールド名のvalue値(Double型)を保存する関数です
PUTFLOAT	Tmax FDL関数のfbchg_tu()を利用して指定したnthに該当する順番でfldNameに指定されたフィールド名のvalue値(Single型)を保存する関数です
PUTINT	Tmax FDL関数のfbchg_tu()を利用して指定したnthに該当する順番でfldNameに指定されたフィールド名のvalue値(Integer型)を保存する関数です
PUTLONG	Tmax FDL関数のfbchg_tu()を利用して指定したnthに該当する順番でfldNameに指定されたフィールド名のvalue値(Long型)を保存する関数です
PUTSHORT	Tmax FDL関数のfbchg_tu()を利用して指定したnthに該当する順番でfldNameに指定されたフィールド名のvalue値(Short型)を保存する関数です
PUTVAR	Tmax FDL関数のfbchg_tu()を利用して指定したnthに該当する順番でfldNameに指定されたフィールド名のvalue値(String型)を保存する関数です

参考

atmi関数とフィールド・キー関数のプロトタイプおよび機能については、『Tmax リファレンスガイド』および『Tmax アプリケーション開発ガイド』を参照してください。

5.2. 関数

5.2.1. ErrorMessage

フィールド・キー「STATLIN」の内容とtpurcodeのエラー・メッセージを出力する関数です。TmaxMacros.vb ファイルに定義されています。

マクロの内容は、フィールド・キー「STATLIN」とtpurcodeの内容をMsgBoxで表示します。サーバーからエラー・メッセージが送信された場合は、ユーザーが出力しようとするmsgと一緒にサーバーのメッセージをMsgBoxで表示します。

● プロトタイプ

```
Public Shared Function ErrorMessage (ByVal pBuffer As Integer, ByVal msg As String)
    As Integer
```

● パラメータ

パラメータ	説明
pBuffer	バッファのポインターです
msg	出力しようとするデータです

● 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

● 例

```
Dim lsndbuf, lrcvbuf As Integer
Dim rlen, lret As Integer

lret = tpccall("FDLToupper", lsndbuf, 0, lrcvbuf, rlen, TPNOFLAGS)
If lret < 0 Then
```

```
ErrorMsg(lrcvbuf, "FDLTOUPPER")
End If
```

- 関連関数

FdlErrorMsg()

5.2.2. FdlErrorMsg

FDL関数のエラーを出力する関数です。この関数を呼び出すとmsgとエラー値をMsgBoxで表示します。

- プロトタイプ

```
Public Shared Function FdlErrorMsg(ByVal msg As String)
    As Integer
```

- パラメータ

パラメータ	説明
msg	出力しようとするデータです

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 例

```
Dim lsndbuf As Integer
Dim lret As Integer

lret = fbput(lsndbuf, fbget_fldkey("INPUT"), txtSmallF.Text, txtSmallF.TextLength)
If lret < 0 Then
    FdlErrorMsg("fbput")
End If
```

- 関連関数

ErrorMsg()

5.2.3. FilltpstartBuf

Tmaxシステムと接続するために設定するバッファであるtpstart_t構造体を構成する関数です。

- プロトタイプ

```
Public Shared Function FilltpstartBuf(ByVal pBuffer As Integer,  
    _ ByVal startInfo As tpstart_t)  
    As Integer
```

- パラメータ

パラメータ	説明
pBuffer	tpalloc()を利用してメモリに割り当てられたバッファです
startInfo	tpstart_t型で宣言された変数です

- 戻り値

戻り値	説明
0 (primary host)	関数呼び出しに成功した場合は 0 (primary host) 1 (backup host)
1 (backup host)	
-1	関数呼び出しに失敗した場合は

- 例

```
Dim lsndbuf As Integer  
Dim lret As Integer  
Dim cinfo As tpstart_t  
  
lsndbuf = tpalloc("TPSTART", "", 0)  
If lsndbuf = 0 Then  
    error processing  
End If  
  
cinfo.cltname = "tmax" + Chr(0)  
cinfo.usrname = "tmax" + Chr(0)  
cinfo.dompwd = "xamt" + Chr(0)  
cinfo.usrpwd = "batman" + Chr(0)  
cinfo.flags = TPUNSOL_HND  
  
lret = FilltpstartBuf(lsndbuf, cinfo)  
If ret < 0 Then  
    error processing  
End If
```

```

ret = tpstart(lsndbuf)
If ret < 0 Then
    error processing
End

```

5.2.4. GETCAR

tpalloc()を利用してメモリーに割り当てられたバッファからString型データを読み込む関数です。

- プロトタイプ

```

Public Shared Function GETCAR(ByVal pBuffer As Integer, _ByRef value As String,
                               _ByVal len As Integer)
    As Integer

```

- パラメータ

パラメータ	説明
pBuffer	tpalloc()を利用してメモリーに割り当てられたバッファです
value	読み込むデータです
len	読み込むデータ長です

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 例

```

Dim lsndbuf As Integer
Dim lret As Integer
Dim tempS As String

lsndbuf = tpalloc("STRING", "", 0)
lret = GETCAR(lrcvbuf, tempS, 10)
If lret < 0 Then
    error processing
End If

```

- 関連関数

PUTCAR()

5.2.5. GETCAR2

tpalloc()を利用してメモリーに割り当てられたバッファからBinary型データを読み込む関数です。

- プロトタイプ

```
Public Shared Function GETCAR2(ByVal pBuffer As Integer,
                               _ByRef value() As Byte, _ByVal len As Integer)
    As Integer
```

- パラメータ

パラメータ	説明
pBuffer	tpalloc()を利用してメモリーに割り当てられたバッファです
value	読み込むデータです
len	読み込むデータ長です

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合は
-1	関数呼び出しに失敗した場合は

- 例

```
Dim lsndbuf As Integer
Dim lret As Integer
Dim tempB() As Byte

lsndbuf = tpalloc("STRING", "", 0)

lret = GETCAR2(lrcvbuf, tempB, 1024)
If lret < 0 Then
    error processing
End If
```

- 関連関数

PUTCAR2()

5.2.6. GETCAR3

Tmax FDL関数のfbget_tu()を利用して指定したfldNameに指定されたフィールド名のnth番に該当するフィールド・データをvalue値(バイト配列)に返す関数です。

- プロトタイプ

```
Public Shared Function GETCAR3(ByVal pFBUF As Integer, _ByRef fldName As String,  
  
                                _ByVal nth As Integer, _ByRef value() As Byte,  
                                _ByRef len As Integer)  
  
    As Integer
```

- パラメータ

パラメータ	説明
pFBUF	tpalloc()またはfballoc()を利用してメモリーに割り当てられたバッファです
fldName	フィールド・バッファ(pFBUF)から読み込むフィールド名です
nth	フィールド順です
value	読み込むフィールド・データです
len	読み込むフィールド・データ長です

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 例

```
Dim lsndbuf As Integer  
Dim lret As Integer  
Dim tempB() As Byte  
Dim len As Integer  
  
lsndbuf = fballoc(10, 100)  
  
lret = GETCAR3(lsndbuf, "TP_BITMAP", 0, tempB, len)  
If lret < 0 Then  
    error processing  
End If
```

- 関連関数

GETINT()、GETLONG()、GETDOUBLE()、GETFLOAT()、GETSHORT()、GETVAR()、GETCHR()

5.2.7. GETCHR

Tmax FDL関数のfbget_tu()を利用して指定したfldNameに指定されたフィールド名のnth番に該当するフィールド・データをvalue値(CARRAY型)に返す関数です

GETVARはStringですが、GETCHRはCARRAY型です。CARRAY型は長さを明示的に表現する必要があるため、関数に文字列の長さを表すlen値をパラメータとして取得します。参考までに、String型は文字列の後尾にNULL値があります。

- プロトタイプ

```
Public Shared Function GETCHR(ByVal pFBUF As Integer, _ByRef fldName As String,
                              _ByVal nth As Integer, _ByRef value As Char,
                              _ByRef len As Integer)
    As Integer
```

- パラメータ

パラメータ	説明
pFBUF	tpalloc()またはfballoc()を利用してメモリーに割り当てられたバッファです
fldName	フィールド・バッファ(pFBUF)から読み込むフィールド名です
nth	フィールド順です
value	読み込むフィールド・データです
len	読み込むフィールド・データ長です

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 例

```
Dim lsndbuf As Integer
Dim lret As Integer
Dim tempC As Char
Dim len As Integer
```

```

lsndbuf = fballloc(10, 100)

lret = GETCHR(lsndbuf, "CHR", 0, tempC, len)
If lret < 0 Then
    error processing
End If

```

- 関連関数

GETINT()、GETLONG()、GETDOUBLE()、GETFLOAT()、GETSHORT()、GETCAR3()、GETVAR()

5.2.8. GETDOUBLE

Tmax FDL関数のfbget_tu()を利用して指定したfldNameに指定されたフィールド名のnth番に該当するフィールド・データをvalue値(Double型)に返す関数です。

- プロトタイプ

```

Public Shared Function GETINT(ByVal pFBUF As Integer, _ByRef fldName As String,
                               _ByVal nth As Integer, _ByRef value As Double)
    As Integer

```

- パラメータ

パラメータ	説明
pFBUF	tpalloc()またfballloc()を利用してメモリーに割り当てられたバッファです
fldName	フィールド・バッファ(pFBUF)から読み込むフィールド名です
nth	フィールド順です
value	読み込むデータです

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 例

```

Dim lsndbuf As Integer
Dim lret As Integer
Dim tempID As Double

```



```

lsndbuf = fballloc(10, 100)
lret = GETDOUBLE(lsndbuf, "DOUBLEDATA", 0, tempD)
If lret < 0 Then
    error processing
End If

```

- 関連関数

GETINT()、GETLONG()、GETFLOAT()、GETSHORT()、GETCAR3()、GETVAR()、GETCHR()

5.2.9. GETFLOAT

TmaxFDL関数のfbget_tu()を利用して指定したfldNameに指定されたフィールド名のnth番に該当するフィールド・データをvalue値(Single型)に返す関数です。

- プロトタイプ

```

Public Shared Function GETFLOAT(ByVal pFBUF As Integer, _ByRef fldName As String,
                                _ByVal nth As Integer, _ByRef value As Single)
    As Integer

```

- パラメータ

パラメータ	説明
pFBUF	tpalloc()またはfballoc()を利用してメモリーに割り当てられたバッファです
fldName	フィールド・バッファ(pFBUF)から読み込むフィールド名です
nth	フィールド順です
value	読み込むフィールド・データです

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 例

```

Dim lsndbuf As Integer
Dim lret As Integer
Dim tempS As Single
lsndbuf = fballloc(10, 100)
lret = GETFLOAT(lsndbuf, "TAPE_SENT", 0, tempS)

```

```
If lret < 0 Then
    error processing
End If
```

- 関連関数

GETINT()、GETLONG()、GETDOUBLE()、GETSHORT()、GETCAR3()、GETVAR()、GETCHR()

5.2.10. GETINT

Tmax FDL関数のfbget_tu()を利用して指定したfldNameに指定されたフィールド名のnth番に該当するフィールド・データをvalue値(Integer型)に返す関数です。

- プロトタイプ

```
Public Shared Function GETINT(ByVal pFBUF As Integer, _ByRef fldName As String,
                              _ByVal nth As Integer, _ByRef value As Integer)
    As Integer
```

- パラメータ

パラメータ	説明
pFBUF	tpalloc()またはfballoc()を利用してメモリーに割り当てられたバッファです
fldName	フィールド・バッファ(pFBUF)から読み込むフィールド名です
nth	フィールド順です
value	読み込むフィールド・データです

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 例

```
Dim lsndbuf As Integer
Dim lret As Integer
Dim tempI As Integer

lsndbuf = fballoc(10, 100)

lret = GETINT(lsndbuf, "INTDATA", 0, tempI)
```

```
If lret < 0 Then
    error processing
End If
```

- 関連関数

GETLONG()、GETDOUBLE()、GETFLOAT()、GETSHORT()、GETCAR3()、GETVAR()、GETCHR()

5.2.11. GETLONG

Tmax FDL関数のfbget_tu()を利用して指定したfldNameに指定されたフィールド名のnth番に該当するフィールド・データをvalue値(Long型)に返す関数です。

- プロトタイプ

```
Public Shared Function GETLONG(ByVal pFBUF As Integer, _ByRef fldName As String,
                                _ByVal nth As Integer, _ByRef value As Long)
    As Integer
```

- パラメータ

パラメータ	説明
pFBUF	tpalloc()またはfballoc()を利用してメモリーに割り当てられたバッファです
fldName	フィールド・バッファ(pFBUF)から読み込むフィールド名です
nth	フィールド順です
value	読み込むフィールド・データです

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 例

```
Dim lsndbuf As Integer
Dim lret As Integer
Dim tempL As Long

lsndbuf = fballoc(10, 100)

lret = GETINT(lsndbuf, "LONGDATA", 0, tempL)
```

```
If lret < 0 Then
    error processing
End If
```

- 関連関数

GETINT()、GETDOUBLE()、GETFLOAT()、GETSHORT()、GETCAR3()、GETVAR()、GETCHR()

5.2.12. GETSHORT

Tmax FDL関数のfbget_tu()を利用して指定したfldNameに指定されたフィールド名のnth番に該当するフィールド・データをvalue値(Short型)に返す関数です。

- プロトタイプ

```
Public Shared Function GETSHORT(ByVal pFBUF As Integer, _ByRef fldName As String,
                                _ByVal nth As Integer, _ByRef value As Short)
    As Integer
```

- パラメータ

パラメータ	説明
pFBUF	tpalloc()またはfballoc()を利用してメモリーに割り当てられたバッファです
fldName	フィールド・バッファ(pFBUF)から読み込むフィールド名です
nth	フィールド順です
value	読み込むフィールド・データです

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 例

```
Dim lsndbuf As Integer
Dim lret As Integer
Dim tempS As Short

lsndbuf = fmalloc(10, 100)

lret = GETSHORT(lsndbuf, "SUPER_NUM", 0, tempS)
If lret < 0 Then
    error processing
End If
```

- 関連関数

GETINT()、GETLONG()、GETDOUBLE()、GETFLOAT()、GETCAR3()、GETVAR()、GETCHR()

5.2.13. GETVAR

Tmax FDL関数のfbget_tu()を利用して指定したfldNameに指定されたフィールド名のnth番に該当するフィールド・データをvalue値(String型)に返す関数です。

- プロトタイプ

```
Public Shared Function GETVAR(ByVal pFBUF As Integer, _ByRef fldName As String,
                              _ByVal nth As Integer, _ByRef value As String)
    As Integer
```

- パラメータ

パラメータ	説明
pFBUF	tpalloc()またはfballoc()を利用してメモリーに割り当てられたバッファです
fldName	フィールド・バッファ(pFBUF)から読み込むフィールド名です
nth	フィールド順です
value	読み込むフィールド・データです

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 例

```
Dim lsndbuf As Integer
Dim lret As Integer
Dim tempS As String

lsndbuf = fmalloc(10, 100)

lret = GETVAR(lsndbuf, "FILENAM", 0, tempS)
If lret < 0 Then
    error processing
End If
```

- 関連関数

GETINT()、GETLONG()、GETDOUBLE()、GETFLOAT()、GETSHORT()、GETCAR3()、GETCHR()

5.2.14. PUTCAR

tpalloc()を利用してメモリーに割り当てられたバッファにCARRAY型データを保存する関数です。

- プロトタイプ

```
Public Shared Function PUTCAR(ByVal pBuffer As Integer, _ByVal value As String,
                              _ByVal len As Integer)
    As Integer
```

- パラメータ

パラメータ	説明
pBuffer	tpalloc()を利用してメモリーに割り当てられたバッファです
value	保存するString型データです
len	保存するString型データの長さです

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 例

```
Dim lsndbuf As Integer
Dim rlet As Integer
Dim tempS As String

lsndbuf = tpalloc("CARRAY", "", 0)
tempS = txtString.Text

lret = PUTCAR(lsndbuf, tempS, tempS.Length)
If lret < 0 Then
    MsgBox("PUTCAR fail...[" & tpstrerror(gettperrno()) & "]")
End If
```

- 関連関数

GETCAR()

5.2.15. PUTCAR2

tpalloc()を利用してメモリーに割り当てられたバッファーにBinary型データを保存する関数です。

- プロトタイプ

```
Public Shared Function PUTCAR2(ByVal pBuffer As Integer, _ByRef value() As Byte,
                                _ByVal len As Integer)
    As Integer
```

- パラメータ

パラメータ	説明
pBuffer	tpalloc()を利用してメモリーに割り当てられたバッファーです
value	保存するBinary型データです
len	保存するBinary型データの長さです

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 例

```
Dim lsndbuf As Integer
Dim lret As Integer
Dim tempB() As Short

Dim fs As FileStream
Dim br As BinaryReader
Dim FilePath As String

lsndbuf = tpalloc("CARRAY", "", 0)

fs = New FileStream(FilePath, FileMode.Open, FileAccess.Read, FileShare.None)
br = New BinaryReader(fs)
ReDim tempB(CInt(fs.Length))
    br.Read(tempB, 0, CInt(fs.Length))
    br.Close()
    fs.Close()

lret = PUTCAR2(lsndbuf, tempB, CInt(fs.Length))
If lret < 0 Then
    MsgBox("PUTCAR2 fail..." & tpstrerror(gettperrno()) & " ")
End If
```

- 関連関数

GETCAR2()

5.2.16. PUTCAR3

Tmax FDL関数のfbchg_tu()を利用して指定したnthに該当する順番でfldNameに指定されたフィールド名のvalue値(バイト配列)を保存する関数です。この関数は、Binary型データの処理時に使用されます。

fbchg_tu()関数を使用するため、フィールド順(nth)に既存のデータが存在する場合は与えられたデータ(value)に変更され、フィールド順に既存のデータが存在しない場合は、与えられたデータが自動でフィールドに追加されます。

- プロトタイプ

```
Public Shared Function PUTCAR3(ByVal pFBUF As Integer, _ByVal fldName As String,
                                _ByVal nth As Integer, _ByRef value() As Byte,
                                _ByVal len As Integer)
    As Integer
```


- パラメータ

パラメータ	説明
pFBUF	tpalloc()またはfballoc()を利用してメモリーに割り当てられたバッファです
fldName	フィールド・バッファ(pFBUF)に保存するフィールド名です
nth	フィールド順です
value	保存するBinary型データです
len	保存するBinary型データの長さです

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合は
-1	関数呼び出しに失敗した場合は

- 例

```

Dim lsndbuf As Integer
Dim lret As Integer
Dim tempB() As Short

Dim fs As FileStream
Dim br As BinaryReader
Dim FilePath As String

lsndbuf = fballoc(10, 100)

fs = New FileStream(FilePath, FileMode.Open, FileAccess.Read, FileShare.None)
br = New BinaryReader(fs)
ReDim tempB(CInt(fs.Length))
    br.Read(tempBr, 0, CInt(fs.Length))
    br.Close()
    fs.Close()

lret = PUTCAR3(lsndbuf, "TP_BITMAP", 0, tempB, CInt(fs.Length))
If lret < 0 Then
    error processing
End If

```

- 関連関数

PUTINT()、PUTLONG()、PUTSHORT()、PUTDOUBLE()、PUTFLOAT()、PUTVAR()、PUTCHR()

5.2.17. PUTCHR

Tmax FDL関数のfbchg_tu()を利用して指定したnthに該当する順番でfldNameに指定されたフィールド名のvalue値(Char型)を保存する関数です。

fbchg_tu()関数を使用するため、フィールド順(nth)に既存のデータが存在する場合は与えられたデータ(value)に変更され、フィールド順に既存のデータが存在しない場合は、与えられたデータが自動でフィールドに追加されます。

- プロトタイプ

```
Public Shared Function PUTCHR(ByVal pFBUF As Integer, _ByVal fldName As String,
                              _ByVal nth As Integer, _ByRef value As Char,
                              _ByVal len As Integer)
    As Integer
```

- パラメータ

パラメータ	説明
pFBUF	tpalloc()またはfballoc()を利用してメモリに割り当てられたバッファです
fldName	フィールド・バッファ(pFBUF)に保存するフィールド名です
nth	フィールド順です
value	保存するCharデータです
len	保存するCharデータ長です

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 例

```
Dim lsndbuf As Integer
Dim lret As Integer
Dim tempC As Char

lsndbuf = fballoc(10, 100)
tempC = New String("a").ToCharArray()

lret = PUTCHR(lsndbuf, "CHR", 0, tempC(0), 1)
If lret < 0 Then
    error processing
End If
```

- 関連関数

PUTINT()、PUTLONG()、PUTSHORT()、PUTDOUBLE()、PUTFLOAT()、PUTCAR3()、PUTVAR()

5.2.18. PUTDOUBLE

Tmax FDL関数のfbchg_tu()を利用して指定したnthに該当する順番でfldNameに指定されたフィールド名のvalue値(Double型)を保存する関数です。

fbchg_tu()関数を使用するため、フィールド順(nth)に既存のデータが存在する場合は与えられたデータ(value)に変更され、フィールド順に既存のデータが存在しない場合は、与えられたデータが自動でフィールドに追加されます。

- プロトタイプ

```
Public Shared Function PUTDOUBLE(ByVal pFBUF As Integer,
                                _ByVal fldName As String, _ByVal nth As Integer,
                                _ByRef value As Double)
    As Integer
```

- パラメータ

パラメータ	説明
pFBUF	tpalloc()またはfballoc()を利用してメモリーに割り当てられたバッファです
fldName	フィールド・バッファ(pFBUF)に保存するフィールド名です
nth	フィールド順です
value	保存するDouble型データです

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 例

```
Dim lsndbuf As Integer
Dim lret As Integer
Dim tempD As Double

lsndbuf = fballoc(10, 100)
tempD = CDb1(txtDouble.Text)
```

```

lret = PUTDOUBLE(lsndbuf, "DOUBLEDATA", 2, tempD)
If lret < 0 Then
    error processing
End If

```

- 関連関数

PUTINT()、PUTLONG()、PUTFLOAT()、PUTSHORT()、PUTCAR3()、PUTVAR()、PUTCHR()

5.2.19. PUTFLOAT

Tmax FDL関数のfbchg_tu()を利用して指定したnthに該当する順番でfldNameに指定されたフィールド名のvalue値(Single型)を保存する関数です。

fbchg_tu()関数を使用するため、フィールド順(nth)に既存のデータが存在する場合は与えられたデータ(value)に変更され、フィールド順に既存のデータが存在しない場合は、与えられたデータが自動でフィールドに追加されます。

- プロトタイプ

```

Public Shared Function PUTFLOAT(ByVal pFBUF As Integer,
                                _ByVal fldName As String, _ByVal nth As Integer,
                                _ByRef value As Single)
    As Integer

```

- パラメータ

パラメータ	説明
pFBUF	tpalloc()またはfballoc()を利用してメモリーに割り当てられたバッファです
fldName	フィールド・バッファ(pFBUF)に保存するフィールド名です
nth	フィールド順です
value	保存するString型データです

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 例

```

Dim lsndbuf As Integer
Dim lret As Integer
Dim tempS As Single

lsndbuf = fmalloc(10, 100)
tempS = CSng(txtSingle.Text)

lret = PUTFLOAT(lsndbuf, "TAPE_SENT", 2, tempS)
If lret < 0 Then
    error processing
End If

```

- 関連関数

PUTINT(), PUTLONG(), PUTSHORT(), PUTDOUBLE(), PUTCAR3(), PUTVAR(), PUTCHR()

5.2.20. PUTINT

Tmax FDL関数のfbchg_tu()を利用して指定したnthに該当する順番でfldNameに指定されたフィールド名のvalue値(Integer型)を保存する関数です。

fbchg_tu()関数を使用するため、フィールド順(nth)に既存のデータが存在する場合は与えられたデータ(value)に変更され、フィールド順に既存のデータが存在しない場合は、与えられたデータが自動でフィールドに追加されます。

- プロトタイプ

```

Public Shared Function PUTINT(ByVal pFBUF As Integer, _ByVal fldName As String,
                              _ByVal nth As Integer, _ByRef value As Integer)
    As Integer

```

- パラメータ

パラメータ	説明
pFBUF	tpalloc()またはfballoc()を利用してメモリーに割り当てられたバッファです
fldName	フィールド・バッファ(pFBUF)に保存するフィールド名です
nth	フィールド順です
value	保存するIntegerデータです

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です

戻り値	説明
-1	関数呼び出しに失敗した場合です

- 例

```
Dim lsndbuf As Integer
Dim lret As Integer
Dim tempI As Integer

lsndbuf = fmalloc(10, 100)
tempI = CInt(txtInt.Text)

lret = PUTINT(lsndbuf, "INTDATA", 2, tempI)
If lret < 0 Then
    error processing
End If
```

- 関連関数

PUTLONG(), PUTDOUBLE(), PUTFLOAT(), PUTSHORT(), PUTCAR3(), PUTVAR(), PUTCHR()

5.2.21. PUTLONG

Tmax FDL関数のfbchg_tu()を利用して指定したnthに該当する順番でfldNameに指定されたフィールド名のvalue値(Long型)を保存する関数です。

fbchg_tu()関数を使用するため、フィールド順(nth)に既存のデータが存在する場合は与えられたデータ(value)に変更され、フィールド順に既存のデータが存在しない場合は、与えられたデータが自動でフィールドに追加されます。

- プロトタイプ

```
Public Shared Function PUTLONG(ByVal pFBUF As Integer,
                                _ByVal fldName As String, _ByVal nth As Integer,
                                _ByRef value As Long)
    As Integer
```

- パラメータ

パラメータ	説明
pFBUF	tpalloc()またはfballoc()を利用してメモリーに割り当てられたバッファです
fldName	フィールド・バッファ(pFBUF)に保存するフィールド名です
nth	フィールド順です

パラメータ	説明
value	保存するLongデータです

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 例

```
Dim lsndbuf As Integer
Dim lret As Integer
Dim tempL As Long

lsndbuf = fmalloc(10, 100)
tempL = CLong(txtLong.Text)

lret = PUTLONG(lsndbuf, "LONGDATA", 2, tempL)
If lret < 0 Then
    error processing
End If
```

- 関連関数

PUTINT()、PUTDOUBLE()、PUTFLOAT()、PUTSHORT()、PUTCAR3()、PUTVAR()、PUTCHR()

5.2.22. PUTSHORT

Tmax FDL関数のfbchg_tu()を利用して指定したnthに該当する順番でfldNameに指定されたフィールド名のvalue値(Short型)を保存する関数です。

fbchg_tu()関数を使用するため、フィールド順(nth)に既存のデータが存在する場合は与えられたデータ(value)に変更され、フィールド順に既存のデータが存在しない場合は、与えられたデータが自動でフィールドに追加されます。

- プロトタイプ

```
Public Shared Function PUTSHORT(ByVal pFBUF As Integer,
                                _ByVal fldName As String, _ByVal nth As Integer,
                                _ByRef value As Short)
    As Integer
```

- パラメータ

パラメータ	説明
pFBUF	tpalloc()またはfballoc()を利用してメモリーに割り当てられたバッファです
fldName	フィールド・バッファ(pFBUF)に保存するフィールド名です
nth	フィールド順です
value	保存するShortデータです

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 例

```
Dim lsndbuf As Integer
Dim lret As Integer
Dim tempS As Short

lsndbuf = fballoc(10, 100)
tempS = CShort(txtShort.Text)

lret = PUTSHORT(lsndbuf, "SUPER_NUM", 2, tempS)
If lret < 0 Then
    error processing
End If
```

- 関連関数

PUTINT()、PUTLONG()、PUTFLOAT()、PUTDOUBLE()、PUTCAR3()、PUTVAR()、PUTCHR()

5.2.23. PUTVAR

Tmax FDL関数のfbchg_tu()を利用して指定したnthに該当する順番でfldNameに指定されたフィールド名のvalue値(String型)を保存する関数です。

fbchg_tu()関数を使用するため、フィールド順(nth)に既存のデータが存在する場合は与えられたデータ(value)に変更され、フィールド順に既存のデータが存在しない場合は、与えられたデータが自動でフィールドに追加されます。

- プロトタイプ

```
Public Shared Function PUTVAR(ByVal pFBUF As Integer, _ByVal fldName As String,
                             _ByVal nth As Integer, _ByVal value As String)
    As Integer
```

- パラメータ

パラメータ	説明
pFBUF	tpalloc()またはfballoc()を利用してメモリーに割り当てられたバッファです
fldName	フィールド・バッファ(pFBUF)に保存するフィールド名です
nth	フィールド順です
value	保存するString型データです

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 例

```
Dim lsndbuf As Integer
Dim lret As Integer
Dim tempS As String

lsndbuf = fballoc(10, 100)
tempS = txtString.Text

lret = PUTVAR(lsndbuf, " FILENAME ", 2, tempS)
If lret < 0 Then
    error processing
End If
```

- 関連関数

PUTINT()、PUTLONG()、PUTSHORT()、PUTDOUBLE()、PUTFLOAT()、PUTCAR3()、PUTCHR()

5.3. サンプル・プログラム

account idをキー値として照会、修正、削除、入力するプログラムです。バッファはフィールド・キー・バッファを使用します。

5.3.1. プログラムの構成

atmi.vb、fdl.vb、TmaxMacros.vb、WinAPI.vbをモジュールとしてプロジェクトで追加します。

- 共通プログラム

プログラムファイル	説明
demo.f	フィールド・キー・バッファを定義したファイルです
tmconfig.m	Tmax環境設定ファイルです

- クライアント・プログラム

プログラムファイル	説明
AssemblyInfo.vb	アセンブリを行うときに使用されるデフォルト値を設定するファイルです
CodeFile1.vb	Visual Basic .netインターフェースで自動的に作成されるファイルです
EmployeeGrid.resx	Visual Basic .netインターフェースで自動的に作成されるファイルです
EmployeeGrid.vb	照会結果を表示するプログラムです
EmployeeMgr.resx	Visual Basic .netインターフェースで自動的に作成されるファイルです
EmployeeMgr.sln	Visual Basic .netインターフェースで自動的に作成されるファイルです
EmployeeMgr.suo	Visual Basic .netインターフェースで自動的に作成されるファイルです
EmployeeMgr.vb	メイン・プログラムです。照会、修正、削除、入力を行います
EmployeeMgr.vbproj	Visual Basic .netインターフェースで自動的に作成されるファイルです
EmployeeMgr.vbproj.user	Visual Basic .netインターフェースで自動的に作成されるファイルです
TmaxMacros.vb	ユーザーの便宜のためのマクロ定義クラスです
WinAPI.vb	Windowsで提供する関数を呼び出すためのプロトタイプを定義したモジュールです
atmi.vb	atmi関数のプロトタイプを定義したファイルです
fdl.vb	フィールド・キー関数のプロトタイプを定義したファイルです

- サーバー・プログラム

プログラムファイル	説明
emp_c.mk	メイクファイルです
emp_c.pc	サービスを行うサーバー・プログラムです。AIXとOracle 9iを使用します

5.3.2. プログラムの特徴

下記はプログラムの特徴です。

● クライアント・プログラム

機能	説明
Tmax接続	tpstart_t引数で接続します
バッファータ입	フィールド・キー・バッファ、構造体ファイルをfdlcユーティリティでコンパイルして「fdl」ファイルを作成する必要があります
通信タイプ	tpcall()を利用した同期通信を送信するバッファと受信するバッファを別々に指定します
トランザクションの有無	TMSで自動トランザクションを付与します

● サーバー・プログラム

機能	説明
サービス	FDLSELECT、FDLINSERTを作成します
データベース指定	Oracleデータベースを使用します。システム構成ファイルのSVRGROUPにデータベース情報を指定します

5.3.3. 共通プログラム

データベースのEMP表

下記は、データベースに作成する表領域構造の例です。

EMPNO	NUMBER	NOT NULL	P1
ENAME	VARCHAR(16)		
JOB	VARCHAR(16)		
MGR	NUMBER		
HIREDATE	DATE		
SAL	NUMBER(7,2)		
COMM	NUMBER(7,2)		
DEPTNO	NUMBER		

フィールド・キー・バッファの定義

下記は、フィールド・キー・バッファを定義したファイルの例です。

```
<demo.f>
```

```
#For tmax demo employee program
EMPNO          7500          long          -          -
ENAME          7501          string         -          -
JOB            7502          string         -          -
MGR            7503          long           -          -
DATE           7504          string         -          -
SAL            7505          float          -          -
COMM           7506          float          -          -
DEPTNO         7507          long           -          -

E_TYPE         9009          long           -          -
E_CODE         9010          long           -          -
E_MSG          9011          string         -          -
E_TMP          9012          long           -          -
```

Tmax環境設定

下記は、Tmax環境設定ファイルの例です。

<tmconfig.m>

```
*DOMAIN
dom1          SHMKEY = 70000, MAXUSER = 200, MINCLH = 1, MAXCLH = 5,
              TPORTNO = 8888, BLOCKTIME = 200, TXTIME = 200

*NODE
tmax1         TMAXDIR = "/home/tmax",
              APPDIR  = "/home/tmax/appbin",
              PATHDIR = "/home/tmax/path",
              TLOGDIR = "/home/tmax/log/tlog",
              SLOGDIR = "/home/tmax/log/slog",
              ULOGDIR = "/home/tmax/log/ulog"

*SVRGROUP
svg1          NODENAME = tmax1, DBNAME = ORACLE,
              OPENINFO = "ORACLE_XA+Acc=P/scott/tiger+SesTm=60",
              TMSNAME  = svg1_tms

*SERVER
emp_c         SVGNAME = svg1, MIN = 1

*SERVICE
FDLSELECT     SVRNAME = emp_c
FDLUPDATE     SVRNAME = emp_c
FDLDELETE     SVRNAME = emp_c
FDLINSERT     SVRNAME = emp_c
```

5.3.4. クライアント・プログラム

メイン画面

下記は、メイン画面フォームのデザイン画面です。

下記は、例で使用するメイン画面フォームのデザイン構成表です。

制御名	制御タイプ	備考
EmployeeMgrForm	フォーム	Caption="社員管理プログラム"
LabelErr	ラベル	Caption="エラー"
BtnExit	コマンドボタン	Caption="終了"
BtnIns	コマンドボタン	Caption="入力"
BtnDel	コマンドボタン	Caption="削除"
BtnUdt	コマンドボタン	Caption="修正"
BtnSel	コマンドボタン	Caption="照会"
EditName	テキストボックス	
EditEmpNo	テキストボックス	
EditDept	テキストボックス	
EditComm	テキストボックス	
EditSal	テキストボックス	
EditDate	テキストボックス	
EditMgr	テキストボックス	
EditJob	テキストボックス	

制御名	制御タイプ	備考
MList	テキストボックス	
BtnReturn	テキストボックス	

下記は、社員管理のメイン画面デザイン・プログラムの例です。

```
Option Explicit
' 照会画面を閉じる '
Private Sub BtnReturn_Click()
    Hide
    EmployeeMgr.Show
End Sub
' 照会ボタンのクリック時に照会一覧の結果を表示する関数 '
Private Sub Form_Activate()

    ' tpstart 実行関数 '
    tmaxStart

    Dim Isendbuf As Long
    Dim Irecvbuf As Long
    Dim Irbuflen As Long
    Dim Iret As Integer
    Dim text As String
    Dim value As Long
    Dim dvalue As Double
    Dim svalue As Single
    Dim lenL As Long
    Dim initS, outputS As String
    Dim empnoS, enameS, jobS, mgrS, dateS, salS, commS, deptnoS As String
    Dim cntL As Long
    Dim eNo As Long
    Dim txbool As Integer ' トランザクションが開始されると 1、されなければ 0 '

    ' tpstart '
    tmaxStart

    ' トランザクションの開始有無を初期化 '
    txbool = 0

    initS = vbCrLf & vbTab & vbTab & vbTab & "***** 検索結果 *****"
            & vbCrLf & vbTab & "===== "

            & vbCrLf & vbTab & "社員番号 名前 役職 担当役員 入社日 給料 COMM 部署"
            & vbCrLf & vbTab & "===== "
```

```

        & vbCrLf & vbCrLf

If EmployeeMgr.EditEmpNo.text = "" Then
    MsgBox "社員番号を入力してください。"
    Hide
    EmployeeMgr.Show
    Exit Sub
End If

' 送信するデータのためのバッファ割り当て '
Isendbuf = fmalloc(100, 1024)
If Isendbuf = Null Then
    TmaxError ("fmalloc")
    Call fbfree(Isendbuf)
    tmaxEnd
    Exit Sub
End If

' 受信するデータのためのバッファ割り当て '
Irecvbuf = fmalloc(100, 1024)
If Irecvbuf = Null Then
    TmaxError ("fmalloc")
    Call fbfree(Irecvbuf)
    tmaxEnd
    Exit Sub
End If

' 社員管理フォームから社員番号を読み込みます。
eNo = Trim(EmployeeMgr.EditEmpNo.text)
Iret = fbput(ByVal Isendbuf, ByVal fbget_fldkey("EMPNO"), eNo, ByVal lenL)
If Iret = -1 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

' 照会サービスを要求 '
Iret = tpcall(ByVal "FDLSELECT", ByVal Isendbuf, ByVal 0, Irecvbuf, Irbuflen,

                ByVal 0)
If Iret = -1 Then
    ViewErr (Irecvbuf)
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

cntL = fbkeyoccur(ByVal Irecvbuf, ByVal fbget_fldkey("EMPNO"))

Dim i As Long

```

```

For i = 0 To cntL - 1

    Iret = GETLONG(ByVal Irecvbuf, "EMPNO", i, value)
    If Iret = -1 Then
        TmaxError ("GETLONG(EMPNO)")
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    End If
    empnoS = value

    Iret = GETVAR(ByVal Irecvbuf, "ENAME", i, text)
    If Iret = -1 Then
        TmaxError ("GETVAR(ENAME)")
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    End If
    enameS = text

    Iret = GETVAR(ByVal Irecvbuf, "JOB", i, text)
    If Iret = -1 Then
        TmaxError ("GETVAR(JOB)")
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    End If
    jobS = text

    Iret = GETLONG(ByVal Irecvbuf, "MGR", i, value)
    If Iret = -1 Then
        TmaxError ("GETLONG(MGR)")
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    End If
    mgrS = value

    Iret = GETVAR(ByVal Irecvbuf, "DATE", i, text)
    If Iret = -1 Then
        TmaxError ("GETVAR(DATE)")
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    End If
    dateS = text

    ' SAL データフィールドからfloatデータを読み込みます。 '
    ' comm.basに関数が定義されています。 '
    Iret = fbget_tu(ByVal Irecvbuf, ByVal fbget_fldkey("SAL"), i, svalue, 0)
    If Iret = -1 Then
        FdlErrorMsg ("fbget_tu")
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    End If
    salS = svalue

    Iret = GETFLOAT(ByVal Irecvbuf, "COMM", i, svalue)

```



```

    If Iret = -1 Then
        TmaxError ("GETFLOAT(COMM)")
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    End If
    commS = svalue

    Iret = GETLONG(ByVal Irecvbuf, "DEPTNO", i, value)
    If Iret = -1 Then
        TmaxError ("GETLONG(DEPTNO)")
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    End If
    deptnoS = value & vbCrLf

    outputS = outputS & "          " & empnoS & vbTab & enameS & vbTab &
        jobS & vbTab & mgrS & vbTab & dateS & vbTab &
        salS & vbTab & commS & vbTab & deptnoS

Next i

InfoText.text = initS & outputS
' 返されたデータをテキストボックスに出力 '
LabelErr.Caption = "SAL = " & dvalue

Call ExitSub(txbool, Isendbuf, Irecvbuf)

End Sub

Private Sub ExitSub(txbool As Integer, Isendbuf As Long, Irecvbuf As Long)
    If txbool = 1 Then
        tx_rollback
    End If

    ' 割り当てられたバッファを解除 '
    Call fbfree(ByVal Isendbuf&)
    Call fbfree(ByVal Irecvbuf&)

    ' tpend 実行関数 '
    tmaxEnd

Exit Sub

End Sub

' tpstart '
Private Sub tmaxStart()
    Dim ret As Integer

```

```

' 環境設定ファイルを読み込みます。 '
' atmi.bas ファイルに定義されています。 '
' Declare Function tmaxreadenv Lib "TMAX4GL.DLL"(ByVal envfile As String,
                                                ByVal label As String) As Long'

' 詳細説明については、Tmax リファレンスガイドを参照してください。 '
ret = tmaxreadenv("C:\tmax.env", "aix51389")
If ret < 0 Then
    TmaxError ("tmaxreadenv")
    Exit Sub
End If

ret = tpstart(ByVal 0&)
If ret = -1 Then
    LabelErr.Caption = "tpstart error = " & gettperrno()
    TmaxError ("tpstart")
    Exit Sub
Else
    LabelErr.Caption = "tpstart return value = " & ret & " tpstart success"
End If

End Sub

' tpend '
Private Sub tmaxEnd()
    Dim ret As Integer
    ret = tpend()
    If ret = -1 Then
        LabelErr.Caption = "tpend error = " & gettperrno()
        TmaxError ("tpend")
        Exit Sub
    Else
        LabelErr.Caption = "tpreturn return value = " & ret
    End If
End Sub

End Sub

' Oracleのエラーを出力する関数 '
Private Sub ViewErr(ByVal a As Long)

'E_TYPE      9009      long      -      -
'E_CODE      9010      long      -      -
'E_MSG       9011      string    -      -
'E_TMP       9012      long      -      -

    Dim typeL As Long
    Dim codeL As Long
    Dim msgS As String

```

```

Dim tmpL As Long
Dim Iret As Integer

Iret = fbget_tu(ByVal a, ByVal fbget_fldkey("E_TYPE"), 0, typeL, 0)

If Iret = -1 Then
    FdlErrorMsg ("fbget_tu")
    Exit Sub
End If

Iret = fbget_tu(ByVal a, ByVal fbget_fldkey("E_CODE"), 0, codeL, 0)
If Iret = -1 Then
    FdlErrorMsg ("fbget_tu")
    Exit Sub
End If

Iret = GETVAR(ByVal a, "E_MSG", 0, msgS)
Iret = fbget_tu(ByVal a, ByVal fbget_fldkey("E_TMP"), 0, tmpL, 0)
If Iret = -1 Then
    FdlErrorMsg ("fbget_tu")
    Exit Sub
End If

MsgBox ("Error Type : "&typeL&" ,Code : "&codeL&" ,Message : "&msgS&"
        ,Tmp : " & tmpL)

End Sub

```

下記は、社員管理を処理するプログラムの例です。

<EmployeeMgr.vb>

```

Option Strict Off
Option Explicit On

Imports WindowsApplication1.TmaxUtils.TmaxMacros

Public Class EmployeeMgr
    Inherits System.Windows.Forms.Form

    Public Shared EmpNo As Integer

    #Region " Windowsフォーム・デザイナーで作成したコード "
    #End Region

    Private Sub BtnExit_Click(ByVal sender As System.Object,
                              ByVal e As System.EventArgs) Handles BtnExit.Click
        End
    End Sub

```

```

End Sub

Function getEmpNo()
    Return EmpNo
End Function

Protected Overrides Sub Finalize()
    MyBase.Finalize()
End Sub

Private Sub BtnSel_Click(ByVal sender As System.Object,
                        ByVal e As System.EventArgs) Handles BtnSel.Click
    Dim EmpGrid As EmployeeGrid
    EmpGrid = New EmployeeGrid()

    Hide()
    EmpGrid.EmpNo = EditEmpNo.Text()
    EmpGrid.ShowDialog()
    Show()
End Sub

Private Sub BtnUdt_Click(ByVal sender As System.Object,
                        ByVal e As System.EventArgs) Handles BtnUdt.Click

    Dim Isendbuf As Long
    Dim Irecvbuf As Long
    Dim Irbuflen As Long
    Dim Iret As Long
    Dim text As String
    Dim value As Long
    Dim dvalue As Double
    Dim svalue As Single
    Dim txbool As Integer ' トランザクションが開始されると 1、されなければ 0 '

    ' tpstart '
    tmaxStart()

    ' トランザクションの開始有無を初期化 '
    txbool = 0

    If EditEmpNo.Text = "" Then
        MsgBox("社員番号を入力してください。")
        tmaxEnd()
        Exit Sub
    End If

    ' 送信するデータのためのバッファ割り当て '

```

```

Isendbuf = fmalloc(100, 1024)

If Isendbuf = 0 Then
    Call fbfree(Isendbuf)
    tmaxEnd()
    Exit Sub
End If

' 受信するデータのためのバッファ割り当て '
Irecvbuf = fmalloc(100, 1024)

If Irecvbuf = 0 Then
    Call fbfree(Irecvbuf)
    tmaxEnd()
    Exit Sub
End If

Iret = PUTLONG(Isendbuf, "EMPNO", 0, Trim(EditEmpNo.Text))
If Iret = -1 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

Iret = PUTVAR(Isendbuf, "ENAME", 0, Trim(EditName.Text))
If Iret = -1 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

Iret = PUTVAR(Isendbuf, "JOB", 0, Trim(EditJob.Text))
If Iret = -1 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

Iret = PUTLONG(Isendbuf, "MGR", 0, Trim(EditMgr.Text))
If Iret = -1 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

Iret = PUTVAR(Isendbuf, "DATE", 0, Trim(EditDate.Text))
If Iret = -1 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

Iret = PUTFLOAT(Isendbuf, "SAL", 0, Trim(EditSal.Text))
If Iret = -1 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

```

```

Iret = PUTFLOAT(Isendbuf, "COMM", 0, Trim(EditComm.Text))
If Iret = -1 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

Iret = PUTLONG(Isendbuf, "DEPTNO", 0, Trim(EditDept.Text))
If Iret = -1 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

' トランザクション開始 '
Iret = tx_begin()
If Iret = -1 Then
    txbool = 1
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
Else
    txbool = 1
End If

' 修正サービスを要求 '
Iret = tpcall("FDLUPDATE", Isendbuf, 0, Irecvbuf, Irbuflen, 0)

LabelErr.Text = "sbuf = " & Isendbuf & ", rbuf = " & Irecvbuf

If Iret < 0 Then
    ViewErr(Irecvbuf)
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

' 割り当てられたバッファを解除

' トランザクション保存 '
Iret = tx_commit()
If Iret < 0 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
Else
    txbool = 0
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

End Sub

Private Sub BtnDel_Click(ByVal sender As System.Object,
                        ByVal e As System.EventArgs) Handles BtnDel.Click

    Dim Isendbuf As Integer
    Dim Irecvbuf As Integer

```

```

Dim Irbuflen As Long
Dim Iret As Long
Dim text As String
Dim value As Long
Dim dvalue As Double
Dim svalue As Single
Dim txbool As Integer ' トランザクションが開始されると 1、されなければ 0 '

' tpstart '
tmaxStart()

' トランザクションの開始有無を初期化 '
txbool = 0

If EditEmpNo.Text = "" Then
    MsgBox("社員番号を入力してください。")
    tmaxEnd()
    Exit Sub
End If

value = Trim(EditEmpNo.Text)

' 送信するデータのためのバッファ割り当て '
Isendbuf = fballoc(100, 1024)

If Isendbuf = 0 Then
    Call fbfree(Isendbuf)
    tmaxEnd()
    Exit Sub
End If

' 受信するデータのためのバッファ割り当て '
Irecvbuf = fballoc(100, 1024)

If Irecvbuf = 0 Then
    Call fbfree(Irecvbuf)
    tmaxEnd()
    Exit Sub
End If

Iret = PUTLONG(Isendbuf, "EMPNO", 0, value)
If Iret = -1 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

' トランザクション開始 '

```

```

Iret = tx_begin()
If Iret = -1 Then
    txbool = 1
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
Else
    txbool = 1
End If

' 照会サービスを要求 '
Iret = tpcall("FDLDELETE", Isendbuf, 0, Irecvbuf, Irbuflen, 0)
If Iret = -1 Then
    ViewErr(Irecvbuf)
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

' トランザクション保存 '
Iret = tx_commit()
If Iret < 0 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
Else
    txbool = 0
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

End Sub

Private Sub BtnIns_Click(ByVal sender As System.Object,
                        ByVal e As System.EventArgs) Handles BtnIns.Click

    Dim Isendbuf As Long
    Dim Irecvbuf As Long
    Dim Irbuflen As Long
    Dim Iret As Long
    Dim text As String
    Dim value As Long
    Dim dvalue As Double
    Dim svalue As Single
    Dim txbool As Integer ' トランザクションが開始されると 1、されなければ 0 '

    ' tpstart '
    tmaxStart()

    ' トランザクションの開始有無を初期化 '
    txbool = 0

```



```

If EditEmpNo.Text = "" Then
    MsgBox("社員番号を入力してください。")
    tmaxEnd()
    Exit Sub
End If

' 送信するデータのためのバッファ割り当て '
Isendbuf = fmalloc(100, 1024)

If Isendbuf = 0 Then
    Call fbfree(Isendbuf)
    tmaxEnd()
    Exit Sub
End If

' 受信するデータのためのバッファ割り当て '
Irecvbuf = fmalloc(100, 1024)

If Irecvbuf = 0 Then
    Call fbfree(Irecvbuf)
    tmaxEnd()
    Exit Sub
End If

Iret = PUTLONG(Isendbuf, "EMPNO", 0, Trim(EditEmpNo.Text))
If Iret = -1 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

Iret = PUTVAR(Isendbuf, "ENAME", 0, Trim(EditName.Text))
If Iret = -1 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

Iret = PUTVAR(Isendbuf, "JOB", 0, Trim(EditJob.Text))
If Iret = -1 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

Iret = PUTLONG(Isendbuf, "MGR", 0, Trim(EditMgr.Text))
If Iret = -1 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

Iret = PUTVAR(Isendbuf, "DATE", 0, Trim(EditDate.Text))
If Iret = -1 Then

```

```

        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    End If

    Iret = PUTFLOAT(Isendbuf, "SAL", 0, Trim(EditSal.Text))
    If Iret = -1 Then
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    End If

    Iret = PUTFLOAT(Isendbuf, "COMM", 0, Trim(EditComm.Text))
    If Iret = -1 Then
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    End If

    Iret = PUTLONG(Isendbuf, "DEPTNO", 0, Trim(EditDept.Text))
    If Iret = -1 Then
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    End If

    ' トランザクション開始 '
    Iret = tx_begin()
    If Iret = -1 Then
        txbool = 1
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    Else
        txbool = 1
    End If

    ' 修正サービスを要求 '
    Iret = tpcall("FDLINSERT", Isendbuf, 0, Irecvbuf, Irbuflen, 0)

    If Iret = -1 Then
        ViewErr(Irecvbuf)
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    End If

    ' トランザクション保存 '
    Iret = tx_commit()
    If Iret < 0 Then
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    Else
        txbool = 0
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    End If

End Sub

```

```

Private Sub ExitSub(ByVal txbool As Integer, ByVal Isendbuf As Integer,
                    ByVal Irecvbuf As Integer)

    If txbool = 1 Then
        tx_rollback()
    End If

    ' 割り当てられたバッファを解除 '
    Call fbfree(Isendbuf)
    Call fbfree(Irecvbuf)

    ' tpend '
    tmaxEnd()

Exit Sub
End Sub

Private Sub tmaxStart()
    Dim ret As Integer
    Dim envFile As String

    ' 環境設定ファイルを読み込みます。 '
    ' atmi.basファイルに定義されています。 '
    ' Declare Function tmaxreadenv Lib "TMAX4GL.DLL" (ByVal envfile As String,

                                                ByVal label As String)

                                                As Long '

    ' 詳細説明については、Tmax リファレンスガイドを参照してください。 '
    envFile = "C:\tmax.env"
    ret = tmaxreadenv(envFile, "aix51389")
    If ret < 0 Then
        MsgBox("Tmaxreadenv Error" & envFile)
        Exit Sub
    End If

    ret = tpstart(0&)
    If ret = -1 Then
        LabelErr.Text = "tpstart error = " & gettperrno()
    Else
        LabelErr.Text = "tpstart return value = " & ret & " tpstart success"
    End If

End Sub

Private Sub tmaxEnd()
    Dim ret As Integer

```

```

ret = tpend()
If ret = -1 Then
    LabelErr.Text = "tpend error = " & gettperrno()
Else
    LabelErr.Text = "tpreturn return value = " & ret
End If

End Sub

```

Oracleのエラー出力画面

下記は、Oracleのエラー出力画面です。



下記は、Oracleのエラー出力画面を実行するために必要な環境設定の例です。

```

' Oracleのエラーを出力する関数
Private Sub ViewErr(ByVal a As Long)

    'E_TYPE      9009    long    -    -
    'E_CODE      9010    long    -    -
    'E_MSG       9011    string   -    -
    'E_TMP       9012    long     -    -

    Dim typeL As Long
    Dim codeL As Long
    Dim msgS As String
    Dim tmpL As Long
    Dim Iret As Integer

    GETLONG(a, "E_TYPE", 0, typeL)
    GETLONG(a, "E_CODE", 0, codeL)
    GETVAR(a, "E_MSG", 0, msgS)
    GETLONG(a, "E_TMP", 0, tmpL)
    MsgBox("Error Type : " & typeL & " ,Code : " & codeL & " ,Message : " & msgS & " ,Tmp : " & tmpL)

End Sub
End Class

```

照会画面フォームのデザイン画面

下記は、照会画面フォームのデザイン画面です。

下記は、照会画面フォームのデザイン構成表です。

制御名	制御タイプ	修正が必要なプロパティ
EmployeeGrid	フォーム	Caption="社員情報一覧"
ListTextBox	テキストボックス	MultiLine=True
BtnReturn	コマンドボタン	Caption="戻る"

下記は、社員管理照会画面のデザイン・プログラムの例です。

<EmployeeGrid.frm>

```
Option Strict Off
Option Explicit On

Imports WindowsApplication1.TmaxUtils.TmaxMacros

Imports System.IO

Public Class EmployeeGrid
    Inherits System.Windows.Forms.Form

    Public Shared EmpNo As Integer

    #Region " Windowsフォーム・デザイナーで作成したコード "
```

```

#End Region

Private Sub BtnReturn_Click(ByVal sender As System.Object,
                             ByVal e As System.EventArgs) Handles BtnReturn.Click

    Hide()
End Sub

Protected Overrides Sub OnLoad(ByVal e As System.EventArgs)

    Dim Isendbuf, Isendlen As Integer
    Dim Irecvbuf, Irecvlen As Integer
    Dim Irbuflen As Long
    Dim Iret As Integer
    Dim text As String
    Dim value As Long
    Dim dvalue As Double
    Dim svalue As Single
    Dim lenL As Integer
    Dim initS, outputS As String
    Dim empnoS, enameS, jobS, mgrS, dateS, salS, commS, deptnoS As String
    Dim cntL As Long
    Dim eNo As Long
    Dim i As Integer
    Dim txbool As Integer ' トランザクションが開始されると 1、されなければ 0'

    ' tpstart '
    tmaxStart()

    ' トランザクションの開始有無を初期化 '
    txbool = 0

    ' 送信するデータのためのバッファ割り当て '
    Isendbuf = fmalloc(100, 1024)
    If Isendbuf = 0 Then
        Call fbfree(Isendbuf)
        tmaxEnd()
        Exit Sub
    End If

    ' 受信するデータのためのバッファ割り当て '
    Irecvbuf = fmalloc(100, 1024)
    If Irecvbuf = 0 Then
        Call fbfree(Irecvbuf)
        tmaxEnd()
        Exit Sub
    End If

```

```

Iret = fbput(Isendbuf, fbget_fldkey("EMPNO"), EmpNo, lenL)
If Iret = -1 Then
    FdlErrorMsg("fbput")
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

' 照会サービスを要求 '
Iret = tpcall("FDLSELECT", Isendbuf, 0, Irecvbuf, Irbuflen, 0)
If Iret = -1 Then
    ViewErr(Irecvbuf)
    initS = vbCrLf & vbTab & vbTab & vbTab & "***** 検索結果 *****"
        & vbCrLf & vbTab & "===== "

        & vbCrLf & vbTab & "社員番号 名前 役職 担当役員 入社日 給料 COMM 部
署"

        & vbCrLf & vbTab & "===== "

        & vbCrLf & vbCrLf
    ListTextBox.Text = initS & " "
    LabelErr.Text = "SAL = " & dvalue
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

Iret = fbkeyoccur(Irecvbuf, fbget_fldkey("EMPNO"))

For i = 0 To Iret - 1

    ' 返されたデータをテキストボックスに出力 '
    Iret = GETLONG(Irecvbuf, "EMPNO", i, value)
    If Iret = -1 Then
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    End If
    empnoS = value

    Iret = GETVAR(Irecvbuf, "ENAME", i, text)
    If Iret = -1 Then
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    End If
    enameS = text

    Iret = GETVAR(Irecvbuf, "JOB", i, text)
    If Iret = -1 Then
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    End If
    jobS = text

```

```

Iret = GETLONG(Irecvbuf, "MGR", i, value)
If Iret = -1 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If
mgrS = value

Iret = GETVAR(Irecvbuf, "DATE", i, text)
If Iret = -1 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If
dateS = text

Iret = fbget_tu(Irecvbuf, fbget_fldkey("SAL"), i, svalue, 0)
If Iret = -1 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If
salS = svalue

Iret = fbget_tu(Irecvbuf, fbget_fldkey("COMM"), i, svalue, 0)
If Iret = -1 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If
commS = svalue

Iret = GETLONG(Irecvbuf, "DEPTNO", i, value)
If Iret = -1 Then
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If
deptnoS = value & vbCrLf

outputS = outputS & "          " & empnoS & vbTab & enameS & vbTab & jobsS
          & vbTab & mgrS & vbTab & dateS & vbTab & salS & vbTab & commS
          & vbTab & deptnoS

Next i

initS = vbCrLf & vbTab & vbTab & vbTab & "***** 検索結果 *****"
        & vbCrLf & vbTab & "===== "
        & vbCrLf & vbTab & "社員番号 名前 役職 担当役員 入社日 給料 COMM 部署"
        & vbCrLf & vbTab & "===== "
        & vbCrLf & vbCrLf
ListTextBox.Text = initS & outputS
LabelErr.Text = "SAL = " & dvalue

' 割り当てられたバッファを解除 '

```



```

    Call fbfree(Isendbuf)
    Call fbfree(Irecvbuf)

    ' tpend 実行関数 '
    tmaxEnd()
End Sub

Private Sub ExitSub(ByVal txbool As Integer, ByVal Isendbuf As Integer,
                    ByVal Irecvbuf As Integer)

    If txbool = 1 Then
        tx_rollback()
    End If

    ' 割り当てられたバッファを解除 '
    Call fbfree(Isendbuf)
    Call fbfree(Irecvbuf)

    ' tpend '
    tmaxEnd()

End
End Sub

Public Sub tmaxStart()
    Dim ret As Integer
    Dim envFile As String

    ' 環境設定ファイルを呼び出します。 '
    ' atmi.basファイルに定義されています。 '
    ' Declare Function tmaxreadenv Lib "TMAX4GL.DLL" (ByVal envfile As String,
                                                    ByVal label As String) As Long
    ,

    ' 詳細説明については、『Tmax リファレンスガイド』を参照してください。 '
    envFile = "C:\tmax.env"
    ret = tmaxreadenv(envFile, "aix51389")
    If ret < 0 Then
        MsgBox("Tmaxreadenv Error : " & envFile)
    End
End If

    ret = tpstart(0&)
    If ret = -1 Then
        MsgBox("Tpstart Error")
    End
Else

```

```

        LabelErr.Text = "tpstart return value = " & ret & " tpstart success"
    End If

End Sub

Public Sub tmaxEnd()
    Dim ret As Integer
    ret = tpend()
    If ret = -1 Then
        MsgBox("Tpend Error")
    End
    Else
        LabelErr.Text = "tpreturn return value = " & ret
    End If
End Sub

Private Sub ViewErr(ByVal a As Long)

    'E_TYPE      9009    long    -    -
    'E_CODE      9010    long    -    -
    'E_MSG       9011    string   -    -
    'E_TMP       9012    long     -    -

    Dim typeL As Long
    Dim codeL As Long
    Dim msgS As String
    Dim tmpL As Long
    Dim Iret As Integer

    GETLONG(a, "E_TYPE", 0, typeL)
    GETLONG(a, "E_CODE", 0, codeL)
    GETVAR(a, "E_MSG", 0, msgS)
    GETLONG(a, "E_TMP", 0, tmpL)
    MsgBox("Error Type : " & typeL & " ,Code : " & codeL & " ,Message :
        " & msgS & " ,Tmp : " & tmpL)

End Sub

End ClassOption Strict Off
Option Explicit On

Imports WindowsApplication1.TmaxUtils.TmaxMacros

Imports System.IO

Public Class EmployeeGrid
    Inherits System.Windows.Forms.Form

```

```

Public Shared EmpNo As Integer

#Region " Windowsフォームのデザイナーで作成したコード "

Public Sub New()
    MyBase.New()

    'この呼び出しはWindowsフォーム・デザイナーが必要です。
    InitializeComponent()

    'InitializeComponent()の呼び出し後、初期化作業を追加してください。

End Sub

'フォームはDisposeを再定義して構成要素の一覧を整理します。
Protected Overloads Overrides Sub Dispose(ByVal disposing As Boolean)
    If disposing Then
        If Not (components Is Nothing) Then
            components.Dispose()
        End If
    End If
    MyBase.Dispose(disposing)
End Sub

'Windowsフォーム・デザイナーが必要です。
Private components As System.ComponentModel.IContainer

'参考: 下記のプロシージャは、Windowsフォーム・デザイナーが必要です。
'Windowsフォーム・デザイナーを使用して修正できます。
'コード・エディターを使用して修正しないでください。
Friend WithEvents BtnReturn As System.Windows.Forms.Button
Friend WithEvents LabelErr As System.Windows.Forms.Label
Protected WithEvents ListTextBox As System.Windows.Forms.TextBox
<System.Diagnostics.DebuggerStepThrough> Private Sub InitializeComponent()
    Me.ListTextBox = New System.Windows.Forms.TextBox()
    Me.BtnReturn = New System.Windows.Forms.Button()
    Me.LabelErr = New System.Windows.Forms.Label()
    Me.SuspendLayout()
    '
    'ListTextBox
    '
    Me.ListTextBox.Location = New System.Drawing.Point(10, 12)
    Me.ListTextBox.Multiline = True
    Me.ListTextBox.Name = "ListTextBox"
    Me.ListTextBox.Size = New System.Drawing.Size(472, 272)
    Me.ListTextBox.TabIndex = 0
    Me.ListTextBox.TabStop = False

```

```

Me.ListTextBox.Text = "TextBox" & Microsoft.VisualBasic.ChrW(13)
                        & Microsoft.VisualBasic.ChrW(10) & "Hellow"
'
'BtnReturn
'
Me.BtnReturn.Location = New System.Drawing.Point(392, 288)
Me.BtnReturn.Name = "BtnReturn"
Me.BtnReturn.Size = New System.Drawing.Size(88, 32)
Me.BtnReturn.TabIndex = 1
Me.BtnReturn.Text = "戻る"
'
'LabelErr
'
Me.LabelErr.Location = New System.Drawing.Point(20, 294)
Me.LabelErr.Name = "LabelErr"
Me.LabelErr.Size = New System.Drawing.Size(350, 23)
Me.LabelErr.TabIndex = 2
Me.LabelErr.Text = "Label1"
'
'EmployeeGrid
'
Me.AutoScaleBaseSize = New System.Drawing.Size(6, 14)
Me.ClientSize = New System.Drawing.Size(492, 323)
Me.Controls.AddRange(New System.Windows.Forms.Control()
                        {Me.LabelErr, Me.BtnReturn, Me.ListTextBox})
Me.Name = "EmployeeGrid"
Me.Text = "社員情報一覧"
Me.ResumeLayout(False)

End Sub

#End Region

Private Sub BtnReturn_Click(ByVal sender As System.Object,
                            ByVal e As System.EventArgs) Handles BtnReturn.Click

    Hide()
End Sub

Protected Overrides Sub OnLoad(ByVal e As System.EventArgs)

    Dim Isendbuf, Isendlen As Integer
    Dim Irecvbuf, Irecvlen As Integer
    Dim Irbuflen As Long
    Dim Iret As Integer
    Dim text As String
    Dim value As Long

```

```

Dim dvalue As Double
Dim svalue As Single
Dim lenL As Integer
Dim initS, outputS As String
Dim empnoS, enameS, jobS, mgrS, dateS, salS, commS, deptnoS As String
Dim cntL As Long
Dim eNo As Long
Dim i As Integer
Dim txbool As Integer ' トランザクションが開始されると 1、されなければ 0 '

' tpstart '
tmaxStart()

' トランザクションの開始有無を初期化 '
txbool = 0

' 送信するデータのためのバッファ割り当て '
Isendbuf = fmalloc(100, 1024)
If Isendbuf = 0 Then
    Call fbfree(Isendbuf)
    tmaxEnd()
    Exit Sub
End If

' 受信するデータのためのバッファ割り当て '
Irecvbuf = fmalloc(100, 1024)
If Irecvbuf = 0 Then
    Call fbfree(Irecvbuf)
    tmaxEnd()
    Exit Sub
End If

Iret = fbput(Isendbuf, fbget_fldkey("EMPNO"), EmpNo, lenL)
If Iret = -1 Then
    FdlErrorMsg("fbput")
    Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

' 照会サービスを要求 '
Iret = tpcall("FDLSELECT", Isendbuf, 0, Irecvbuf, Irbuflen, 0)
If Iret = -1 Then
    ViewErr(Irecvbuf)
    initS = vbCrLf & vbTab & vbTab & vbTab & "***** 検索結果 *****"
        & vbCrLf & vbTab & "===== "

        & vbCrLf & vbTab & "社員番号 名前 役職 担当役員 入社日 給料 COMM 部署"

```

```

        & vbCrLf & vbTab & "===== "

        & vbCrLf & vbCrLf
ListTextBox.Text = initS & " "
LabelErr.Text = "SAL = " & dvalue
Call ExitSub(txbool, Isendbuf, Irecvbuf)
End If

Iret = fbkeyoccur(Irecvbuf, fbget_fldkey("EMPNO"))

For i = 0 To Iret - 1

    ' 返されたデータをテキストボックスに出力 '
    Iret = GETLONG(Irecvbuf, "EMPNO", i, value)
    If Iret = -1 Then
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    End If
    empnoS = value

    Iret = GETVAR(Irecvbuf, "ENAME", i, text)
    If Iret = -1 Then
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    End If
    enameS = text

    Iret = GETVAR(Irecvbuf, "JOB", i, text)
    If Iret = -1 Then
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    End If
    jobS = text

    Iret = GETLONG(Irecvbuf, "MGR", i, value)
    If Iret = -1 Then
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    End If
    mgrS = value

    Iret = GETVAR(Irecvbuf, "DATE", i, text)
    If Iret = -1 Then
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    End If
    dateS = text

    Iret = fbget_tu(Irecvbuf, fbget_fldkey("SAL"), i, svalue, 0)
    If Iret = -1 Then
        Call ExitSub(txbool, Isendbuf, Irecvbuf)
    End If

```

```

        sals = svalue

        Iret = fbget_tu(Irecvbuf, fbget_fldkey("COMM"), i, svalue, 0)
        If Iret = -1 Then
            Call ExitSub(txbool, Isendbuf, Irecvbuf)
        End If
        commS = svalue

        Iret = GETLONG(Irecvbuf, "DEPTNO", i, value)
        If Iret = -1 Then
            Call ExitSub(txbool, Isendbuf, Irecvbuf)
        End If
        deptnoS = value & vbCrLf

        outputS = outputS & "          " & empnoS & vbTab & enameS & vbTab & jobs
&
        vbTab & mgrS & vbTab & dateS & vbTab & sals & vbTab & commS &
        vbTab & deptnoS

    Next i

    initS = vbCrLf & vbTab & vbTab & vbTab & "***** 検索結果 *****"
        & vbCrLf & vbTab & "===== "
        & vbCrLf & vbTab & "社員番号 名前 役職 担当役員 入社日 給料 COMM 部署"
        & vbCrLf & vbTab & "===== "
        & vbCrLf & vbCrLf
    ListTextBox.Text = initS & outputS
    LabelErr.Text = "SAL = " & dvalue

    ' 割り当てられたバッファを解除 '
    Call fbfree(Isendbuf)
    Call fbfree(Irecvbuf)

    ' tpend 実行関数 '
    tmaxEnd()
End Sub

Private Sub ExitSub(ByVal txbool As Integer, ByVal Isendbuf As Integer,
                    ByVal Irecvbuf As Integer)

    If txbool = 1 Then
        tx_rollback()
    End If

    ' 割り当てられたバッファを解除 '
    Call fbfree(Isendbuf)
    Call fbfree(Irecvbuf)

```

```

        ' tpend '
    tmaxEnd()

End
End Sub

Public Sub tmaxStart()
    Dim ret As Integer
    Dim envFile As String

    ' 環境設定ファイルを呼び出します。 '
    ' atmi.basファイルに定義されています。 '
    ' Declare Function tmaxreadenv Lib "TMAX4GL.DLL" (ByVal envfile As String,
                                                                    ByVal label As String) As Long
    ,

    ' 詳細説明については、『Tmax リファレンスガイド』を参照してください。 '
    envFile = "C:\tmax.env"
    ret = tmaxreadenv(envFile, "aix51389")
    If ret < 0 Then
        MsgBox("Tmaxreadenv Error : " & envFile)
    End
End If

    ret = tpstart(0&)
    If ret = -1 Then
        MsgBox("Tpstart Error")
    End
Else
    LabelErr.Text = "tpstart return value = " & ret & " tpstart success"
End If

End Sub

Public Sub tmaxEnd()
    Dim ret As Integer
    ret = tpend()
    If ret = -1 Then
        MsgBox("Tpend Error")
    End
Else
    LabelErr.Text = "tpreturn return value = " & ret
End If
End Sub

Private Sub ViewErr(ByVal a As Long)

```



```

'E_TYPE      9009    long    -    -
'E_CODE      9010    long    -    -
'E_MSG       9011    string   -    -
'E_TMP       9012    long     -    -

Dim typeL As Long
Dim codeL As Long
Dim msgS As String
Dim tmpL As Long
Dim Iret As Integer

GETLONG(a, "E_TYPE", 0, typeL)
GETLONG(a, "E_CODE", 0, codeL)
GETVAR(a, "E_MSG", 0, msgS)
GETLONG(a, "E_TMP", 0, tmpL)
MsgBox("Error Type : " & typeL & " ,Code : " & codeL & " ,Message :
      " & msgS & " ,Tmp : " & tmpL)

End Sub
End Class

```

5.3.5. サーバー・プログラム

サービス・プログラム

下記は、サービスを行うサーバー・プログラムの例です。

<emp_c.pc>

```

#include <stdio.h>
#include <ctype.h>
#include <tuxinc/macro.h>
#include "../fcl/demo_fcl.h"

EXEC SQL include sqlca.h;
EXEC SQL INCLUDE ORACA;
EXEC ORACLE  OPTION (ORACA=YES);
EXEC ORACLE  OPTION (RELEASE_CURSOR=YES);

#define INP    1
#define ORA    2
#define TMX    3
#define APP    4

EXEC SQL begin declare section;
int  h_empno;

```

```

char h_ename[11];
char h_job[10];
int h_mgr;
char h_date[11];
float h_sal;
float h_comm;
int h_deptno;
EXEC SQL end declare section;

void svc_error(long type, long err_code, char *msg, long tmp);

FDLINSERT( TPSVCINFO *msg )
{
    FBUF *rcvbuf;
    int i, occurrence;
    rcvbuf = (FBUF *)msg->data;
    h_empno = h_mgr = h_sal = h_comm = h_deptno = 0;

    memset( h_ename, 0x00, sizeof( h_ename ) );
    memset( h_job, 0x00, sizeof( h_job ) );
    memset( h_date, 0x00, sizeof( h_date ) );

    occurrence = fbkeyoccur(rcvbuf, EMPNO);

    for (i=0; i< occurrence; i++){
        fbget_tu ( rcvbuf, EMPNO, i, (char *)&h_empno, 0 );
        fbget_tu ( rcvbuf, MGR, i, (char *)&h_mgr, 0 );
        fbget_tu ( rcvbuf, SAL, i, (char *)&h_sal, 0 );
        fbget_tu ( rcvbuf, COMM, i, (char *)&h_comm, 0 );
        fbget_tu ( rcvbuf, DEPTNO,i, (char *)&h_deptno, 0 );
        fbget_tu ( rcvbuf, ENAME, i, (char *)h_ename, 0 );
        fbget_tu ( rcvbuf, JOB , i, (char *)h_job, 0 );
        fbget_tu ( rcvbuf, DATE , i, (char *)h_date, 0 );

        EXEC SQL INSERT
        INTO emp( empno, ename, job, mgr, hiredate, sal,comm, deptno)
        VALUES ( :h_empno, :h_ename, :h_job, :h_mgr,
                  to_date(:h_date,'yyyymmdd'), :h_sal, :h_comm, :h_deptno );
    }

    if(sqlca.sqlcode != 0)
        goto LB_DBERROR;

    EXEC SQL WHENEVER SQLERROR
        goto LB_DBERROR;

    tpreturn(TPSUCCESS, 0L, (char *)rcvbuf, 0L, 0L);

    LB_DBERROR :
        EXEC SQL WHENEVER SQLERROR CONTINUE;
        svc_error(ORA, sqlca.sqlcode, sqlca.sqlerrm.sqlerrmc, 0) ;
}

```

```

}

FDLDELETE( TPSVCINFO *msg )
{
    FBUF *rcvbuf;
    int i, occurrence;

    rcvbuf = ( FBUF *)msg->data;
    occurrence = fbkeyoccur(rcvbuf, EMPNO);

    for (i=0; i< occurrence; i++){
        fbget_tu (rcvbuf, EMPNO, i, (char *)&h_empno , 0);

        EXEC SQL DELETE
        FROM emp
        WHERE empno = :h_empno;
    }
    if(sqlca.sqlcode != 0)
        goto LB_DBERROR;

    EXEC SQL WHENEVER SQLERROR
        goto LB_DBERROR ;

    EXEC SQL WHENEVER NOT FOUND
        goto LB_DBERROR ;

    tpreturn(TPSUCCESS, 0L, (char *)rcvbuf, 0L, 0L);

    LB_DBERROR :
        EXEC SQL WHENEVER SQLERROR CONTINUE;
        svc_error(ORA, sqlca.sqlcode, sqlca.sqlerrm.sqlerrmc, 0) ;
}

FDLSELECT( TPSVCINFO *msg )
{
    FBUF *rcvbuf;
    FLDLEN fldlen;
    int i=0, ROWMEM=200, CHKROW=500;

    rcvbuf = (FBUF *)msg->data;

    if((rcvbuf=(FBUF *)tprealloc((char *)rcvbuf,ROWMEM*CHKROW))==NULL){
        rcvbuf=(FBUF *)msg->data;
        goto LB_TMAXERROR ;
    }

    h_empno = h_mgr = h_sal = h_comm = h_deptno = 0;

    memset( h_ename, 0x00, sizeof( h_ename ) );
    memset( h_job, 0x00, sizeof( h_job ) );

```

```

memset( h_date, 0x00, sizeof( h_date ) );

fbget_tu( rcvbuf, EMPNO, 0, (char *)&h_empno, &fldlen);

EXEC SQL DECLARE DB_CUR CURSOR FOR
SELECT  nvl(empno,0), nvl(ename,' '), nvl(job,' '),
        nvl(to_char(hiredate,'yyyymmdd'),' '), nvl(mgr,0),
        nvl(sal,0), nvl(comm,0), nvl(deptno,0)
FROM    emp
WHERE   empno >= :h_empno-50 AND empno <= :h_empno+50;
EXEC SQL OPEN DB_CUR;

if(sqlca.sqlcode != 0)
    goto LB_DBERROR;
EXEC SQL WHENEVER SQLERROR
    goto LB_DBERROR ;
EXEC SQL WHENEVER NOT FOUND
    Do break ;

while(1) {
    EXEC SQL FETCH DB_CUR
    INTO      :h_empno,
              :h_ename,
              :h_job,
              :h_date,
              :h_mgr,
              :h_sal,
              :h_comm,
              :h_deptno;

    fbchg_tu(rcvbuf, EMPNO, i, (char *)&h_empno, 0);
    fbchg_tu(rcvbuf, MGR,   i, (char *)&h_mgr, 0);
    fbchg_tu(rcvbuf, SAL,   i, (char *)&h_sal, 0);
    fbchg_tu(rcvbuf, DEPTNO, i, (char *)&h_deptno, 0);
    fbchg_tu(rcvbuf, COMM,  i, (char *)&h_comm, 0);
    fbchg_tu(rcvbuf, ENAME, i, (char *)h_ename, 0);
    fbchg_tu(rcvbuf, JOB,   i, (char *)h_job, 0);
    fbchg_tu(rcvbuf, DATE,  i, (char *)h_date, 0);

    i++;
}

if (i < 1) goto LB_DBERROR;

EXEC SQL CLOSE DB_CUR;

tpreturn(TPSUCCESS, 0L, (char *)rcvbuf, 0L, 0L);

LB_DBERROR :
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    EXEC SQL CLOSE DB_CUR ;

```

```

        svc_error(ORA, sqlca.sqlcode, sqlca.sqlerrm.sqlerrmc, 0L) ;

LB_TMAXERROR :
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    EXEC SQL CLOSE DB_CUR ;
    svc_error(TMX, tperrno, "", 0L) ;
}

FDLUPDATE( TPSVCINFO *msg )
{

    FBUF *rcvbuf ;
    int i, occurrence;
    rcvbuf = (FBUF *)msg->data;
    h_empno = h_mgr = h_sal = h_comm = h_deptno = 0;

    memset( h_ename, 0x00, sizeof( h_ename ) );
    memset( h_job, 0x00, sizeof( h_job ) );
    memset( h_date, 0x00, sizeof( h_date ) );

    occurrence = fbkeyoccur(rcvbuf, EMPNO);

    for (i=0; i< occurrence; i++){
        fbget_tu ( rcvbuf, EMPNO, i, (char *)&h_empno, 0);
        fbget_tu ( rcvbuf, ENAME, i, (char *)h_ename, 0);
        fbget_tu ( rcvbuf, JOB, i, (char *)h_job, 0);
        fbget_tu ( rcvbuf, MGR, i, (char *)&h_mgr, 0);
        fbget_tu ( rcvbuf, SAL, i, (char *)&h_sal, 0);
        fbget_tu ( rcvbuf, COMM, i, (char *)&h_comm, 0);
        fbget_tu ( rcvbuf, DEPTNO, i, (char *)&h_deptno, 0);
        fbget_tu ( rcvbuf, DATE, i, (char *)h_date, 0 );

        EXEC SQL UPDATE emp
        SET ename = nvl(:h_ename, ename),
            job = nvl(:h_job, job),
            mgr = :h_mgr,
            hiredate = nvl(to_date(:h_date, 'yyyymmdd'), hiredate),
            sal = :h_sal,
            comm = :h_comm,
            deptno = :h_deptno
        WHERE empno = :h_empno;

        if(sqlca.sqlcode != 0)
            goto LB_DBERROR;

        EXEC SQL WHENEVER SQLERROR
            goto LB_DBERROR;

        EXEC SQL WHENEVER NOT FOUND
            goto LB_DBERROR;
    }
}

```

```

        tpreturn(TPSUCCESS, 0L, (char *)rcvbuf, 0L, 0L);

    LB_DBERROR :
        EXEC SQL WHENEVER SQLERROR CONTINUE;
        svc_error(ORA, sqlca.sqlcode, sqlca.sqlerrm.sqlerrmc, 0L) ;
}

/*****
 * エラー処理:サービスでエラーの発生時にそのエラーをバッファに入れてクライアントに転送します。
 *****/
void svc_error(long type, long err_code, char *msg, long tmp) {
    FBUF *transf;
    char *svcname;
    char err_msg[100];
    char temp[100];
    int i = 0;

    printf("type      ==>[%ld]\n", type);
    printf("err_code ==>[%ld]\n", err_code);
    printf("msg        ==>[%s]\n",  msg);
    strcpy(err_msg, msg);

    if ((transf = (FBFR *)tpalloc("FML", NULL, 0)) == NULL) {
        printf("tpalloc failed! errno = %d\n", tperrno);
    }

    switch(type) {
        case INP:
            switch(err_code) {
                case -1000:
                    strcpy(err_msg, "当該ユーザーに権限がありません。");
                    break;
                default:
                    strcpy(err_msg,
                        "Inputエラー・メッセージが登録されていません。");
            }
            break;
        case ORA:
            if (strlen(err_msg)== 0) strcpy(err_msg, sqlca.sqlerrm.sqlerrmc);
            break;
        case TMX:
            if (strlen(err_msg)== 0) strcpy(err_msg, tpstrerror(tperrno));
            break;
        case APP:
            if (strlen(err_msg)== 0) {
                /* err_mssg=""の場合はエラー・メッセージを設定します。  */
                switch(err_code) {
                    case -500: /* システム関連エラー */
                        strcpy(err_msg, "ファイル作成エラーです。");
                        break;
                    case -502:

```

```

        strcpy(err_msg, "内部サービスが呼び出せませんでした。");
        break;
    case -504:
        strcpy(err_msg, "ソケット通信エラーです。");
        break;
    case -505: /* 他のトランザクションで修正された場合はMSG処理*/
        /* "照会の後、他の場所で[%s]データが変更されました。
        \n\nを再照会した後処理してください。": クライアントで処理*/
        strcpy(err_msg, "がありません。");
        break;
    case -5002:
        strcpy(err_msg, "電算室にお問い合わせください。");
        break;
    default:
        strcpy(err_msg,
            "アプリケーション・エラー・メッセージが登録されていません。");
    }
}
break;
}

/* ROLLBACK */
EXEC SQL WHENEVER SQLERROR CONTINUE;
EXEC SQL ROLLBACK;

fbchg_tu ( transf, E_TYPE, i, (char *)&type,0);
fbchg_tu ( transf, E_CODE, i, (char *)&err_code,0);
fbchg_tu ( transf, E_MSG, i, (char *)&err_msg,0);
fbchg_tu ( transf, E_TMP, i, (char *)&tmp,0);

tpreturn(TPFAIL, 0, (char *)transf, 0L, 0L);
}

```

メイクファイル

下記は、<emp_c.pc>ソースをTmaxアプリケーションに作成するメイクファイルの例です。

<emp_c.mk>

```

include $(ORACLE_HOME)/precomp/lib/env32.mk
ORALIBDIR = $(LIBHOME)
ORALIB = -L/home/oracle/OraHome/lib32/ -lclntsh -lld -lm
        `cat /home/oracle/OraHome/lib32/sysliblist` -lm -lc_r -lpthreads
TARGET = emp_c
APOBJS = emp_c.o
NSDLOBJ = $(TMAXDIR)/lib/sdl.o
#CC
CC=cc

```

```

#Oracle
LIBS = -lsvr -loras

OBS = $(APOBS) $(SVCTOBJ)
SVCTOBJ = $(TARGET)_svctab.o

CFLAGS = -q32 -O -I$(TMAXDIR)
LDFLAGS = -brtl
APPDIR = $(TMAXDIR)/appbin
SVCTDIR = $(TMAXDIR)/svct
TMAXLIBDIR = $(TMAXDIR)/lib
#
.SUFFIXES : .c

.c.o:
    $(CC) $(CFLAGS) $(LDFLAGS) -c $<

all: $(TARGET)

$(TARGET): $(OBS)
    $(CC) $(CFLAGS) $(LDFLAGS) -L$(TMAXLIBDIR) -o $(TARGET)
    -L$(ORALIBDIR) $(ORALIB) $(OBS) $(LIBS) $(NSDLOBJ)
    mv $(TARGET) $(TMAXDIR)/appbin

$(APOBS): $(TARGET).pc
    proc iname=emp_c include=$(TMAXDIR) define=__LINUX_ORACLE_PROC__
    $(CC) $(CFLAGS) $(LDFLAGS) -c $(TARGET).c

$(SVCTOBJ):
    touch $(SVCTDIR)/$(TARGET)_svctab.c
    $(CC) $(CFLAGS) $(LDFLAGS) -c $(SVCTDIR)/$(TARGET)_svctab.c
#
clean:
    -rm -f *.o core $(TARGET) $(TARGET).lis

```


第6章 C# .netインターフェース

本章では、C# .netインターフェース・モジュールと例について説明します。

参考

UnicodeをサポートするC# .netインターフェースは、Visual Basic .netのインターフェースと類似しているため、本章ではその説明については省略します。詳細内容は、「[第4章 Visual Basic .netインターフェース\(Unicode\)](#)」を参照してください。

6.1. 概要

C# .netインターフェースには、クライアント・ライブラリーの呼び出しができるように関数のプロトタイプを定義したインターフェース・モジュールが存在します。

開発者は、下記のインターフェース・モジュールをインストールすれば、Tmaxクライアント・ライブラリーで提供する関数を呼び出して使用することができます。関数の詳しい使用方法については、「[6.2.4. クライアント・プログラム](#)」のUnicodeサポート社員管理プログラムのデザイン・ソースの例を参照してください。

モジュール	説明
Atmi.cs	atmi関数のプロトタイプ定義モジュールです
Fdl.cs	フィールド・キー関数のプロトタイプ定義モジュールです
Tx.cs	トランザクション関連関数のプロトタイプ定義モジュールです
WinApi.cs	Windowsで提供する関数を呼び出すためのプロトタイプ定義モジュールです
TmaxApi.cs	RQ、非要求メッセージなどの関数のプロトタイプ定義モジュールです

参考

atmiks関数とフィールド・キー関数のプロトタイプおよび機能についての説明は、『Tmax リファレンスガイド』および『Tmax アプリケーション開発ガイド』を参照してください。サンプル・プログラムを利用して、C#インターフェースの使用方法について説明します。

6.2. サンプル・プログラム

社員番号をキー値として、名前、役職、担当役員、入社日、給料、契約期間、部署などを照会、修正、削除、入力するプログラムです。バッファはフィールド・バッファを使用します。

6.2.1. プログラムの構成

プログラムは下記のとおり構成されます。

- 共通プログラム

プログラムファイル	説明
demo.f	フィールド・キー・バッファを定義したファイルです
tmconfig.m	Tmax環境設定ファイルです

- クライアント・プログラム

プログラムファイル	説明
4GL_Sample.sln	Form1.csとTmaxライブラリーで構成されているクライアント・プログラムです

- サーバー・プログラム

プログラムファイル	説明
emp_c.pc	サービス・プログラム(Oracleソース)です
emp_c.mk	メイクファイルです

6.2.2. プログラムの特徴

下記はプログラムの特徴です。

- クライアント・プログラム

機能	説明
Tmaxライブラリー接続	Atmi.cs、Fdl.cs、Tx.cs、WinApi.cs、TmaxApi.csを項目に追加します
Tmax接続	サービスを実行するたびに接続し、サービスを終了すると接続を解除します
バッファータ입	フィールド・キー・バッファ、フィールド・キー・ファイルをfdlcユーティリティーでコンパイルして「fdl」ファイルを作成する必要があります
通信タイプ	pb_tpcall()を利用して同期通信を行います
トランザクションの有無	照会、修正、削除、入力する場合、すべてトランザクションで処理します

- サーバー・プログラム

機能	説明
サービス	FDLSELECT、FDLUPDATE、FDLDELETE、FDLINSERTを作成します
データベース指定	Oracleデータベースを使用します。システム構成ファイルのSVRGROUPにデータベース情報を指定します(XA方式)

6.2.3. 共通プログラム

データベースのEMP表

下記は、データベースに作成するEMP表ファイルの例です。

EMPNO	NUMBER	NOT NULL	P1
ENAME	VARCHAR(16)		
JOB	VARCHAR(16)		
MGR	NUMBER		
HIREDATE	DATE		
SAL	NUMBER(7,2)		
COMM	NUMBER(7,2)		
DEPTNO	NUMBER		

フィールド・キー・バッファの定義

下記は、フィールド・キー・バッファを定義したファイルの例です。

<demo.f>

#For tmax demo employee program				
EMPNO	7500	long	-	-
ENAME	7501	string	-	-
JOB	7502	string	-	-
MGR	7503	long	-	-
DATE	7504	string	-	-
SAL	7505	float	-	-
COMM	7506	float	-	-
DEPTNO	7507	long	-	-
E_TYPE	9009	long	-	-
E_CODE	9010	long	-	-
E_MSG	9011	string	-	-
E_TMP	9012	long	-	-

Tmax環境設定

下記は、Tmax環境設定ファイルの例です。

<tmconfig.m>

```
*DOMAIN
dom1          SHMKEY = 70000, MAXUSER = 200, MINCLH = 1, MAXCLH = 5,
               TPORTNO = 8888, BLOCKTIME = 200, TXTIME = 200

*NODE
tmax1         TMAXDIR = "/home/tmax",
               APPDIR = "/home/tmax/appbin",
               PATHDIR = "/home/tmax/path",
               TLOGDIR = "/home/tmax/log/tlog",
               SLOGDIR = "/home/tmax/log/slog"
               ULOGDIR = "/home/tmax/log/ulog"

*SVRGROUP
svg1          NODENAME = tmax1, DBNAME = ORACLE,
               OPENINFO = "ORACLE_XA+Acc=P/scott/tiger+SesTm=60",
               TMSNAME = svg1_tms

*SERVER
emp_c         SVGNAME = svg1, MIN = 1

*SERVICE
FDLSELECT     SVRNAME = emp_c
FDLUPDATE     SVRNAME = emp_c
FDLDELETE     SVRNAME = emp_c
FDLINSERT     SVRNAME = emp_c
```

6.2.4. クライアント・プログラム

社員管理プログラム

下記は、プログラムの実行時に表示される初期画面です。

画面は8つの単一行エディター、1つのデータ画面、5つのボタンで構成されます。

下記は、各ボタンの機能についての説明です。

- 照会

「社員番号」を入力して[照会]ボタンをクリックすると、社員番号±50の範囲内にあるデータがデータ・ウィンドウに出力されます。

以下はプログラムの照会結果画面です。

- 修正

「社員番号」と修正したいデータを入力して[修正]ボタンをクリックすると、入力されたデータだけが修正されます。

- 削除

「社員番号」でのみ削除が可能です。

- 入力

すべてのデータを入力します。

- 終了

プログラムを終了します。

本節では、Unicodeと一般プログラミングの2つのプログラム・デザインソースの例について説明します。

まず、一般プログラミングの社員管理プログラム・デザインソースの例は下記のとおりです。

<Form1.cs>

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;

using AtmiNS;
using WinApiNS;
using FdlNS;
using TxNS;
using TmaxApiNS;

namespace _4GL_Sample
{
    public class EmployeeMgr : System.Windows.Forms.Form
    {
        private Atmi atmi = new Atmi();
        private WinApi winApi = new WinApi();
        private Fdl fdl = new Fdl();
        private Tx tx = new Tx();
        private TmaxApi tmaxApi = new TmaxApi();

        bool bTxRun, bConnect;
        int nSndBuf, nRcvBuf;
        public string[] EmpFKey =
        {"EMPNO", "ENAME", "JOB", "MGR", "DATE", "SAL", "COMM", "DEPTNO"};
        public string[] ErrFKey = {"E_TYPE", "E_CODE", "E_MSG", "E_TMP"};
        public string[] ColumnName =
        {"社員番号", "名前", "役職", "担当役員", "入社日", "給料", "COMM", "部署名"};

        public const int FIELD_NUM = 8;
        public const int ErrFLD_NUM = 4;
    }
}
```

```

public enum InputCheck { Pri_Key, All };

private System.Windows.Forms.Label label1;
private System.Windows.Forms.Label label2;
private System.Windows.Forms.Label label3;
private System.Windows.Forms.Label label4;
private System.Windows.Forms.Label label5;
private System.Windows.Forms.Label label6;
private System.Windows.Forms.Label label7;
private System.Windows.Forms.Label label8;
private System.Windows.Forms.GroupBox groupBox1;
private System.Windows.Forms.TextBox EditEmpNo;
private System.Windows.Forms.TextBox EditName;
private System.Windows.Forms.TextBox EditJob;
private System.Windows.Forms.TextBox EditMgr;
private System.Windows.Forms.TextBox EditDate;
private System.Windows.Forms.TextBox EditSal;
private System.Windows.Forms.TextBox EditComm;
private System.Windows.Forms.TextBox EditDept;
private System.Windows.Forms.Label LabelErr;
private System.Windows.Forms.Button BtnSel;
private System.Windows.Forms.Button BtnUpt;
private System.Windows.Forms.Button BtnDel;
private System.Windows.Forms.Button BtnIns;
private System.Windows.Forms.Button BtnExit;
private System.Windows.Forms.Button BtnBack;
private System.Windows.Forms.DataGrid dataGrid;

private DataSet ResultData;
private DataTable tEmp;
private DataColumn cEmpNo;
private DataColumn cENAME;
private DataColumn cJob;
private DataColumn cMgr;
private DataColumn cDate;
private DataColumn cSal;
private DataColumn cComm;
private DataColumn cDept;
private System.ComponentModel.Container components = null;

public EmployeeMgr()
{
    bTxRun = false;
    bConnect = false;
    nSndBuf = 0;
    nRcvBuf = 0;
}

```

```

        InitializeComponent();
        MakeDataSet();
    }

    private void ErrorProcess(string ErrMsg)
    {
        LabelErr.Text = ErrMsg;
        if( bTxRun == true )
        {
            tx.TX_ROLLBACK();
            bTxRun = false;
        }
        if( nRcvBuf != 0 )
            atmi.TPFREE(ref nRcvBuf);
        if( nSndBuf != 0 )
            atmi.TPFREE(ref nSndBuf);

        if( bConnect == true )
        {
            atmi.TPEND();
            bConnect = false;
        }
    }

    private void SuccessProcess(string SucMsg)
    {
        LabelErr.Text = SucMsg;
        if( bTxRun == true ) {
            tx.TX_COMMIT();
            bTxRun = false;
        }
        if( nRcvBuf != 0 )
            atmi.TPFREE(ref nRcvBuf);
        if( nSndBuf != 0 )
            atmi.TPFREE(ref nSndBuf);
        if( bConnect == true ) {
            atmi.TPEND();
            bConnect = false;
        }
    }

    private void ClearWindow() {
        EditEmpNo.Clear();
        EditName.Clear();
        EditJob.Clear();
        EditMgr.Clear();
        EditDate.Clear();
    }

```



```

        EditSal.Clear();
        EditComm.Clear();
        EditDept.Clear();
        tEmp.Clear();
    }

    private bool TmaxConnect(){
        tpstart_t  tpinfo = new tpstart_t();
        tpinfo.cltname = "star";
        tpinfo.usrname = "star";
        tpinfo.usrpwd  = "star";
        tpinfo.dompwd  = "star";
        tpinfo.flags   = atmi.TPUNSOL_IGN;

        nRcvBuf = atmi.TPALLOC("TPSTART", null, 0);
        if( nRcvBuf == 0 )
            return false;

        atmi.FilltpstartBuf(ref nRcvBuf, tpinfo);

        if( atmi.TPSTART(nRcvBuf) == -1 )
            return false;

        return true;
    }
    private bool CheckInputData(InputCheck Level) {
        if(EditEmpNo.Text.Length == 0)
            return false;
        if(Level == InputCheck.Pri_Key )
            return true;
        else {
            if(EditName.Text.Length == 0)
                return false;
            if(EditJob.Text.Length == 0)
                return false;
            if(EditMgr.Text.Length == 0)
                return false;
            if(EditDate.Text.Length != 8)
                return false;
            if(EditSal.Text.Length == 0)
                return false;
            if(EditComm.Text.Length == 0)
                return false;
            if(EditDept.Text.Length == 0)
                return false;
            return true;
        }
    }

```

```

    }

    private bool PutMyData(InputCheck Level) {
        int nData;
        float fData;

        if(EditEmpNo.Text.Length != 0) {
            nData = int.Parse(EditEmpNo.Text);
            if( fdl.FBPUT(nSndBuf,fdl.FBGET_FLDKEY(EmpFKey[0]),
                        ref nData,0)< 0)
                return false;
        }
        if(Level == InputCheck.Pri_Key )
            return true;

        if(EditName.Text.Length != 0)
            if(fdl.FBPUT(nSndBuf, fdl.FBGET_FLDKEY(EmpFKey[1]),
                        EditName.Text, EditName.Text.Length) < 0)
                return false;
        if(EditJob.Text.Length != 0)
            if(fdl.FBPUT( nSndBuf, fdl.FBGET_FLDKEY(EmpFKey[2]),
                        EditJob.Text, EditJob.Text.Length) < 0 )
                return false;

        if(EditMgr.Text.Length != 0) {
            nData = int.Parse(EditMgr.Text);
            if(fdl.FBPUT(nSndBuf,fdl.FBGET_FLDKEY(EmpFKey[3]),
                        ref nData,0)< 0 )
                return false;
        }
        if(EditDate.Text.Length == 8)
            if(fdl.FBPUT( nSndBuf, fdl.FBGET_FLDKEY(EmpFKey[4]),
                        EditDate.Text, EditDate.Text.Length) < 0 )
                return false;
        if(EditSal.Text.Length != 0){
            fData = float.Parse(EditSal.Text);
            if(fdl.FBPUT(nSndBuf,fdl.FBGET_FLDKEY(EmpFKey[5]),
                        ref fData,0)< 0 )
                return false;
        }
        if(EditComm.Text.Length != 0) {
            fData = float.Parse(EditComm.Text);
            if(fdl.FBPUT(nSndBuf,fdl.FBGET_FLDKEY(EmpFKey[6]),
                        ref fData,0)< 0 )
                return false;
        }
        if(EditDept.Text.Length != 0) {

```

```

        nData = int.Parse(EditDept.Text);
        if(fdl.FBPUT(nSndBuf,fdl.FBGET_FLDKEY(EmpFKey[7]),
                    ref nData,0)< 0)
            return false;
    }
    return true;
}

private void MakeDataSet() {
    ResultData = new DataSet("dataGrid");

    tEmp = new DataTable("Employee");
    cEmpNo = new DataColumn(ColumnNames[0],typeof(int));
    cENAME = new DataColumn(ColumnNames[1],typeof(string));
    cJob = new DataColumn(ColumnNames[2],typeof(string));
    cMgr = new DataColumn(ColumnNames[3],typeof(int));
    cDate = new DataColumn(ColumnNames[4],typeof(string));
    cSal = new DataColumn(ColumnNames[5],typeof(float));
    cComm = new DataColumn(ColumnNames[6],typeof(float));
    cDept = new DataColumn(ColumnNames[7],typeof(int));

    tEmp.Columns.Add(cEmpNo);
    tEmp.Columns.Add(cENAME);
    tEmp.Columns.Add(cJob);
    tEmp.Columns.Add(cMgr);
    tEmp.Columns.Add(cDate);
    tEmp.Columns.Add(cSal);
    tEmp.Columns.Add(cComm);
    tEmp.Columns.Add(cDept);

    ResultData.Tables.Add(tEmp);

    //DataRow newRow1 = tEmp.NewRow();
    dataGrid.SetDataBinding(ResultData, "Employee");
}

private void ReceiveError() {
    int nLeng = 0;
    object[] oEData = new object[ErrFLD_NUM];

    for(int i = 0; i < ErrFLD_NUM; i++) {
        if(fdl.FBGET(nRcvBuf, fdl.FBGET_FLDKEY(ErrFKey[i]),
                    out oEData[i], ref nLeng) < 0 )
            break;
    }
    ErrorProcess(oEData[2].ToString());
    return;
}

```

```

    }
    protected override void Dispose( bool disposing ) {
        if(disposing) {
            if(components != null) {
                components.Dispose();
            }
        }
        base.Dispose( disposing );
    }

    #region Windows Form Designer generated code
    /*デザイン設計*/

    ...
    #endregion

    static void Main() {
        Application.Run(new EmployeeMgr());
    }

    private void EmployeeMgr_Load(object sender, System.EventArgs e) {
        const string EnvFile = "D:\\tmax.env";

        if(atmi.TMAXREADENV(EnvFile,"TMAX") == -1 )
            ErrorProcess("環境ファイルの読み込みに失敗、-"+ EnvFile);
    }

    private void BtnExit_Click(object sender, System.EventArgs e) {
        Dispose();
    }

    private void BtnIns_Click(object sender, System.EventArgs e) {
        int nLeng = 0;

        if(CheckInputData( InputCheck.All ) == false) {
            ErrorProcess("入力していないスペースがあります。\\n
                           入社日は8桁(YYYYMMDD)で入力してください。");
            return;
        }

        bConnect = TmaxConnect();
        if(bConnect == false )
            ErrorProcess("サーバーに接続できません。");

        if(bTxRun == false) {
            if(tx.TX_BEGIN() < 0) {
                ErrorProcess( "TX_BEGIN Error");
                return;
            }
        }
    }

```

```

        }
        bTxRun = true;
    }

    if(nSndBuf != 0 )
        atmi.TPFREE(ref nSndBuf);
    if((nSndBuf = fdl.FBALLOC( 5, 1000) ) == 0 )
        ErrorProcess("FBALLOC - メモリー割り当てエラー");
    if(nRcvBuf != 0 )
        atmi.TPFREE(ref nRcvBuf);
    if((nRcvBuf = fdl.FBALLOC( 5, 1000) ) == 0 )
        ErrorProcess("FBALLOC - メモリー割り当てエラー");
    if(PutMyData( InputCheck.All ) == false) {
        ErrorProcess("フィールド・キーの挿入中にエラーが発生しました。");
        return;
    }
    if(atmi.TPCALL( "FDLINSERT", nSndBuf, 0, ref nRcvBuf,
        ref nLeng, atmi.TPNOFLAGS) < 0 ) {
        ReceiveError();
        return;
    }
    SuccessProcess( EditEmpNo.Text + "社員番号が挿入されました。");
}

private void BtnSel_Click(object sender, System.EventArgs e) {
    int nLeng = 0;
    object[] oData = new object[FIELD_NUM];

    if(CheckInputData( InputCheck.Pri_Key ) == false ) {
        ErrorProcess("社員番号を入力してください。");
        return;
    }

    bConnect = TmaxConnect();
    if(bConnect == false )
        ErrorProcess("サーバーに接続できません。");

    if(bTxRun == false ) {
        if(tx.TX_BEGIN() < 0 ) {
            ErrorProcess( "TX_BEGIN Error");
            return;
        }
        bTxRun = true;
    }

    if(nSndBuf != 0 )
        atmi.TPFREE(ref nSndBuf);

```

```

        if((nSndBuf = fdl.FBALLOC( 5, 1000) ) == 0 )
            ErrorProcess("FBALLOC - メモリー割り当てエラー");
        if(nRcvBuf != 0)
            atmi.TPFREE(ref nRcvBuf);
        if((nRcvBuf = fdl.FBALLOC( 5, 1000)) == 0)
            ErrorProcess("FBALLOC - メモリー割り当てエラー");
        if ( PutMyData( InputCheck.Pri_Key ) == false) {
            ErrorProcess("フィールド・キーの挿入中にエラーが発生しました。");
            return;
        }

        if(atmi.TPCALL("FDLSELECT", nSndBuf, 0, ref nRcvBuf,
            ref nLeng, atmi.TPNOFLAGS) < 0 ) {
            ReceiveError();
            return;
        }

        int OccrN = fdl.FBKEYOCCUR(nRcvBuf, fdl.FBGET_FLDKEY(EmpFKey[0]));
        for(int j=0; j< OccrNum; j++) {
            DataRow OutputRow = tEmp.NewRow();
            for(int i=0; i< FIELD_NUM; i++) {
                if(fdl.FBGET_TU( nRcvBuf, fdl.FBGET_FLDKEY(EmpFKey[i]),

                    j, out oData[i], ref nLeng) < 0 )
                    break;
                OutputRow[ColumnName[i]] = oData[i];
            }
            tEmp.Rows.Add(OutputRow);
        }

        SuccessProcess("照会内容です。");
        dataGrid.Visible = true;
        BtnBack.Visible = true;
        dataGrid.RowHeadersVisible = false;
    }

    private void BtnUpt_Click(object sender, System.EventArgs e) {

        int nLeng=0;
        if(CheckInputData( InputCheck.Pri_Key ) == false ) {
            ErrorProcess("入力していないスペースがあります。\\n
                入社日は8桁(YYYYMMDD)で入力してください。");
            return;
        }

        bConnect = TmaxConnect();
        if(bConnect == false )

```

```

        ErrorProcess("サーバーに接続できません。");

    if( bTxRun == false ) {
        if(tx.TX_BEGIN() < 0 ) {
            ErrorProcess( "TX_BEGIN Error");
            return;
        }
        bTxRun = true;
    }

    if(nSndBuf != 0 )
        atmi.TPFREE(ref nSndBuf);
    if((nSndBuf = fdl.FBALLOC( 5, 1000) ) == 0 )
        ErrorProcess("FBALLOC - メモリー割り当てエラー");
    if(nRcvBuf != 0 )
        atmi.TPFREE(ref nRcvBuf);
    if((nRcvBuf = fdl.FBALLOC( 5, 1000) ) == 0 )
        ErrorProcess("FBALLOC - メモリー割り当てエラー");

    if(PutMyData( InputCheck.All ) == false) {
        ErrorProcess("フィールド・キーの挿入中にエラーが発生しました。");
        return;
    }

    if(atmi.TPCALL("FDLUPDATE", nSndBuf, 0, ref nRcvBuf,
        ref nLeng, atmi.TPNOFLAGS) < 0 ) {
        ReceiveError();
        return;
    }

    SuccessProcess(EditEmpNo.Text + "社員番号が修正されました。");
    ClearWindow();
}

private void BtnDel_Click(object sender, System.EventArgs e) {
    int nLeng = 0;

    if( CheckInputData( InputCheck.Pri_Key ) == false ) {
        ErrorProcess("社員番号を入力してください。");
        return;
    }

    bConnect = TmaxConnect();
    if( bConnect == false )
        ErrorProcess("サーバーに接続できません。");

    if( bTxRun == false ) {

```

```

        if( tx.TX_BEGIN() < 0 ) {
            ErrorProcess( "TX_BEGIN Error");
            return;
        }
        bTxRun = true;
    }
    if( nSndBuf != 0 )
        atmi.TPFREE(ref nSndBuf);
    if( ( nSndBuf = fdl.FBALLOC( 5, 1000) ) == 0 )
        ErrorProcess("FBALLOC - メモリー割り当てエラー");
    if( nRcvBuf != 0 )
        atmi.TPFREE(ref nRcvBuf);
    if( ( nRcvBuf = fdl.FBALLOC( 5, 1000) ) == 0 )
        ErrorProcess("FBALLOC - メモリー割り当てエラー");

    if( PutMyData( InputCheck.Pri_Key ) == false ) {
        ErrorProcess("フィールド・キーの挿入中にエラーが発生しました。");
        return;
    }

    if(atmi.TPCALL("FDLDELETE", nSndBuf, 0, ref nRcvBuf,
        ref nLeng, atmi.TPNOFLAGS) < 0) {
        ReceiveError();
        return;
    }

    SuccessProcess( EditEmpNo.Text + "社員番号が削除されました。");
    ClearWindow();
}

private void BtnBack_Click(object sender, System.EventArgs e) {
    dataGrid.Visible = false;
    BtnBack.Visible = false;
    ClearWindow();
}
}

```

下記は、UTF8で設定したUnicodeサポートの社員管理プログラム・デザインソースの例です。

通常、Atmi.SetServerEncoding(Encoding.UTF8)のみの設定で問題ありませんが、クライアント側で別途のエンコーディング方式を使用している場合は、SetClientEncoding()を利用して設定します。また、64bitをサポートするため、nSndBuf、nRcvBufがintからIntPtrに変更されました。

<Form1.cs>

```

using System;
using System.Drawing;

```



```

using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;

using AtmiNS;
using WinApiNS;
using FdlNS;
using TxNS;
using TmaxApiNS;

namespace _4GL_Sample
{
public class EmployeeMgr : System.Windows.Forms.Form
{
    Atmi.SetServerEncoding(Encoding.UTF8);
    private Atmi atmi = new Atmi();
    private WinApi winApi = new WinApi();
    private Fdl fdl = new Fdl();
    private Tx tx = new Tx();
    private TmaxApi tmaxApi = new TmaxApi();

    bool bTxRun, bConnect;
    IntPtr nSndBuf, nRcvBuf;
    public string[] EmpFKey =
    {"EMPNO", "ENAME", "JOB", "MGR", "DATE", "SAL", "COMM", "DEPTNO"};
    public string[] ErrFKey = {"E_TYPE", "E_CODE", "E_MSG", "E_TMP"};
    public string[] ColumnName =
    {"社員番号", "名前", "役職", "担当役員", "入社日", "給料", "COMM", "部署名"};

    public const int FIELD_NUM = 8;
    public const int ErrFLD_NUM = 4;
    public enum InputCheck { Pri_Key, All };

    private System.Windows.Forms.Label label1;
    private System.Windows.Forms.Label label2;
    private System.Windows.Forms.Label label3;
    private System.Windows.Forms.Label label4;
    private System.Windows.Forms.Label label5;
    private System.Windows.Forms.Label label6;
    private System.Windows.Forms.Label label7;
    private System.Windows.Forms.Label label8;
    private System.Windows.Forms.GroupBox groupBox1;
    private System.Windows.Forms.TextBox EditEmpNo;
    private System.Windows.Forms.TextBox EditName;
    private System.Windows.Forms.TextBox EditJob;
    private System.Windows.Forms.TextBox EditMgr;

```

```

private System.Windows.Forms.TextBox EditDate;
private System.Windows.Forms.TextBox EditSal;
private System.Windows.Forms.TextBox EditComm;
private System.Windows.Forms.TextBox EditDept;
private System.Windows.Forms.Label LabelErr;
private System.Windows.Forms.Button BtnSel;
private System.Windows.Forms.Button BtnUpt;
private System.Windows.Forms.Button BtnDel;
private System.Windows.Forms.Button BtnIns;
private System.Windows.Forms.Button BtnExit;
private System.Windows.Forms.Button BtnBack;
private System.Windows.Forms.DataGrid dataGrid;

private DataSet ResultData;
private DataTable tEmp;
private DataColumn cEmpNo;
private DataColumn cENAME;
private DataColumn cJob;
private DataColumn cMgr;
private DataColumn cDate;
private DataColumn cSal;
private DataColumn cComm;
private DataColumn cDept;
private System.ComponentModel.Container components = null;

public EmployeeMgr()
{
    bTxRun = false;
    bConnect = false;
    nSndBuf = IntPtr.Zero;
    nRcvBuf = IntPtr.Zero;
    InitializeComponent();
    MakeDataSet();
}

private void ErrorProcess(string ErrMsg)
{
    LabelErr.Text = ErrMsg;
    if( bTxRun == true )
    {
        tx.TX_ROLLBACK();
        bTxRun = false;
    }
    if( nRcvBuf != IntPtr.Zero )
        atmi.TPFFREE(ref nRcvBuf);
    if( nSndBuf != IntPtr.Zero )

```

```

        atmi.TPFFREE(ref nSndBuf);

        if( bConnect == true )
        {
            atmi.TPEND();
            bConnect = false;
        }
    }

private void SuccessProcess(string SucMsg)
{
    LabelErr.Text = SucMsg;
    if( bTxRun == true ) {
        tx.TX_COMMIT();
        bTxRun = false;
    }
    if( nRcvBuf != IntPtr.Zero )
        atmi.TPFFREE(ref nRcvBuf);
    if( nSndBuf != IntPtr.Zero )
        atmi.TPFFREE(ref nSndBuf);
    if( bConnect == true ) {
        atmi.TPEND();
        bConnect = false;
    }
}

private void ClearWindow() {
    EditEmpNo.Clear();
    EditName.Clear();
    EditJob.Clear();
    EditMgr.Clear();
    EditDate.Clear();
    EditSal.Clear();
    EditComm.Clear();
    EditDept.Clear();
    tEmp.Clear();
}

private bool TmaxConnect(){
    tpstart_t  tpinfo = new tpstart_t();
    tpinfo.cltname = "star";
    tpinfo.usrname = "star";
    tpinfo.usrpwd  = "star";
    tpinfo.dompwd  = "star";
    tpinfo.flags   = atmi.TPUNSOL_IGN;

    nRcvBuf = atmi.TPALLOC("TPSTART", null, 0);

```

```

        if( nRcvBuf == IntPtr.Zero)
            return false;

        atmi.FilltpstartBuf(ref nRcvBuf, tpinfo);

        if( atmi.TPSTART(nRcvBuf) == -1 )
            return false;

        return true;
    }
    private bool CheckInputData(InputCheck Level) {
        if(EditEmpNo.Text.Length == 0)
            return false;
        if(Level == InputCheck.Pri_Key )
            return true;
        else {
            if(EditName.Text.Length == 0)
                return false;
            if(EditJob.Text.Length == 0)
                return false;
            if(EditMgr.Text.Length == 0)
                return false;
            if(EditDate.Text.Length != 8)
                return false;
            if(EditSal.Text.Length == 0)
                return false;
            if(EditComm.Text.Length == 0)
                return false;
            if(EditDept.Text.Length == 0)
                return false;
            return true;
        }
    }

    private bool PutMyData(InputCheck Level) {
        int nData;
        float fData;

        if(EditEmpNo.Text.Length != 0) {
            nData = int.Parse(EditEmpNo.Text);
            if( fdl.FBPUT(nSndBuf,fdl.FBGET_FLDKEY(EmpFKey[0]),
                        ref nData,0)< 0)
                return false;
        }
        if(Level == InputCheck.Pri_Key )
            return true;
    }

```

```

        if(EditName.Text.Length != 0)
            if(fdl.FBPUT(nSndBuf, fdl.FBGET_FLDKEY(EmpFKey[1]),
                EditName.Text, EditName.Text.Length) < 0)
                return false;
        if(EditJob.Text.Length != 0)
            if(fdl.FBPUT( nSndBuf, fdl.FBGET_FLDKEY(EmpFKey[2]),
                EditJob.Text, EditJob.Text.Length) < 0 )

                return false;
        if(EditMgr.Text.Length != 0) {
            nData = int.Parse(EditMgr.Text);
            if(fdl.FBPUT(nSndBuf,fdl.FBGET_FLDKEY(EmpFKey[3]),
                ref nData,0)< 0 )
                return false;
        }
        if(EditDate.Text.Length == 8)
            if(fdl.FBPUT( nSndBuf, fdl.FBGET_FLDKEY(EmpFKey[4]),
                EditDate.Text, EditDate.Text.Length) < 0 )
                return false;
        if(EditSal.Text.Length != 0){
            fData = float.Parse(EditSal.Text);
            if(fdl.FBPUT(nSndBuf,fdl.FBGET_FLDKEY(EmpFKey[5]),
                ref fData,0)< 0 )
                return false;
        }
        if(EditComm.Text.Length != 0) {
            fData = float.Parse(EditComm.Text);
            if(fdl.FBPUT(nSndBuf,fdl.FBGET_FLDKEY(EmpFKey[6]),
                ref fData,0)< 0 )
                return false;
        }
        if(EditDept.Text.Length != 0) {
            nData = int.Parse(EditDept.Text);
            if(fdl.FBPUT(nSndBuf,fdl.FBGET_FLDKEY(EmpFKey[7]),
                ref nData,0)< 0)
                return false;
        }
        return true;
    }

    private void MakeDataSet() {
        ResultData = new DataSet("dataGrid");

        tEmp = new DataTable("Employee");
        cEmpNo = new DataColumn(ColumnNames[0],typeof(int));
        cENAME = new DataColumn(ColumnNames[1],typeof(string));
        cJob = new DataColumn(ColumnNames[2],typeof(string));
    }

```

```

        cMgr = new DataColumn(ColumnName[3],typeof(int));
        cDate = new DataColumn(ColumnName[4],typeof(string));
        cSal = new DataColumn(ColumnName[5],typeof(float));
        cComm = new DataColumn(ColumnName[6],typeof(float));
        cDept = new DataColumn(ColumnName[7],typeof(int));

        tEmp.Columns.Add(cEmpNo);
        tEmp.Columns.Add(cENAME);
        tEmp.Columns.Add(cJob);
        tEmp.Columns.Add(cMgr);
        tEmp.Columns.Add(cDate);
        tEmp.Columns.Add(cSal);
        tEmp.Columns.Add(cComm);
        tEmp.Columns.Add(cDept);

        ResultData.Tables.Add(tEmp);

        //DataRow newRow1 = tEmp.NewRow();
        dataGrid.SetDataBinding(ResultData, "Employee");
    }

    private void ReceiveError() {
        int nLeng = 0;
        object[] oEData = new object[ErrFLD_NUM];

        for(int i = 0; i < ErrFLD_NUM; i++) {
            if(fdl.FBGET(nRcvBuf, fdl.FBGET_FLDKEY(ErrFKey[i]),
                        out oEData[i], ref nLeng) < 0 )
                break;
        }
        ErrorProcess(oEData[2].ToString());
        return;
    }

    protected override void Dispose( bool disposing ) {
        if(disposing) {
            if(components != null) {
                components.Dispose();
            }
        }
        base.Dispose( disposing );
    }
}

#region Windows Form Designer generated code
/*デザイン設計*/
...
#endregion

```

```

static void Main() {
    Application.Run(new EmployeeMgr());
}

private void EmployeeMgr_Load(object sender, System.EventArgs e) {
    const string EnvFile = "D:\\tmax.env";

    if(atmi.TMAXREADENV(EnvFile,"TMAX") == -1 )
        ErrorProcess("環境ファイルの読み込みに失敗、-"+ EnvFile);
}

private void BtnExit_Click(object sender, System.EventArgs e) {
    Dispose();
}

private void BtnIns_Click(object sender, System.EventArgs e) {
    int nLeng = 0;

    if(CheckInputData( InputCheck.All ) == false) {
        ErrorProcess("入力していないスペースがあります。\\n
        入社日は8桁(YYYYMMDD)で入力してください。");
        return;
    }

    bConnect = TmaxConnect();
    if(bConnect == false )
        ErrorProcess("サーバーに接続できません。");

    if(bTxRun == false) {
        if(tx.TX_BEGIN() < 0) {
            ErrorProcess( "TX_BEGIN Error");
            return;
        }
        bTxRun = true;
    }

    if(nSndBuf != IntPtr.Zero )
        atmi.TPFREE(ref nSndBuf);
    if((nSndBuf = fdl.FBALLOC( 5, 1000) ) == IntPtr.Zero )
        ErrorProcess("FBALLOC - メモリー割り当てエラー");
    if(nRcvBuf != IntPtr.Zero )
        atmi.TPFREE(ref nRcvBuf);
    if((nRcvBuf = fdl.FBALLOC( 5, 1000) ) == IntPtr.Zero )
        ErrorProcess("FBALLOC - メモリー割り当てエラー");
    if(PutMyData( InputCheck.All ) == false) {
        ErrorProcess("フィールド・キーの挿入中にエラーが発生しました。");
        return;
    }
}

```

```

    }
    if(atmi.TPCALL( "FDLINSERT", nSndBuf, 0, ref nRcvBuf,
        ref nLeng, atmi.TPNOFLAGS) < 0 ) {
        ReceiveError();
        return;
    }
    SuccessProcess( EditEmpNo.Text + "社員番号が挿入されました。");
}

private void BtnSel_Click(object sender, System.EventArgs e) {
    int nLeng = 0;
    object[] oData = new object[FIELD_NUM];

    if(CheckInputData( InputCheck.Pri_Key ) == false ) {
        ErrorProcess("社員番号を入力してください。");
        return;
    }

    bConnect = TmaxConnect();
    if(bConnect == false )
        ErrorProcess("サーバーに接続できません。");

    if(bTxRun == false ) {
        if(tx.TX_BEGIN() < 0 ) {
            ErrorProcess( "TX_BEGIN Error");
            return;
        }
        bTxRun = true;
    }

    if(nSndBuf != IntPtr.Zero )
        atmi.TPFREE(ref nSndBuf);
    if((nSndBuf = fdl.FBALLOC( 5, 1000 ) ) == IntPtr.Zero )
        ErrorProcess("FBALLOC - メモリー割り当てエラー");
    if(nRcvBuf != IntPtr.Zero)
        atmi.TPFREE(ref nRcvBuf);
    if((nRcvBuf = fdl.FBALLOC( 5, 1000)) == IntPtr.Zero)
        ErrorProcess("FBALLOC - メモリー割り当てエラー");
    if ( PutMyData( InputCheck.Pri_Key ) == false) {
        ErrorProcess("フィールド・キーの挿入中にエラーが発生しました。");
        return;
    }

    if(atmi.TPCALL("FDLSELECT", nSndBuf, 0, ref nRcvBuf,
        ref nLeng, atmi.TPNOFLAGS) < 0 ) {
        ReceiveError();
        return;
    }

```



```

    }

    int OccrN = fdl.FBKEYOCCUR(nRcvBuf, fdl.FBGET_FLDKEY(EmpFKey[0]));
    for(int j=0; j< OccrNum; j++) {
        DataRow OutputRow = tEmp.NewRow();
        for(int i=0; i< FIELD_NUM; i++) {
            if(fdl.FBGET_TU( nRcvBuf, fdl.FBGET_FLDKEY(EmpFKey[i]),

                           j, out oData[i], ref nLeng) < 0 )
                break;
            OutputRow[ColumnName[i]] = oData[i];
        }
        tEmp.Rows.Add(OutputRow);
    }

    SuccessProcess("照会内容です。");
    dataGrid.Visible = true;
    BtnBack.Visible = true;
    dataGrid.RowHeadersVisible = false;
}

private void BtnUpt_Click(object sender, System.EventArgs e) {

    int nLeng=0;
    if(CheckInputData( InputCheck.Pri_Key ) == false ) {
        ErrorProcess("入力していないスペースがあります。 \n
                     入社日は8桁(YYYYMMDD)で入力してください。");
        return;
    }

    bConnect = TmaxConnect();
    if(bConnect == false )
        ErrorProcess("サーバーに接続できません。");

    if( bTxRun == false ) {
        if(tx.TX_BEGIN() < 0 ) {
            ErrorProcess( "TX_BEGIN Error");
            return;
        }
        bTxRun = true;
    }

    if(nSndBuf != IntPtr.Zero )
        atmi.TPFREE(ref nSndBuf);
    if((nSndBuf = fdl.FBALLOC( 5, 1000) ) == IntPtr.Zero )
        ErrorProcess("FBALLOC - メモリー割り当てエラー");
    if(nRcvBuf != IntPtr.Zero )

```

```

        atmi.TPFFREE(ref nRcvBuf);
    if((nRcvBuf = fdl.FBALLOC( 5, 1000) ) == IntPtr.Zero )
        ErrorProcess("FBALLOC - メモリー割り当てエラー");

    if(PutMyData( InputCheck.All ) == false) {
        ErrorProcess("フィールド・キーの挿入中にエラーが発生しました。");
        return;
    }

    if(atmi.TPCALL("FDLUPDATE", nSndBuf, 0, ref nRcvBuf,
        ref nLeng, atmi.TPNOFLAGS) < 0 ) {
        ReceiveError();
        return;
    }

    SuccessProcess(EditEmpNo.Text + "社員番号が修正されました。");
    ClearWindow();
}

private void BtnDel_Click(object sender, System.EventArgs e) {
    int nLeng = 0;

    if( CheckInputData( InputCheck.Pri_Key ) == false ) {
        ErrorProcess("社員番号を入力してください。");
        return;
    }

    bConnect = TmaxConnect();
    if( bConnect == false )
        ErrorProcess("サーバーに接続できません。");

    if( bTxRun == false ) {
        if( tx.TX_BEGIN() < 0 ) {
            ErrorProcess( "TX_BEGIN Error");
            return;
        }
        bTxRun = true;
    }

    if( nSndBuf != IntPtr.Zero )
        atmi.TPFFREE(ref nSndBuf);
    if( ( nSndBuf = fdl.FBALLOC( 5, 1000) ) == IntPtr.Zero )
        ErrorProcess("FBALLOC - メモリー割り当てエラー");
    if( nRcvBuf != IntPtr.Zero )
        atmi.TPFFREE(ref nRcvBuf);
    if( ( nRcvBuf = fdl.FBALLOC( 5, 1000) ) == IntPtr.Zero )
        ErrorProcess("FBALLOC - メモリー割り当てエラー");
}

```

```

        if( PutMyData( InputCheck.Pri_Key ) == false ) {
            ErrorProcess("フィールド・キーの挿入中にエラーが発生しました。");
            return;
        }

        if(atmi.TPCALL("FDLDELETE", nSndBuf, 0, ref nRcvBuf,
            ref nLeng, atmi.TPNOFLAGS) < 0) {
            ReceiveError();
            return;
        }

        SuccessProcess( EditEmpNo.Text + "社員番号が削除されました。");
        ClearWindow();
    }

private void BtnBack_Click(object sender, System.EventArgs e) {
    dataGrid.Visible = false;
    BtnBack.Visible = false;
    ClearWindow();
}
}

```

6.2.5. サーバー・プログラム

サービス・プログラム

下記は、サービス・プログラムの例です。

<emp_c.pc>

```

#include <stdio.h>
#include <ctype.h>
#include <tuxinc/macro.h>
#include "../fdl/demo_fdl.h"

EXEC SQL include sqlca.h;
EXEC SQL INCLUDE ORACA;
EXEC ORACLE OPTION (ORACA=YES);
EXEC ORACLE OPTION (RELEASE_CURSOR=YES);

#define INP    1
#define ORA    2
#define TMX    3
#define APP    4

EXEC SQL begin declare section;

```

```

int h_empno;
char h_ename[11];
char h_job[10];
int h_mgr;
char h_date[11];
float h_sal;
float h_comm;
int h_deptno;
EXEC SQL end declare section;

void svc_error(long type, long err_code, char *msg, long tmp);

FDLINSERT( TPSVCINFO *msg ) {
    FBUF *rcvbuf;
    int i, occurrence;
    rcvbuf = (FBUF *)msg->data;
    h_empno = h_mgr = h_sal = h_comm = h_deptno = 0;

    memset( h_ename, 0x00, sizeof( h_ename ) );
    memset( h_job, 0x00, sizeof( h_job ) );
    memset( h_date, 0x00, sizeof( h_date ) );

    occurrence = fbkeyoccur(rcvbuf, EMPNO);

    for (i=0; i< occurrence; i++){
        fbget_tu (rcvbuf, EMPNO, i, (char *)&h_empno, 0);
        fbget_tu (rcvbuf, MGR, i, (char *)&h_mgr, 0);
        fbget_tu (rcvbuf, SAL, i, (char *)&h_sal, 0);
        fbget_tu (rcvbuf, COMM, i, (char *)&h_comm, 0);
        fbget_tu (rcvbuf, DEPTNO,i, (char *)&h_deptno, 0);
        fbget_tu (rcvbuf, ENAME, i, (char *)h_ename, 0);
        fbget_tu (rcvbuf, JOB , i, (char *)h_job, 0);
        fbget_tu (rcvbuf, DATE , i, (char *)h_date, 0);

        EXEC SQL INSERT
        INTO emp(empno, ename, job, mgr, hiredate, sal,comm, deptno)
        VALUES (:h_empno, :h_ename, :h_job, :h_mgr,
                to_date(:h_date,'yyyymmdd'), :h_sal, :h_comm, :h_deptno);
    }

    if(sqlca.sqlcode != 0)
        goto LB_DBERROR;

    EXEC SQL WHENEVER SQLERROR
        goto LB_DBERROR;

    tpreturn(TPSUCCESS, 0L, (char *)rcvbuf, 0L, 0L);

    LB_DBERROR :
        EXEC SQL WHENEVER SQLERROR CONTINUE;
        svc_error(ORA, sqlca.sqlcode, sqlca.sqlerrm.sqlerrmc, 0) ;
}

```

```

}

FDLDELETE( TPSVCINFO *msg )
{
    FBUF *rcvbuf;
    int i, occurrence;

    rcvbuf = ( FBUF *)msg->data;
    occurrence = fbkeyoccur(rcvbuf, EMPNO);

    for (i=0; i< occurrence; i++){
        fbget_tu (rcvbuf, EMPNO, i, (char *)&h_empno , 0);

        EXEC SQL DELETE
        FROM emp
        WHERE empno = :h_empno;
    }

    if(sqlca.sqlcode != 0)
        goto LB_DBERROR;

    EXEC SQL WHENEVER SQLERROR
        goto LB_DBERROR;

    EXEC SQL WHENEVER NOT FOUND
        goto LB_DBERROR;

    tpreturn(TPSUCCESS, 0L, (char *)rcvbuf, 0L, 0L);

    LB_DBERROR :
        EXEC SQL WHENEVER SQLERROR CONTINUE;
        svc_error(ORA, sqlca.sqlcode, sqlca.sqlerrm.sqlerrmc, 0);
}

FDLSELECT( TPSVCINFO *msg )
{
    FBUF *rcvbuf;
    FLDLEN fldlen;
    int i=0, ROWMEM=200, CHKROW=500;

    rcvbuf = (FBUF *)msg->data;

    if((rcvbuf=(FBUF *)tprealloc((char *)rcvbuf,ROWMEM*CHKROW))==NULL){
        rcvbuf=(FBUF *)msg->data;
        goto LB_TMAXERROR ;
    }

    h_empno = h_mgr = h_sal = h_comm = h_deptno = 0;

    memset( h_ename, 0x00, sizeof( h_ename ) );

```

```

memset( h_job, 0x00, sizeof( h_job ) );
memset( h_date, 0x00, sizeof( h_date ) );

fbget_tu( rcvbuf, EMPNO, 0, (char *)&h_empno, &fldlen);

EXEC SQL DECLARE DB_CUR CURSOR FOR
SELECT  nvl(empno,0), nvl(ename,' '), nvl(job,' '),
        nvl(to_char(hiredate,'yyyymmdd'),' '), nvl(mgr,0),
        nvl(sal,0), nvl(comm,0), nvl(deptno,0)
FROM    emp
WHERE   empno >= :h_empno-50 AND empno <= :h_empno+50;
EXEC SQL OPEN DB_CUR;

if(sqlca.sqlcode != 0)
    goto LB_DBERROR;
EXEC SQL WHENEVER SQLERROR
    goto LB_DBERROR ;
EXEC SQL WHENEVER NOT FOUND
    Do break ;

while(1) {
    EXEC SQL FETCH DB_CUR
    INTO      :h_empno,
              :h_ename,
              :h_job,
              :h_date,
              :h_mgr,
              :h_sal,
              :h_comm,
              :h_deptno;

    fbchg_tu(rcvbuf, EMPNO, i, (char *)&h_empno, 0);
    fbchg_tu(rcvbuf, MGR,   i, (char *)&h_mgr, 0);
    fbchg_tu(rcvbuf, SAL,   i, (char *)&h_sal, 0);
    fbchg_tu(rcvbuf, DEPTNO, i, (char *)&h_deptno, 0);
    fbchg_tu(rcvbuf, COMM,  i, (char *)&h_comm, 0);
    fbchg_tu(rcvbuf, ENAME, i, (char *)&h_ename, 0);
    fbchg_tu(rcvbuf, JOB,   i, (char *)&h_job, 0);
    fbchg_tu(rcvbuf, DATE,  i, (char *)&h_date, 0);

    i++;
}

if (i < 1)
    goto LB_DBERROR;

EXEC SQL CLOSE DB_CUR;

tpreturn(TPSUCCESS, 0L, (char *)rcvbuf, 0L, 0L);

LB_DBERROR :

```

```

        EXEC SQL WHENEVER SQLERROR CONTINUE;
        EXEC SQL CLOSE DB_CUR ;
        svc_error(ORA, sqlca.sqlcode, sqlca.sqlerrm.sqlerrmc, 0L) ;

LB_TMAXERROR :
        EXEC SQL WHENEVER SQLERROR CONTINUE;
        EXEC SQL CLOSE DB_CUR ;
        svc_error(TMX, tperrno, "", 0L) ;
}

FDLUPDATE( TPSVCINFO *msg ) {
    FBUF *rcvbuf ;
    int i, occurrence;
    rcvbuf = (FBUF *)msg->data;
    h_empno = h_mgr = h_sal = h_comm = h_deptno = 0;

    memset( h_ename, 0x00, sizeof( h_ename ) );
    memset( h_job, 0x00, sizeof( h_job ) );
    memset( h_date, 0x00, sizeof( h_date ) );

    occurrence = fbkeyoccur(rcvbuf, EMPNO);

    for (i=0; i< occurrence; i++) {
        fbget_tu (rcvbuf, EMPNO, i, (char *)&h_empno, 0);
        fbget_tu (rcvbuf, ENAME, i, (char *)h_ename, 0);
        fbget_tu (rcvbuf, JOB, i, (char *)h_job, 0);
        fbget_tu (rcvbuf, MGR, i, (char *)&h_mgr, 0);
        fbget_tu (rcvbuf, SAL, i, (char *)&h_sal, 0);
        fbget_tu (rcvbuf, COMM, i, (char *)&h_comm, 0);
        fbget_tu (rcvbuf, DEPTNO, i, (char *)&h_deptno, 0);
        fbget_tu (rcvbuf, DATE, i, (char *)h_date, 0 );

        EXEC SQL UPDATE emp
        SET ename = nvl(:h_ename, ename),
            job   = nvl(:h_job, job),
            mgr   = :h_mgr,
            hiredate = nvl(to_date(:h_date, 'yyyymmdd'), hiredate),
            sal   = :h_sal,
            comm  = :h_comm,
            deptno = :h_deptno
        WHERE empno = :h_empno;

        if(sqlca.sqlcode != 0)
            goto LB_DBERROR;

        EXEC SQL WHENEVER SQLERROR
            goto LB_DBERROR;

        EXEC SQL WHENEVER NOT FOUND
            goto LB_DBERROR;
    }
}

```

```

    }
    tpreturn(TPSUCCESS, 0L, (char *)rcvbuf, 0L, 0L);

    LB_DBERROR :
        EXEC SQL WHENEVER SQLERROR CONTINUE;
        svc_error(ORA, sqlca.sqlcode, sqlca.sqlerrm.sqlerrmc, 0L) ;
}

/*****
 * エラー処理 : サービスでエラーの発生時にそのエラーをバッファーに入れてクライアントに転送します。
 *****/
void svc_error(long type, long err_code, char *msg, long tmp) {
    FBUF *transf;
    char *svcname;
    char err_msg[100];
    char temp[100];
    int i = 0;

    printf("type      ==>[%ld]\n", type);
    printf("err_code ==>[%ld]\n", err_code);
    printf("msg        ==>[%s]\n", msg);
    strcpy(err_msg, msg);

    if((transf = (FBFR *)tpalloc("FML", NULL, 0)) == NULL) {
        printf("tpalloc failed! errno = %d\n", tperrno);
    }

    switch(type) {
        case INP:
            switch(err_code) {
                case -1000:
                    strcpy(err_msg, "当該ユーザーに権限がありません。");
                    break;
                default:
                    strcpy(err_msg,
                        "Inputエラー・メッセージが登録されていません。");
            }
            break;
        case ORA:
            if(strlen(err_msg) == 0)
                strcpy(err_msg, sqlca.sqlerrm.sqlerrmc);
            break;
        case TMX:
            if(strlen(err_msg) == 0) strcpy(err_msg, tpstrerror(tperrno));
            break;
        case APP:
            if(strlen(err_msg) == 0) {
                /* err_mssg=""の場合はエラー・メッセージを設定します。 *****/
                switch(err_code) {
                    case -500: /* システム関連エラー */
                        strcpy(err_msg, "ファイル作成エラーです。");

```



```

        break;
    case -502:
        strcpy(err_msg, "内部サービスが呼び出せませんでした。");
        break;
    case -504:
        strcpy(err_msg, "ソケット通信エラーです。");
        break;
    case -505: /* 他のトランザクションで修正された場合はMSG処理*/
        /* "照会の後、他の場所で[%s]データが変更されました。
        \n\nを再照会した後処理してください。": クライアントで処理*/
        strcpy(err_msg, "がありません。");
        break;
    case -5002:
        strcpy(err_msg, "電算室にお問い合わせください。");
        break;
    default:
        strcpy(err_msg,
            "アプリケーション・エラー・メッセージが登録されていません。");
    }
}
break;
}

/* ROLLBACK          */
EXEC SQL WHENEVER SQLERROR CONTINUE;
EXEC SQL ROLLBACK;

fbchg_tu (transf, E_TYPE, i, (char *)&type, 0);
fbchg_tu (transf, E_CODE, i, (char *)&err_code, 0);
fbchg_tu (transf, E_MSG, i, (char *)&err_msg, 0);
fbchg_tu (transf, E_TMP, i, (char *)&tmp, 0);

tpreturn(TPFAIL, 0, (char *)transf, 0L, 0L);
}

```

メイクファイル(Makefile)

下記は、<emp_c.pc>ソースをTmaxアプリケーションに作成するメイクファイルの例です。

<emp_c.mk>

```

include $(ORACLE_HOME)/precomp/lib/env32.mk
ORALIBDIR = $(LIBHOME)
ORALIB = -L/home/oracle/OraHome/lib32/ -lclntsh -lld -lm
        `cat /home/oracle/OraHome/lib32/sysliblist` -lm -lc_r -lpthreads

TARGET = emp_c
APOBJS = emp_c.o
NSDLOBJ = $(TMAXDIR)/lib/sdl.o

```

```

#CC
CC=cc

#Oracle
LIBS      = -lsvr -loras
OBSJS     = $(APOBJS) $(SVCTOBJ)
SVCTOBJ   = $(TARGET)_svctab.o
CFLAGS    = -q32 -O -I$(TMAXDIR)
LDFLAGS   = -brtl

APPDIR    = $(TMAXDIR)/appbin
SVCTDIR   = $(TMAXDIR)/svct
TMAXLIBDIR = $(TMAXDIR)/lib
#
.SUFFIXES : .c
.c.o:
        $(CC) $(CFLAGS) $(LDFLAGS) -c $<
all: $(TARGET)

$(TARGET): $(OBSJS)
        $(CC) $(CFLAGS) $(LDFLAGS) -L$(TMAXLIBDIR) -o $(TARGET)
        -L$(ORALIBDIR)
        $(ORALIB) $(OBSJS) $(LIBS) $(NSDLOBJ)
        mv $(TARGET) $(TMAXDIR)/appbin

$(APOBJS): $(TARGET).pc
        proc iname=emp_c include=$(TMAXDIR) define=__LINUX_ORACLE_PROC__
        $(CC) $(CFLAGS) $(LDFLAGS) -c $(TARGET).c

$(SVCTOBJ):
        touch $(SVCTDIR)/$(TARGET)_svctab.c
        $(CC) $(CFLAGS) $(LDFLAGS) -c $(SVCTDIR)/$(TARGET)_svctab.c

#clean:
        -rm -f *.o core $(TARGET) $(TARGET).lis

```

第7章 ASPインターフェース

本章では、ASPインターフェースで使用する関数と例について説明します。

7.1. 概要

ASP(Active Server Pages)インターフェースは、Web環境で動的にWebページを作成してユーザーに提供するスクリプトです。ASPでTmaxのサービス呼び出すには、Tmaxで提供するtmaxcomcli.dllインターフェースを使用する必要があります。

tmaxcomcli.dllはインプロセス・サーバーで動作しており、下記のような4つのインターフェースがあります。

モジュール	説明
Tmaxcom.Atmi	基本的なTmax関数をラッピングしたメソッドを含みます
Tmaxcom.Etc	Tmaxcom.Atmiで提供していない関数をラッピングしたメソッドを含みます
Tmaxcom.Fdl	Tmaxフィールド・バッファーを処理する関数をラッピングしたメソッドを含みます
Tmaxcom.Tx	Tmaxトランザクション関数をラッピングしたメソッドを含みます

ASPでのTmaxサービスは、上記のインターフェースに対するオブジェクトを作成した後、該当するメソッドを呼び出すことで実行できます。詳細内容については、「[7.6. サンプル・プログラム](#)」を参照してください。

下記は、ASPインターフェースで使用する関数の一覧です。

● Atmiインターフェース・メソッド

関数	説明
OnGettperrno	Tmaxシステムを呼び出すとき、設定されたエラー値を返す関数です
OnGettpurcode	サービスでtpreturnする場合、設定したurcodeを返す関数です
OnTpacall	非同期型サービス要求を送信する関数です
OnTpalloc	バッファー・タイプ(Typed Buffer)割り当て関数です
OnTpcall	同期型サービス要求を送受信する関数です
OnTpcancel	OnTpacallが返した呼び出し記述子のcdで応答を取り消します
OnTpconnect	会話型サービスとの通信を接続する関数です
OnTpdiscon	会話型通信の接続を終了する関数です
OnTpend	Tmaxシステムとの接続を解除する関数です
OnTpfree	バッファー・タイプに割り当てられたメモリーを解除する関数です

関数	説明
OnTpget	OnTpallocなどで割り当てられたバッファのポインターが指し示す値を取得する関数です
OnTpgetrply	非同期的に要求したサービスの応答を受信する関数です
OnTpput	OnTpallocなどで割り当てられたバッファのポインターが指し示す場所にvalueを設定する関数です
OnTprealloc	バッファ・タイプ(Typed Buffer)を再割り当てする関数です
OnTprecv	会話型通信で接続された相手プログラムから送信されたデータを受信するために使用される関数です
OnTpsend	会話型通信を行うときにメッセージを送信する関数です
OnTpstart	Tmaxシステムに接続する関数です
OnTpypes	バッファ・タイプおよび下位タイプに関する情報を提供する関数です

● その他のインターフェース・メソッド

関数	説明
OnTmax_chk_conn	クライアントがTmaxシステムと接続されている状態なのかをチェックする関数です
OnTmaxreadenv	環境変数をファイルから読み込む関数です
OnTp_sleep	最大sec時間の間にスリープ(sleep)状態になり、その時間内にデータが受信されると即リターンする関数です
OnTp_usleep	最大usec時間の間にスリープ状態になり、その時間内にデータが受信されると即リターンする関数です
OnTpdeq	tpenq()を利用してサービスを要求した結果や、tpenq()を利用するときにサービス名をNULLにして保存したデータを読み込む関数です
OnTpenq	RQにデータを保存する関数です
OnTperrordetail	Tmaxシステムを呼び出すとき発生したエラーの詳細情報を取得するときに使用する関数です
OnTpextsvcinfo	OnTpdeq()を利用してRQからデータを読み込む場合、当該データの詳細情報を取得するときに使用する関数です
OnTpextsvcname	OnTpdeq()を利用してRQからデータを読み込む場合、当該データのサービス名を確認するために使用する関数です
OnTpgetenv	strという名前で登録された環境変数の値を返す関数です
OnTpputenv	strで入力された「名前=値」を環境変数に適用する関数です
OnTpqstat	現在のキューに保存されたデータの統計値を返す関数です
OnTpqsvcstat	現在のキューに保存されたデータのうち、特定のサービスに対する統計値を返す関数です

関数	説明
OnTpissue	OnTpdeq()を利用してRQでサービスを実行する際に、ネットワークの不安定やその他のサーバー・エラーによってサービスに失敗し、failキューに蓄積されたクライアントのサービス要求データを再びrequestキューに格納する関数です
OnTpreset	現在接続されているクライアントを即解除する関数です
OnTpset_timeout	サーバーに設定されているサービス制限時間、すなわちブロッキング・タイムアウト時間を変更するときに使用する関数です
OnTpsubqname	サブキュー番号に該当するキューの名前を返す関数です
OnTptobackup	クライアントがTmaxのバックアップ・システムに接続する関数です

● FDLインターフェース・メソッド

関数	説明
OnFballoc	フィールド・バッファのサイズを計算した後、メモリーを割り当てる関数です
OnFbcalcsz	OnTpalloc()を利用してフィールド・バッファをメモリーに割り当てるときに、適切なメモリーを割り当てるため、事前にフィールド・バッファ・サイズを計算する関数です
OnFbdelall	フィールド・バッファで指定したフィールド・キーの全データを削除する関数です
OnFbdelete	フィールド・バッファに指定したフィールド・キーのフィールド順に該当するデータを削除する関数です
OnFb fldcount	フィールド・バッファ内に存在するすべてのフィールド数を返す関数です
OnFbfree	フィールド・バッファのメモリーを解除する関数です
OnFbget	フィールド・バッファでフィールド・キーに該当するフィールド・データを返す関数です
OnFbget_fbsz	フィールド・バッファのサイズを返す関数です
OnFbget_fldkey	指定したフィールド名と一致するフィールド・キーを返す関数です
OnFbget_fldname	指定したフィールド・キーと一致するフィールド名を返す関数です
OnFbget_fldtype	指定したフィールド・キーのフィールド・タイプを整数で返す関数です
OnFbget_strfldtype	指定したフィールド・キーのフィールド・タイプを文字列で返す関数です
OnFbget_unused	メモリーに割り当てられたフィールド・バッファの中で、使用前のフィールド・バッファ・サイズを計算する関数です
OnFbget_used	メモリーに割り当てられたフィールド・バッファの中で、使用中のフィールド・バッファ・サイズを計算する関数です
OnFbgetc	フィールド・バッファに保存されているフィールド・データを指定したタイプに変換して順次リターンする関数です

関数	説明
OnFbgetf	フィールド・バッファに保存されているフィールド・データを順次リターンする関数です
OnFbgetnth	指定したフィールド・キーに保存されているフィールド・データの中で、指定したデータが保存されているフィールド数を返す関数です
OnFbgetntht	指定したフィールド・キーに保存されているフィールド・データの中で、指定したデータが保存されているフィールド順を返す関数です。
OnFbgetval	フィールド・バッファで指定したフィールド・キーのフィールド順に該当するフィールド・データを返す関数です
OnFbgetvali	指定したフィールド・キーのフィールド順に該当するフィールド・データを整数に変換して返す関数です
OnFbgetvals	指定したフィールド・キーのフィールド順に該当するフィールド・データを文字列に変換して返す関数です
OnFbgetvalt	指定したフィールド・キーのフィールド順に該当するフィールド・データを指定したタイプに変換して返す関数です
OnFbinit	フィールド・バッファを初期化する関数です
OnFbinsert	フィールド・バッファに指定したフィールド・キーのフィールド順にデータを保存する関数です
OnFbisfbuf	指定したフィールド・バッファが有効なのか(OnFballocまたはOnTpallocでバッファを割り当てられた変数なのか)を確認する関数です
OnFbispres	フィールド・バッファに指定したフィールド・キーのフィールド順にフィールド・データが存在するの否かを確認する関数です
OnFbkeynm_unload	メモリにローディングされたFDLファイルを新たにローディングする場合、この関数を呼び出すことができます
OnFbkeyoccur	フィールド・バッファで指定したフィールド・キーのフィールド順を返す関数です
OnFbmake_fldkey	新しいフィールド・キーを動的に作成する関数です
OnFbnmkey_unload	メモリにローディングされたFDLファイルを新たにローディングする場合、この関数を呼び出すことができます
OnFbput	フィールドバッファに新しいフィールドを追加する関数です
OnFbputt	データをフィールドバッファに指定したタイプに変換した後、新しいフィールドを追加して保存する関数です
OnFbrealloc	フィールドバッファをメモリに再割り当てする関数です
OnFbsterror	フィールド・バッファの操作時にエラー内容を文字列で返す関数です
OnFbtypecvt	指定したフィールド・データのタイプを指定したタイプに変換して返す関数です
OnFbupdate	フィールド・バッファに指定したフィールド・キーのフィールド順に該当するデータを変更する関数です

関数	説明
OnGetferrno	フィールド・バッファに関連したAPI関数の実行中にエラーが発生した場合、エラー値を返す関数です
OnGetferror	フィールド・バッファに関連したAPI関数の実行中にエラーが発生した場合、エラー値を返す関数です

• Txインターフェース・メソッド

関数	説明
OnTx_begin	グローバル・トランザクションを開始する関数です
OnTx_commit	グローバル・トランザクションをコミットする関数です
OnTx_rollback	グローバル・トランザクションをロールバックする関数です
OnTx_set_transaction_timeout	グローバル・トランザクションのタイムアウトを設定する関数です

参考

内部的に呼び出される各関数の詳細内容については、『Tmax FDL リファレンスガイド』または『Tmax リファレンスガイド』を参照してください。

7.2. Atmiインターフェース・メソッド

7.2.1. OnGettperrno

Tmaxシステムの呼び出し時に設定されたエラー値を返す関数です

• プロトタイプ

```
OnGettperrno()
```

• 戻り値

エラー値を返します。

• 関連関数

[OnTpurcode\(\)](#)、[OnTpsterror\(\)](#)

7.2.2. OnGettpurcode

サービスでtpreturnする場合、設定したurcodeを返す関数です。

- プロトタイプ

```
OnGettpurcode( )
```

- 戻り値

urcodeを返します。

- 関連関数

OnGettperrno(), OnTpsterror()

7.2.3. OnTpacall

非同期型サービス要求を送信する関数です。内部的にtpacallを呼び出します。

- プロトタイプ

```
OnTpacall(VARIANT svc, VARIANT data, VARIANT len, VARIANT flags)
```

- パラメータ

パラメータ	説明
svc	処理を要求するサービス名です
data	サービス要求データのポインターです。データがNULLでない場合は、OnTpallocまたはOnFballocで割り当てられたバッファのポインターである必要があります。
len	送信するデータ長を指定します
flags	tpcallを行うためのフラグです

- 戻り値

戻り値	説明
記述子(descriptor)	関数呼び出しに成功した場合です。記述子はOnTpgetrply()、OnTpcancel()などのメソッドでパラメータとして使用されます
-1	関数呼び出しに失敗した場合です。OnGettperrno()を呼び出してエラー値を取得します

- 関連関数

OnTpcall()、OnTpgetrply()、OnGettperrno()

7.2.4. OnTpalloc

バッファ・タイプ(Typed Buffer)割り当て関数です。内部的にtpallocを呼び出します。

- プロトタイプ

```
OnTpalloc(VARIANT type, VARIANT subtype, VARIANT size)
```

- パラメータ

パラメータ	説明
type	指定されたタイプを示します(FIELD/STRING etc.)
subtype	構造体バッファを使用する場合に使われるパラメータで、ASPでは使用されません
size	バッファ・サイズです

- 戻り値

戻り値	説明
適切なバッファ・タイプのポインター	関数呼び出しに成功した場合です
NULL	関数呼び出しに失敗した場合です。OnGettperrno()を呼び出してエラー値を取得します

- 関連関数

OnTprealloc()、OnTpfree()、OnGettperrno()

7.2.5. OnTpcall

同期型サービス要求を送受信する関数です。内部的にtpcallを呼び出します。

- プロトタイプ

```
OnTpcall(VARIANT svc, VARIANT idata, VARIANT ilen, VARIANT *odata,  
          VARIANT *olen, VARIANT flags)
```

- パラメータ

パラメータ	説明
svc	処理を要求するサービス名です
idata	サービス要求データのポインターです。データがNULLでない場合は、OnTpallocまたはOnFballocで割り当てられたバッファのポインターである必要があります。
ilen	送信するデータ長を指定します
odata	受信される応答バッファのポインターです。OnTpallocまたはOnFballocで割り当てられたバッファのポインターである必要があります
olen	受信したデータ長を取得します
flags	tpcallを行うためのフラグです

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です。OnGettperrno()を呼び出してエラー値を取得します

- 関連関数

OnTpalloc()、OnTpacall()、OnTpgetrply()、OnGettperrno()

7.2.6. OnTpcancel

OnTpacallが返した呼び出し記述子であるcdで応答を取り消す関数です。内部的にtpcancelを呼び出します。

- プロトタイプ

```
OnTpcancel(VARIANT cd)
```

- パラメータ

パラメータ	説明
cd	OnTpacallが返した記述子です

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です

戻り値	説明
-1	関数呼び出しに失敗した場合です。OnGettperrno()を呼び出してエラー値を取得します

- 関連関数

OnTpacall()、OnTpgetrply()、OnGettperrno()

7.2.7. OnTpconnect

会話型サービスとの通信を接続する関数です。内部的にtpconnectを呼び出します。

- プロトタイプ

```
OnTpconnect(VARIANT svc, VARIANT data, VARIANT len, VARIANT flags)
```

- パラメータ

パラメータ	説明
svc	通信接続を行うサービス名です
data	接続を設定する手順で、サービス・ルーチンに渡そうとするデータのポインターです
len	接続を設定する手順で、サービス・ルーチンに渡そうとするデータのサイズです
flags	通信接続を行うときに設定できるフラグです。 使用可能なフラグは、TPNOTRAN/ TPSENDONLY/ TPRECVONLY/ TPNOTIME/ TPSIGRSTRTなどがあります。OnTpconnectの場合は、TPSENDONLYまたはTPRECVONLYを必ず設定します

- 戻り値

戻り値	説明
記述子(descriptor)	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です。OnGettperrno()を呼び出してエラー値を取得します

- 関連関数

OnTprecv()、OnTpseend(), OnTpdiscon(), OnGettperrno()

7.2.8. OnTpdicon

会話型通信の接続を終了する関数です。内部的にtpdisconを呼び出します。

- プロトタイプ

```
OnTpdicon(VARIANT cd)
```

- パラメータ

パラメータ	説明
cd	OnTpconnectが返した記述子です

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合は
-1	関数呼び出しに失敗した場合は。OnGettperrno()を呼び出してエラー値を取得します

- 関連関数

OnTpconnect()、OnTprecv()、OnTpsend()、OnGettperrno()

7.2.9. OnTpend

Tmaxシステムとの接続を解除する関数です。内部的にtpendを呼び出します。

- プロトタイプ

```
OnTpend( )
```

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合は
-1	関数呼び出しに失敗した場合は。OnGettperrno()を呼び出してエラー値を取得します

- 関連関数

OnTpstart()、OnGettperrno()

7.2.10. OnTpfree

バッファ・タイプ(Typed Buffer)に割り当てられたメモリを解除する関数です。内部的にtpfreeを呼び出します。

- プロトタイプ

```
OnTpfree(VARIANT ptr)
```

- パラメータ

パラメータ	説明
ptr	メモリを解除したいバッファのポインタです

- 戻り値

関数callerに何の値も返しません。

- 関連関数

OnTpalloc()

7.2.11. OnTpget

OnTpallocなどで割り当てられたバッファのポインタが指し示す値を取得する関数です。

- プロトタイプ

```
OnTpget(VARIANT buf, VARIANT *loc)
```

- パラメータ

パラメータ	説明
buf	OnTpallocを利用して取得したバッファのポインタです
value	値を取得する変数です

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 例

```

Dim buf
Dim rVal
Dim msg
Dim i
...
buf = OnTpalloc ("STRING", "", CINT(0))
...
OnTpput buf, "abcd"
rVal = OnTpcall("TOUPPER", buf, CINT(0), buf2, i, CINT(0))
...
OnTpget buf, msg
...

```

- 関連関数

OnTpput()

7.2.12. OnTpgetrply

非同期的に要求したサービスの応答を受信する関数です。内部的にtpgetrplyを呼び出します。

- プロトタイプ

```
OnTpgetrply(VARIANT cd, VARIANT *data, VARIANT *len, VARIANT flags)
```

- パラメータ

パラメータ	説明
cd	OnTpacallが返した記述子です
data	受信される応答バッファのポインターです。OnTpallocまたはOnFballocで割り当てられたバッファのポインターである必要があります
len	受信したデータ長を取得します
flags	tpcallを行うためのフラグです

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です。OnGettperrno()を呼び出してエラー値を取得します

- 関連関数

OnTppalloc(), OnTppacall(), OnGettperrno()

7.2.13. OnTpput

OnTppallocで割り当てられたバッファのポインタが指し示す場所にvalueを設定する関数です。

- プロトタイプ

```
OnTpput(VARIANT *buf, VARIANT value)
```

- パラメータ

パラメータ	説明
buf	OnTppallocで取得したバッファのポインタです
value	bufに設定する値です

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 例

```
Dim buf
Dim rVal
Dim msg
Dim i
...
buf = OnTppalloc ("STRING", "", CINT(0))
...
OnTpput buf, "abcd"
rVal = OnTppacall("TOUPPER", buf, CINT(0), buf2, i, CINT(0))
...
OnTpget buf, msg
...
```

- 関連関数

OnTpget()

7.2.14. OnTprealloc

バッファ・タイプ(Typed Buffer)を再割り当てする関数です。内部的にtpreallocを呼び出します。

- プロトタイプ

```
OnTprealloc(VARIANT ptr, VARIANT size)
```

- パラメータ

パラメータ	説明
ptr	再割り当てするバッファ・タイプのポインターです
size	再割り当てするバッファ・タイプのサイズです

- 戻り値

戻り値	説明
適切なバッファ・タイプのポインター	関数呼び出しに成功した場合です
NULL	関数呼び出しに失敗した場合です。OnGettperrno()を呼び出してエラー値を取得します

- 関連関数

OnTpalloc()、OnTpfree()、OnGettperrno()

7.2.15. OnTprecv

会話型通信で接続された相手プログラムから送信されたデータを受信するために使用される関数です。内部的にtprecvを呼び出します。

- プロトタイプ

```
OnTprecv(VARIANT cd, VARIANT *data, VARIANT *len, VARIANT flags, VARIANT *revent)
```

- パラメータ

パラメータ	説明
cd	OnTpconnectが返した記述子です
data	OnTpallocまたはOnFballocなどで割り当てられた、受信するデータを取得するバッファです
len	データ長を取得する変数です

パラメータ	説明
flags	会話型サービスとの通信で設定可能なフラグです。使用できるフラグには、TPNOCHANGE/ TPNOBLOCK/ TPNOTIME/ TPSIGRSTRTなどがあります
revent	記述子のcdに対するイベントが存在する場合、OnTpsendは失敗し、データは送信されません。イベント・タイプはreventに返されます。使用可能なイベントには、TPEV_DISCONIMM/ TPEV_SVCERR/ TPEV_SVCFAILなどがあります。詳細説明については、『Tmax リファレンスガイド』を参照してください

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です。リターン時にreventがTREV_SVCFAILの場合は、OnGettpurcode()でサービスからtpreturn()を呼び出すときに渡されたurcode値になります

- 関連関数

OnTpconnect()、OnTpsend()、OnTpdicon()、OnGettperrno()

7.2.16. OnTpsend

会話型通信を行うときにメッセージを送信する関数です。内部的にtpsendを呼び出します。Callerは通信制御権を有する必要があります。

- プロトタイプ

```
OnTpsend(VARIANT cd, VARIANT data, VARIANT len, VARIANT flags, VARIANT *revent)
```

- パラメータ

パラメータ	説明
cd	OnTpconnectが返した記述子です
data	OnTpallocまたはOnFballocで割り当てられたバッファです
len	データ長です
flags	会話型サービスとの通信で設定可能なフラグです。使用できるフラグには、TPNOBLOCK/ TPNOTIME/ TPNOVONLY/ TPSIGRSTRTなどがあります
revent	記述子のcdに対するイベントが存在する場合、OnTpsendは失敗し、データは送信されません。イベント・タイプはreventに返されます。使用可能なイベントには、

パラメータ	説明
	TPEV_DISCONIMM/ TPEV_SVCERR/ TPEV_SVCFAILなどがあります。詳細説明については、『Tmax リファレンスガイド』を参照してください

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です。リターン時にreventがTREV_SVCFAILの場合は、OnGettpurcode()でサービスからtpreturn()を呼び出すときに渡されたurcode値になります

- 関連関数

OnTpconnect()、OnTprecv()、OnTpdiscon()、OnGettperrno()

7.2.17. OnTpstart

Tmaxシステムに接続する関数です。内部的にtpstartを呼び出します。TPSTART_Tのパラメータに格納するメンバーを取得します。

- プロトタイプ

```
OnTpstart(VARIANT username, VARIANT cltname, VARIANT dompwd, VARIANT usrpwd,
          VARIANT flags)
```

- パラメータ

パラメータ	説明
username	クライアント名です
cltname	システム接続セキュリティのパスワードです
dompwd	ユーザー認証セキュリティのアカウントです
usrpwd	ユーザー認証セキュリティのパスワードです
flags	非要求メッセージ・タイプとシステム・アクセス方法を指定します

- 戻り値

戻り値	説明
0 (main)	関数呼び出しに成功した場合です
1 (backup)	

戻り値	説明
-1	関数呼び出しに失敗した場合です。OnGettperrno()を呼び出してエラー値を取得します

- 関連関数

OnTpend()、OnTmaxreadenv()、OnGettperrno()

7.2.18. OnTptypes

バッファ・タイプおよび下位タイプに関する情報を提供する関数です。内部的にtptypesを呼び出します。

- プロトタイプ

```
OnTptypes(VARIANT ptr, VARIANT *type, VARIANT *subtype)
```

- パラメータ

パラメータ	説明
ptr	OnTpallocまたはOnFballocで割り当てられたバッファのポインターです
type	OnTpallocまたはOnFballocで割り当てられたバッファ・ポインターのバッファ・タイプです
subtype	OnTpallocまたはOnFballocで割り当てられたバッファ・ポインターのサブタイプです

- 戻り値

戻り値	説明
バッファ・サイズ	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です。OnGettperrno()を呼び出してエラー値を取得します

- 関連関数

OnTpsubtypes()、OnGettperrno()

7.3. その他のインターフェース・メソッド

7.3.1. OnTmax_chk_conn

クライアントがTmaxシステムと接続されている状態なのかをチェックする関数です。内部的にtptobackupを呼び出します。

- プロトタイプ

```
OnTmax_chk_conn(VARIANT timeout)
```

- パラメータ

パラメータ	説明
timeout	ブロッキング・タイムアウト時間を設定します

- 戻り値

戻り値	説明
0	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です。OnGettperrno()を呼び出してエラー値を取得します

7.3.2. OnTmaxreadenv

環境変数をファイルから読み込む関数です。内部的にtmaxreadenvを呼び出します。

- プロトタイプ

```
OnTmaxreadenv(VARIANT file, VARIANT label)
```

- パラメータ

パラメータ	説明
file	接続するシステムの環境情報が保存されているファイル名です
label	ファイル内に登録されている環境情報の記述子です

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 関連関数

OnTpstart()

7.3.3. OnTp_sleep

最大sec時間の間にスリープ(sleep)状態になり、その時間内にデータが受信されると即リターンする関数です。内部的にtp_sleepを呼び出します。

- プロトタイプ

```
OnTp_sleep(VARIANT sec)
```

- パラメータ

パラメータ	説明
sec	待機する時間を秒単位の正の整数で設定します

- 戻り値

戻り値	説明
-1	関数呼び出しに失敗した場合です。OnGettperrno()を呼び出してエラー値を取得します
0	sec時間までデータが受信されない場合です
正の整数	データが受信された場合です

- 関連関数

OnTp_usleep()

7.3.4. OnTp_usleep

最大usec時間の間にスリープ状態になり、その時間内にデータが受信されると即リターンする関数です。内部的にtp_usleepを呼び出します。

- プロトタイプ

```
OnTp_sleep(VARIANT usec)
```

- パラメータ

パラメータ	説明
usec	待機する時間を100万分の1秒単位の正の整数で設定します

- 戻り値

戻り値	説明
-1	関数呼び出しに失敗した場合は。OnGetperrno()を呼び出してエラー値を取得します
0	usec時間までデータが受信されない場合は
正の整数	データが受信された場合は

- 関連関数

OnTp_usleep()

7.3.5. OnTpdeq

tpenq()を利用してサービスを要求した結果や、tpenq()を利用するときにサービス名をNULLにして保存したデータを読み込む関数です。内部的にtpdeqを呼び出します。ただし、tpenq()を利用するとき、flagsにTPNOREPLYを設定したサービス結果は取得できません。

- プロトタイプ

```
OnTpdeq(VARIANT qname, VARIANT svc, VARIANT *data, VARIANT *len, VARIANT flags)
```

- パラメータ

パラメータ	説明
qname	データを保存するRQ名です
svc	OnTpenq()で使ったサービス名と同じ名前です
data	OnTpallocまたはOnFballocで割り当てられたサービス処理結果を取得するバッファです
len	受信するデータ長です
flags	tpdeqと同じフラグを設定することができます

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です。OnGetperrno()を呼び出してエラー値を取得します

- 関連関数

OnTpenq()

7.3.6. OnTpenq

RQにデータを保存する関数です。内部的にtpenqを呼び出します。

- プロトタイプ

```
OnTpenq(VARIANT qname, VARIANT svc, VARIANT data, VARIANT len, VARIANT flags)
```

- パラメータ

パラメータ	説明
qname	データを保存するRQ名です
svc	要求するサービス名のsvc名がNULLでない場合は、直ちにサービスを要求します。svc名がNULLの場合、データはRQに保存され、サービスは実行されません
data	サービスで要求するデータのポインターです。NULLの場合を除けば、常にOnTppallocまたはOnFballocで割り当てられたバッファである必要があります
len	送信するデータ長です
flags	tpenqと同じフラグを設定することができます

- 戻り値

戻り値	説明
qd (queue descriptor)	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です。OnGetperrno()を呼び出してエラー値を取得します

- 関連関数

OnTpdeq()

7.3.7. OnTperrordetail

Tmaxシステムを呼び出すとき発生したエラーの詳細情報を取得するときに使用する関数です。内部的にtperrordetailを呼び出します。

- プロトタイプ

```
OnTperrordetail(VARIANT i)
```

- パラメータ

パラメータ	説明
i	エラー値です

- 戻り値

戻り値	説明
-1	不明なエラーが発生した場合です
1	アプリケーション・エラーが発生した場合です
2	システム・エラーが発生した場合です

7.3.8. OnTpextsvcinfnfo

OnTpdeq()を利用してRQからデータを読み込む場合、当該データの詳細情報を取得するときに使用する関数です。内部的にtpextsvcinfnfoを呼び出します。

- プロトタイプ

```
OnTpextsvcinfnfo(VARIANT data, VARIANT *svc, VARIANT *type, VARIANT *errcode)
```

- パラメータ

パラメータ	説明
data	OnTpallocまたはOnFballocで割り当てられ、OnTpdeq()を利用してRQから読み込んだデータが保存されているポインターです
svc	当該サービス名が取得できる変数です
type	当該データ処理結果で、下記の値を有します <ul style="list-style-type: none">– TPREQ(0)– TPFAIL(1)– TPSUCCESS(2)

パラメータ	説明
	– TPERR(-1)
errcode	エラー状況の場合、この変数に該当するエラーコード値が保存されます

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です。OnGettperrno()を呼び出してエラー値を取得します

7.3.9. OnTpextsvcname

OnTpdeq()を利用してRQからデータを読み込む場合、当該データのサービス名を確認するために使用する関数です。内部的にtpextsvcnameを呼び出します。

- プロトタイプ

```
OnTpextsvcname(VARIANT data, VARIANT *svc)
```

- パラメータ

パラメータ	説明
data	OnTpallocまたはOnFballocで割り当てられ、OnTpdeq()を利用してRQから読み込んだデータが保存されているポインターです
svc	当該サービス名を取得する変数です

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です。OnGettperrno()を呼び出してエラー値を取得します

7.3.10. OnTpgetenv

strという名前で登録された環境変数の値を返す関数です。内部的にtpgetenvを呼び出します。

- プロトタイプ

```
OnTpgetenv(VARIANT str)
```

- パラメータ

パラメータ	説明
str	環境変数に登録された名前です

- 戻り値

戻り値	説明
環境変数値のポインター	関数呼び出しに成功した場合です
NULL	関数呼び出しに失敗した場合です

- 関連関数

OnTpputenv()

7.3.11. OnTpputenv

strで入力された「名前=値」を環境変数に適用する関数です。既存の環境変数が存在する場合は修正し、存在しない場合は追加します。内部的にtpputenvを呼び出します。

- プロトタイプ

```
OnTpputenv(VARIANT str)
```

- パラメータ

パラメータ	説明
str	環境変数に登録する値です

- 戻り値

戻り値	説明
0	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です

- 関連関数

OnTpgetenv()

7.3.12. OnTpqstat

現在のキューに保存されたデータの統計値を返す関数です。内部的にtpqstatを呼び出します。

- プロトタイプ

```
OnTpqstat(VARIANT qname, VARIANT type)
```

- パラメータ

パラメータ	説明
qname	Tmax環境ファイルに登録されたRQ名です
type	フラグで下記の値が使用できます <ul style="list-style-type: none">– 0 : fail/request/replyキューのデータ統計を出すときに使用します– 1 : failキューのデータ統計を出すときに使用します– 2 : requestキューのデータ統計を出すときに使用します– 3 : replyキューのデータ統計を出すときに使用します

- 戻り値

戻り値	説明
統計値	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です。OnGetperrno()を呼び出してエラー値を取得します

- 関連関数

OnTpsvcstat()

7.3.13. OnTpqsvcstat

現在のキューに保存されたデータのうち、特定のサービスに対する統計値を返す関数です。内部的にtpqsvcstatを呼び出します。

- プロトタイプ

```
OnTpqsvcstat(VARIANT qname, VARIANT svc, VARIANT type)
```

- パラメータ

パラメータ	説明
qname	Tmax環境ファイルに登録されたRQ名です
svc	統計を出したいサービス名です。この名前がNULLの場合、OnTpqstat()と同様な結果を返します
type	フラグで下記の値が使用できます – 0 : fail/request/replyキューのデータ統計を出すときに使用します – 1 : failキューのデータ統計を出すときに使用します – 2 : requestキューのデータ統計を出すときに使用します – 3 : replyキューのデータ統計を出すときに使用します

- 戻り値

戻り値	説明
統計値	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です。OnGettperrno()を呼び出してエラー値を取得します

- 関連関数

OnTpqstat()

7.3.14. OnTpreissue

OnTpdeq()を利用してRQでサービスを実行する際に、ネットワークの不安定やその他のサーバー・エラーによってサービスに失敗し、failキューに蓄積されたクライアントのサービス要求データを再びrequestキューに格納する関数です。内部的にtpreissueを呼び出します。

- プロトタイプ

```
OnTpreissue(VARIANT qname, VARIANT filter, VARIANT flags)
```

- パラメータ

パラメータ	説明
qname	Tmax環境ファイルに登録されたRQ名です
filter	このパラメータは現在サポートしていないため、NULLに設定します
flags	このパラメータは現在サポートしていないため、0に設定します

- 戻り値

戻り値	説明
0(正常)	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です。OnGetperrno()を呼び出してエラー値を取得します

- 関連関数

OnTpenq()、OnTpdeq()

7.3.15. OnTppreset

現在接続されているクライアントを即解除する関数です。内部的にtpresetを呼び出します。

- プロトタイプ

```
OnTppreset ( )
```

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です。OnGetperrno()を呼び出してエラー値を取得します

7.3.16. OnTpset_timeout

サーバーに設定されているサービス制限時間、すなわちブロッキング・タイムアウト時間を変更するときに使用する関数です。内部的にtpset_timeoutを呼び出します。

- プロトタイプ

```
OnTpset_timeout (VARIANT sec)
```

- パラメータ

パラメータ	説明
sec	ブロッキング・タイムアウト時間を設定します

- 戻り値

戻り値	説明
0	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です。OnGettperrno()を呼び出してエラー値を取得します

7.3.17. OnTpsubqname

サブキュー番号に該当するキューの名前を返す関数です。内部的にtpsubqnameを呼び出します。

- プロトタイプ

```
OnTpsubqname(VARIANT type)
```

- パラメータ

パラメータ	説明
type	下記のうち1つをサブキュー番号に指定します <ul style="list-style-type: none"> – 0 : RQ_ANY(fail/request/replyキュー) – 1 : RQ_FAIL(failキュー) – 2 : RQ_REQ(requestキュー) – 3 : RQ_RPLY(replyキュー)

- 戻り値

戻り値	説明
サブキュー名	関数呼び出しに成功した場合です
NULL	関数呼び出しに失敗した場合です。OnGettperrno()を呼び出してエラー値を取得します

7.3.18. OnTptobackup

クライアントがTmaxのバックアップ・システムに接続する関数です。内部的にtptobackupを呼び出します。関数を使用するには、TMAX_BACKUP_ADDRとTMAX_BACKUP_PORTが設定されている必要があります。

- プロトタイプ

```
OnTptobackup ( )
```

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です。OnGettperrno()を呼び出してエラー値を取得します

- 関連関数

OnTpstart()

7.4. FDLインターフェース・メソッド

7.4.1. OnFballoc

フィールド・バッファのサイズを計算した後、メモリーを割り当てる関数です。内部的にfballocを呼び出します。

- プロトタイプ

```
OnFballoc(VARIANT count, VARIANT len)
```

- パラメータ

パラメータ	説明
count	データを保存するフィールド数です
len	データの全体サイズです(単位：バイト)

- 戻り値

戻り値	説明
割り当てたメモリーのポインター	関数呼び出しに成功した場合です
NULL	関数呼び出しに失敗した場合です。OnGetperrno()を呼び出してエラー値を取得します

- 関連関数

OnFbcalcsize(), OnTpalloc()

7.4.2. OnFbcalcsize

OnTpalloc()を利用してフィールド・バッファをメモリーに割り当てるときに、適切なメモリーを割り当てるため、事前にフィールド・バッファ・サイズを計算する関数です。内部的にfbcalcsizeを呼び出します。

- プロトタイプ

```
OnFbcalcsize(VARIANT count, VARIANT datalen)
```

- パラメータ

パラメータ	説明
count	データを保存するフィールド数です
datalen	フィールド・バッファの全体データ・サイズをバイト単位で指定します

- 戻り値

与えられたcountとdatalenに基づいて、フィールド・バッファのサイズをバイト単位で返します。

- 関連関数

OnFballoc(), OnTpalloc()

7.4.3. OnFbdelall

フィールド・バッファで指定したフィールド・キーの全データを削除する関数です。内部的にfbdelallを呼び出します。

- プロトタイプ

```
OnFbdelall(VARIANT fbuf, VARIANT fldkey)
```


- パラメータ

パラメータ	説明
fbuf	OnTpallocまたはOnFballocで割り当てられたバッファ・ポインターです
fldkey	フィールド・キーです

- 戻り値

戻り値	説明
削除したフィールド数	関数呼び出しに成功した場合です
-1	フィールド・バッファに、指定したフィールド・キーが存在しない場合です。OnGetperrno()を呼び出してエラー値を取得します

7.4.4. OnFbdelete

フィールド・バッファに指定したフィールド・キーのフィールド順に該当するデータを削除する関数です。内部的にfbdeleteを呼び出します。

- プロトタイプ

```
OnFbdelete(VARIANT fbuf, VARIANT fldkey, VARIANT nth)
```

- パラメータ

パラメータ	説明
fbuf	OnTpallocまたはOnFballocで割り当てられたバッファ・ポインターです
fldkey	フィールド・キーです
nth	フィールド順です

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です。OnGetperrno()を呼び出してエラー値を取得します

- 関連関数

OnFbupdate()、OnFbinsert()

7.4.5. OnFbfldcount

フィールド・バッファ内に存在するすべてのフィールド数を返す関数です。内部的にfbfldcountを呼び出します。

- プロトタイプ

```
OnFbfldcount(VARIANT fbuf)
```

- パラメータ

パラメータ	説明
fbuf	OnTpallocまたはOnFballocで割り当てられたバッファ・ポインターです

- 戻り値

指定したフィールド・バッファで現在保存されているフィールド数を返します。フィールド・バッファに保存されているフィールドが存在しない場合は0を返します。

7.4.6. OnFbfree

フィールド・バッファのメモリーを解除する関数です。内部的にfbfreeを呼び出します。

- プロトタイプ

```
OnFbfree(VARIANT fbuf)
```

- パラメータ

パラメータ	説明
fbuf	OnTpallocまたはOnFballocで割り当てられたバッファ・ポインターです

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	フィールド・バッファの解除に失敗した場合です (OnGetfberno()) を呼び出してエラー値を取得します)

- 関連関数

OnFballoc(), OnTpalloc(), OnTpfree()

7.4.7. OnFbget

フィールド・バッファでフィールド・キーに該当するフィールド・データを返す関数です。内部的にfbgetを呼び出します。

- プロトタイプ

```
OnFbget(VARIANT fbuf, VARIANT fldkey, VARIANT *loc, VARIANT *fldlen)
```

- パラメータ

パラメータ	説明
fbuf	OnTpallocまたはOnFballocで割り当てられたバッファ・ポインタです
fldkey	フィールド・キーです
loc	フィールド・データを取得するバッファです。OnTpallocで割り当てられたバッファである必要があります
fldlen	取得したフィールド・データ長です

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です。OnGetperrno()を呼び出してエラー値を取得します

- 関連関数

OnFbput()、OnFbgett()

7.4.8. OnFbget_fbsize

フィールド・バッファのサイズを返す関数です。内部的にfbfreeを呼び出します。

- プロトタイプ

```
OnFbget_fbsize(VARIANT fbuf)
```

- パラメータ

パラメータ	説明
fbuf	OnTpallocまたはOnFballocで割り当てられたバッファ・ポインタです

- 戻り値

メモリーに割り当てられたフィールド・バッファのサイズを返します。

- 関連関数

OnFballoc()、OnTpalloc()

7.4.9. OnFbget_fldkey

指定したフィールド名と一致するフィールド・キーを返す関数です。内部的にfbget_fldkeyを呼び出します。

- プロトタイプ

```
OnFbget_fldkey(VARIANT name)
```

- パラメータ

パラメータ	説明
name	確認したいフィールド・キーのフィールド名です

- 戻り値

戻り値	説明
フィールド・キー	関数呼び出しに成功した場合です
0	フィールド表に、指定したフィールド名が存在しない場合です

- 関連関数

OnFbget_fldname()、OnFbget_fldtype()

7.4.10. OnFbget_fldname

指定したフィールド・キーと一致するフィールド名を返す関数です。内部的にfbget_fldkeyを呼び出します。

- プロトタイプ

```
OnFbget_fldname(VARIANT fldkey)
```

- パラメータ

パラメータ	説明
fldkey	確認したいフィールド名のフィールド・キーです

- 戻り値

戻り値	説明
指定したフィールド・キーと一致するフィールド名	関数呼び出しに成功した場合です
NULL	フィールド表に、指定したフィールド・キーのフィールド名が存在しない場合です

- 関連関数

OnFbget_fldkey()、OnFbget_fldtype()

7.4.11. OnFbget_fldtype

指定したフィールド・キーのフィールド・タイプを整数で返す関数です。内部的にfbget_fldtypeを呼び出します。

- プロトタイプ

```
OnFbget_fldtype(VARIANT fldkey)
```

- パラメータ

パラメータ	説明
fldkey	フィールド・キーです

- 戻り値

戻り値	フィールド・タイプ
1	character
2	short integer
3	integer
4	long integer
5	float
6	double
7	string

戻り値	フィールド・タイプ
8	character array(CARRAY)

- 関連関数

OnFbget_fldkey()、OnFbget_fldname()

7.4.12. OnFbget_strfldtype

指定したフィールド・キーのフィールド・タイプを文字列で返す関数です。内部的にfbget_strfldtypeを呼び出します。

- プロトタイプ

```
OnFbget_strfldtype(VARIANT fldkey)
```

- パラメータ

パラメータ	説明
fldkey	フィールド・キーです

- 戻り値

戻り値	説明
空文字列	パラメータで取得したフィールド・キーに該当するフィールド・タイプが存在しないか、フィールド・キーが誤って指定された場合です
文文字列	パラメータで取得したフィールド・キーに該当するフィールド・タイプが存在する場合、当該フィールド・タイプを文字列で返します

- 関連関数

OnFbget_fldtype()

7.4.13. OnFbget_unused

メモリーに割り当てられたフィールド・バッファの中で、使用前のフィールド・バッファ・サイズを計算する関数です。内部的にfbget_unusedを呼び出します。

- プロトタイプ

```
OnFbget_unused(VARIANT fbuf)
```

- パラメータ

パラメータ	説明
fbuf	OnTpallocまたはOnFballocで割り当てられたバッファ・ポインターです

- 戻り値

メモリーに割り当てられたフィールド・バッファの中で、使用前のメモリー・サイズをバイト単位で返します。

- 関連関数

OnFbget_used()

7.4.14. OnFbget_used

メモリーに割り当てられたフィールド・バッファの中で、使用中のフィールド・バッファ・サイズを計算する関数です。内部的にfbget_usedを呼び出します。

- プロトタイプ

```
OnFbget_used(VARIANT fbuf)
```

- パラメータ

パラメータ	説明
fbuf	OnTpallocまたはOnFballocで割り当てられたバッファ・ポインターです

- 戻り値

メモリーに割り当てられたフィールド・バッファの中で、使用中のメモリー・サイズをバイト単位で返します。

- 関連関数

OnFbget_unused()

7.4.15. OnFbgetc

フィールド・バッファに保存されているフィールド・データのタイプを指定したタイプに変換して順次返す関数です。内部的にfbgetcを呼び出します。

- プロトタイプ

```
OnFbgetc(VARIANT fbuf, VARIANT fldkey, VARIANT *loc, VARIANT *len, VARIANT *pos,
          VARIANT totype)
```

● パラメータ

パラメータ	説明
fbuf	OnTpallocまたはOnFballocで割り当てられたバッファ・ポインターです
fldkey	フィールド・キーです
loc	フィールド・データを取得するバッファです。OnTpallocで割り当てられたバッファである必要があります
fldlen	取得するフィールド・データ長です
pos	次に読み込むデータの位置です
totype	<p>取得したフィールド・データのタイプを変換します。</p> <p>下記のフィールド・タイプが使用できます</p> <ul style="list-style-type: none"> – FB_CARRAY(8) – FB_SHORT(2) – FB_DOUBLE(6) – FB_LONG(4) – FB_STRING(7) – FB_CHAR(1) – FB_FLOAT(5) – FB_INT(3)

● 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	フィールド・バッファに、指定したフィールド・キーが存在しない場合です。OnGetperrno()を呼び出してエラー値を取得します

● 関連関数

OnFbgetc()、OnFbgetf()

7.4.16. OnFbgetf

フィールド・バッファに保存されているフィールド・データを順次リターンする関数です。内部的にfbgetfを呼び出します。

- プロトタイプ

```
OnFbgetf(VARIANT fbuf, VARIANT fldkey, VARIANT *loc, VARIANT *fldlen,  
          VARIANT *pos)
```

- パラメータ

パラメータ	説明
fbuf	OnTpallocまたはOnFballocで割り当てられたバッファ・ポインターです
fldkey	フィールド・キーです
loc	フィールド・データを取得するバッファです。OnTpallocで割り当てられたバッファである必要があります
fldlen	取得するフィールド・データ長です
pos	次に読み込むデータの位置です

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	フィールド・バッファに、指定したフィールド・キーが存在しない場合です。OnGettperrno()を呼び出してエラー値を取得します

- 関連関数

OnFbget()、OnFbgetc()

7.4.17. OnFbgetnth

指定したフィールド・キーに保存されているフィールド・データの中で、指定したデータが保存されているフィールド数を返す関数です。内部的にfbgetnthを呼び出します。

- プロトタイプ

```
OnFbgetnth(VARIANT fbuf, VARIANT fldkey, VARIANT value, VARIANT len)
```

- パラメータ

パラメータ	説明
fbuf	OnTpallocまたはOnFballocで割り当てられたバッファ・ポインターです
fldkey	フィールド・キーです
value	比較するデータです
len	比較するデータ長です

- 戻り値

戻り値	説明
フィールド順	関数呼び出しに成功した場合です
-1	フィールド・バッファに、指定したフィールド・キーのフィールド順が存在しない場合です。OnGetperrno()を呼び出してエラー値を取得します

7.4.18. OnFbgetntht

指定したフィールド・キーに保存されているフィールド・データの中で、指定したデータが保存されているフィールド順を返す関数です。内部的にfbgetnthtを呼び出します。

- プロトタイプ

```
OnFbgetntht(VARIANT fbuf, VARIANT fldkey, VARIANT value, VARIANT len,
            VARIANT fromtype)
```

- パラメータ

パラメータ	説明
fbuf	OnTpallocまたはOnFballocで割り当てられたバッファ・ポインターです
fldkey	フィールド・キーです
value	比較するデータです
len	比較するデータ長です
fromtype	<p>フィールド・タイプです。ユーザーがfromtypeで指定したデータ型と、fldkeyで指定したフィールド・キーのフィールド・タイプが異なる場合、valueのデータ型をフィールド・タイプに変換した後、value、fldlen、フィールド・タイプが一致するフィールド順を見つけて返します。</p> <p>下記のフィールド・タイプが使用できます</p> <ul style="list-style-type: none"> – FB_CHAR(1) – FB_SHORT(2)

パラメータ	説明
	<ul style="list-style-type: none"> – FB_INT(3) – FB_LONG(4) – FB_FLOAT(5) – FB_DOUBLE(6) – FB_STRING(7) – FB_CARRAY(8)

- 戻り値

戻り値	説明
フィールド順	関数呼び出しに成功した場合です
-1	フィールド・バッファに、指定したフィールド・キーのフィールド順が存在しない場合です。OnGetperrno()を呼び出してエラー値を取得します

- 関連関数

OnFbgetntht()

7.4.19. OnFbgetval

フィールド・バッファで指定したフィールド・キーのフィールド順に該当するフィールド・データを返す関数です。内部的にfbgetvalを呼び出します。

- プロトタイプ

```
OnFbgetval(VARIANT fbuf, VARIANT fldkey, VARIANT nth, VARIANT *len)
```

- パラメータ

パラメータ	説明
fbuf	OnTpallocまたはOnFballocで割り当てられたバッファ・ポインタです
fldkey	フィールド・キーです
nth	フィールド順です
len	読み込むデータ長です

- 戻り値

戻り値	説明
文字列	関数呼び出しに成功した場合です
NULL	フィールド・バッファに、指定したフィールド・キーのフィールド順が存在しない場合です。OnGetperrno()を呼び出してエラー値を取得します

- 関連関数

OnFbgetvalt()、OnFbgetvals()、OnFbgetvali()

7.4.20. OnFbgetvali

指定したフィールド・キーのフィールド順に該当するフィールド・データを整数に変換して返す関数です。内部的にfbgetvaliを呼び出します。

- プロトタイプ

```
OnFbgetvali(VARIANT fbuf, VARIANT fldkey, VARIANT nth)
```

- パラメータ

パラメータ	説明
fbuf	OnTpallocまたはOnFballocで割り当てられたバッファ・ポインターです
fldkey	フィールド・キーです
nth	フィールド順です

- 戻り値

戻り値	説明
整数	関数呼び出しに成功した場合です
-1	フィールド・バッファで指定したフィールド・キーが存在しないか、関数の実行中に他のエラーが発生した場合です。OnGetperrno()を呼び出してエラー値を取得します

- 関連関数

OnFbgetvalt()、OnFbgetvals()、OnFbgetval()

7.4.21. OnFbgetvals

指定したフィールド・キーのフィールド順に該当するフィールド・データを文字列に変換して返す関数です。内部的にfbgetvalsを呼び出します。

- プロトタイプ

```
OnFbgetvals(VARIANT fbuf, VARIANT fldkey, VARIANT nth)
```

- パラメータ

パラメータ	説明
fbuf	OnTpallocまたはOnFballocで割り当てられたバッファ・ポインターです
fldkey	フィールド・キーです
nth	フィールド順です

- 戻り値

戻り値	説明
文字列	関数呼び出しに成功した場合です
NULL	フィールド・バッファで指定したフィールド・キーが存在しないか、関数の実行中に他のエラーが発生した場合です。OnGettperrno()を呼び出してエラー値を取得します

- 関連関数

OnFbgetvalt()、OnFbgetval()、OnFbgetvali()

7.4.22. OnFbgetvalt

指定したフィールド・キーのフィールド順に該当するフィールド・データを指定したタイプに変換して返す関数です。内部的にfbgetvaltを呼び出します。

- プロトタイプ

```
OnFbgetvalt(VARIANT fbuf, VARIANT fldkey, VARIANT nth, VARIANT *len,  
            VARIANT totype)
```

- パラメータ

パラメータ	説明
fbuf	OnTpallocまたはOnFballocで割り当てられたバッファ・ポインターです

パラメータ	説明
fldkey	フィールド・キーです
nth	フィールド順です
len	読み込むデータ長です
type	読み込んだフィールド・データのタイプを変換します。 下記のフィールド・タイプが使用できます – FB_CHAR(1) – FB_SHORT(2) – FB_INT(3) – FB_LONG(4) – FB_FLOAT(5) – FB_DOUBLE(6) – FB_STRING(7) – FB_CARRAY(8)

- 戻り値

戻り値	説明
文字列	関数呼び出しに成功した場合です
NULL	フィールド・バッファに、指定したフィールド・キーのフィールド順が存在しない場合です。OnGetperrno()を呼び出してエラー値を取得します

- 関連関数

OnFbgetval()、OnFbgetvals()、OnFbgetvali()

7.4.23. OnFbinit

フィールド・バッファを初期化する関数です。内部的にfbinitを呼び出します。

- プロトタイプ

```
OnFbinit(VARIANT fbuf, VARIANT len)
```

- パラメータ

パラメータ	説明
fbuf	OnTpallocまたはOnFballocで割り当てられたバッファのポインターです
len	初期化するバッファのサイズです

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	フィールド・バッファの初期化に失敗した場合です(OnGetfberno())を呼び出してエラー値を取得します)

7.4.24. OnFbinsert

フィールド・バッファに指定したフィールド・キーのフィールド順にデータを保存する関数です。内部的にfbinsertを呼び出します。

- プロトタイプ

```
OnFbinsert(VARIANT fbuf, VARIANT fldkey, VARIANT nth, VARIANT value,
           VARIANT fldlen)
```

- パラメータ

パラメータ	説明
fbuf	OnTpallocまたはOnFballocで割り当てられたバッファ・ポインターです
fldkey	フィールド・キーです
nth	フィールド順です
value	追加するデータです
fldlen	追加するデータ長です

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です。OnGettperrno()を呼び出してエラー値を取得します

- 関連関数

OnFbdelete()、OnFbupdate()

7.4.25. OnFbisfbuf

指定したフィールド・バッファが有効なのか(OnFballocまたはOnTpallocでバッファを割り当てられた変数なのか)を確認する関数です。内部的にfbisfbufを呼び出します。

- プロトタイプ

```
OnFbisfbuf(VARIANT fbuf)
```

- パラメータ

パラメータ	説明
fbuf	検査する変数です

- 戻り値

戻り値	説明
0	関数呼び出しに成功した場合です
1	指定したフィールド・バッファが有効な場合です

- 関連関数

OnFballoc()、OnTpalloc()

7.4.26. OnFbispres

フィールド・バッファに指定したフィールド・キーのフィールド順にフィールド・データが存在するか否かを確認する関数です。内部的にfbispresを呼び出します。

- プロトタイプ

```
OnFbispres(VARIANT fbuf, VARIANT fldkey, VARIANT nth)
```

- パラメータ

パラメータ	説明
fbuf	OnTpallocまたはOnFballocで割り当てられたバッファのポインターです
fldkey	フィールド・キーです
nth	フィールド順です

- 戻り値

戻り値	説明
0	関数呼び出しに成功した場合です
1	フィールド・バッファーで指定したフィールド・キーのフィールド順に、フィールド・データが存在する場合があります

7.4.27. OnFbkeynm_unload

メモリーにローディングされたFDLファイルを新たにローディングする場合、この関数を呼び出すことができます。この関数を呼び出すと、以前メモリーにローディングされたデータは削除され、環境変数のFDLFILEに設定されたファイルがメモリーにローディングされます。内部的にfbkeynm_unloadを呼び出します。

- プロトタイプ

```
OnFbkeynm_unload()
```

- 関連関数

OnFbnmkey_unload

7.4.28. OnFbkeyoccur

フィールド・バッファーで指定したフィールド・キーのフィールド順を返す関数です。内部的にfbkeyoccurを呼び出します。

- プロトタイプ

```
OnFbkeyoccur(VARIANT fbuf, VARIANT fldkey)
```

- パラメータ

パラメータ	説明
fbuf	OnTpallocまたはOnFballocで割り当てられたバッファー・ポインターです
fldkey	フィールド・キーです

- 戻り値

戻り値	説明
フィールド数	関数呼び出しに成功した場合です

戻り値	説明
0	フィールド・バッファで指定したフィールド・キーに、保存されたフィールド・データが存在しない場合です。OnGettperrno()を呼び出してエラー値を取得します
-1	関数呼び出しに失敗した場合です

7.4.29. OnFbmake_fldkey

新しいフィールド・キーを動的に作成する関数です。内部的にfbmake_fldkeyを呼び出します。

- プロトタイプ

```
OnFbmake_fldkey(VARIANT type, VARIANT no)
```

- パラメータ

パラメータ	説明
type	作成するフィールドのフィールド・タイプです。 下記のフィールド・タイプが使用できます – FB_CHAR(1) – FB_SHORT(2) – FB_INT(3) – FB_LONG(4) – FB_FLOAT(5) – FB_DOUBLE(6) – FB_STRING(7) – FB_CARRAY(8)
no	作成するフィールドの番号です

- 戻り値

戻り値	説明
フィールド・キー	指定されたタイプ(type)と番号(no)を組み合わせで新しいフィールド・キーを作成する場合です
文字列	パラメータで取得したフィールド・キーに該当するフィールド・タイプが存在する場合です

7.4.30. OnFbnmkey_unload

メモリーにローディングされたFDLファイルを新たにローディングする場合、この関数を呼び出すことができます。この関数を呼び出すと、以前メモリーにローディングされたデータは削除され、環境変数のFDLFILEに設定されたファイルがメモリーにローディングされます。内部的にfbnmkey_unloadを呼び出します。

- プロトタイプ

```
OnFbnmkey_unload()
```

- 関連関数

OnFbkeynm_unload

7.4.31. OnFbput

フィールドバッファーに新しいフィールドを追加する関数です。内部的にfbputを呼び出します。

- プロトタイプ

```
OnFbput(VARIANT fbuf, VARIANT fldkey, VARIANT value, VARIANT fldlen)
```

- パラメータ

パラメータ	説明
fbuf	OnTallocまたはOnFballocで割り当てられたバッファー・ポインターです
fldkey	フィールド・キーです
value	追加するデータです
fldlen	追加するデータ長です

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	関数呼び出しに失敗した場合です。OnGetperrno()を呼び出してエラー値を取得します

- 関連関数

OnFbget()、OnFbputt()

7.4.32. OnFbputt

データをフィールドバッファに指定したタイプに変換した後、新しいフィールドを追加して保存する関数です。内部的にfbputtを呼び出します。

- プロトタイプ

```
OnFbputt(VARIANT fbuf, VARIANT fldkey, VARIANT value, VARIANT len, VARIANT type)
```

- パラメータ

パラメータ	説明
fbuf	OnTpallocまたはOnFballocで割り当てられたバッファのポインターです
fldkey	フィールド・キーです
value	追加するデータです
len	追加するデータ長です
type	ユーザーが指定したデータ型です。データがフィールドに保存される前に、こちらで指定したtypeでフィールド・キーのタイプに変換されます。 typeに指定できる値は下記のとおりです – FB_CHAR(1) – FB_SHORT(2) – FB_INT(3) – FB_LONG(4) – FB_FLOAT(5) – FB_DOUBLE(6) – FB_STRING(7) – FB_CARRAY(8)

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合です
-1	フィールド・バッファでフィールドの追加に失敗した場合です。OnGetperrno()を呼び出してエラー値を取得します

- 関連関数

OnFbput()

7.4.33. OnFbrealloc

フィールドバッファをメモリに再割り当てする関数です。内部的にfbget_usedを呼び出します。

- プロトタイプ

```
OnFbrealloc(VARIANT fbuf, VARIANT ncount, VARIANT nlen)
```

- パラメータ

パラメータ	説明
fbuf	OnTpallocまたはOnFballocで割り当てられたバッファ・ポインターです
ncount	データを保存するフィールド数です
nlen	データの全体サイズです(単位: バイト)

- 戻り値

戻り値	説明
割り当てられたメモリ・ポインター	関数呼び出しに成功した場合です
NULL	関数呼び出しに失敗した場合です。OnGetperrno()を呼び出してエラー値を取得します

- 関連関数

OnFballoc()、OnTpalloc()

7.4.34. OnFbsterror

フィールド・バッファの操作時にエラー内容を文字列で返す関数です。内部的にfbsterrorを呼び出します。

- プロトタイプ

```
OnFbsterror(VARIANT err_no)
```

- パラメータ

パラメータ	説明
err_no	エラー値です

- 戻り値

戻り値	説明
文字列	関数呼び出しに成功した場合です
NULL	関数呼び出しに失敗した場合です

- 関連関数

OnFbcalcsize()、OnTpalloc()

7.4.35. OnFbtypecvt

指定したフィールド・データを指定したデータ型に変換して返す関数です。内部的にfbtypecvtを呼び出します。

- プロトタイプ

```
OnFbtypecvt(VARIANT *tolen, VARIANT totype, VARIANT fromval, VARIANT fromtype,
            VARIANT fromlen)
```

- パラメータ

パラメータ	説明
tolen	変換されたデータ長です
totype	変換するデータ型です
fromval	変換する元のデータです
fromtype	変換する元のデータ型です
fromlen	変換する元のデータ長です

- 戻り値

戻り値	説明
変換されたデータ・ポインター	関数呼び出しに成功した場合です
NULL	データの変換に失敗した場合です。OnGettperrno()を呼び出してエラー値を取得します

7.4.36. OnFbupdate

フィールド・バッファに指定したフィールド・キーのフィールド順に該当するデータを変更する関数です。内部的にfbupdateを呼び出します。

- プロトタイプ

```
OnFbupdate(VARIANT fbuf, VARIANT fldkey, VARIANT nth, VARIANT value,  
           VARIANT fldlen)
```

- パラメータ

パラメータ	説明
fbuf	OnTpallocまたはOnFballocで割り当てられたバッファのポインターです
fldkey	フィールド・キーです
nth	フィールド順です
value	変更するデータです
fldlen	変更するデータ長です

- 戻り値

戻り値	説明
1	関数呼び出しに成功した場合は
-1	関数呼び出しに失敗した場合は。OnGettperrno()を呼び出してエラー値を取得します

- 関連関数

OnFbdelete()、OnFbinsert()

7.4.37. OnGetfberrno

フィールド・バッファに関連したAPI関数の実行中にエラーが発生した場合、エラー値を返す関数です。内部的にgetfberrnoを呼び出します。

- プロトタイプ

```
OnGetfberrno()
```

- 戻り値

現在fberrorに設定された値を返します。

- 関連関数

OnGetfberror()

7.4.38. OnGetfberror

フィールド・バッファに関連したAPI関数の実行中にエラーが発生した場合、エラー値を返す関数です。内部的にgetfberrorを呼び出します。

- プロトタイプ

```
OnGetfberror( )
```

- 戻り値

現在fberrorに設定された値を返します。

- 関連関数

OnGetfberrno()

7.5. Txインターフェース・メソッド

7.5.1. OnTx_begin

グローバル・トランザクションを開始する関数です。内部的にtx_beginを呼び出します。

- プロトタイプ

```
OnTx_begin( )
```

- 戻り値

戻り値	説明
0	関数呼び出しに成功した場合です
負の数	関数呼び出しに失敗した場合です

- 関連関数

OnTx_commit()、OnTx_rollback()、OnTx_set_transaction_timeout()

7.5.2. OnTx_commit

グローバル・トランザクションをコミットする関数です。内部的にtx_commitを呼び出します。

- プロトタイプ

```
OnTx_commit()
```

- 戻り値

戻り値	説明
0	関数呼び出しに成功した場合です
負の数	関数呼び出しに失敗した場合です

- 関連関数

OnTx_begin()、OnTx_rollback()、OnTx_set_transaction_timeout()

7.5.3. OnTx_rollback

グローバル・トランザクションをロールバックする関数です。内部的にtx_rollbackを呼び出します。

- プロトタイプ

```
OnTx_rollback()
```

- 戻り値

戻り値	説明
0	関数呼び出しに成功した場合です
負の数	関数呼び出しに失敗した場合です

- 関連関数

OnTx_begin()、OnTx_commit()、OnTx_set_transaction_timeout()

7.5.4. OnTx_set_transaction_timeout

グローバル・トランザクションのタイムアウトを設定する関数です。tx_set_transaction_timeoutを呼び出します。

- プロトタイプ

```
OnTx_set_transaction_timeout(VARIANT timeout)
```

- 戻り値

戻り値	説明
0	関数呼び出しに成功した場合です

- 関連関数

OnTx_begin()、OnTx_commit()、OnTx_rollback()

7.6. サンプル・プログラム

アカウント番号をキー値として、支社番号、電話番号、住所を照会、修正、削除、入力するプログラムです。使用されるバッファはフィールド・キー・バッファを使用しました。

7.6.1. プログラムの構成

プログラムは下記のように構成されます。

プログラムファイル	説明
TmaxTestTx.asp	メイン画面です
insert.asp	[INSERT]ボタンが選択されると、Tmaxに要求した後結果を取得してTmaxTestTx.aspで結果値を引き渡します
update.asp	[UPDATE]ボタンが選択されると、Tmaxに要求した後結果を取得してTmaxTestTx.aspで結果値を引き渡します
delete.asp	[DELETE]ボタンが選択されると、Tmaxに要求した後結果を取得してTmaxTestTx.aspで結果値を引き渡します
select.asp	[SELECT]ボタンが選択されると、Tmaxに要求した後結果を取得してTmaxTestTx.aspで結果値を引き渡します

7.6.2. プログラムの特徴

下記はプログラムの特徴です。

- クライアント・プログラム

機能	説明
Tmaxライブラリー接続	Tmaxcom.Atmi、Tmaxcom.Etc、Tmaxcom.Fdl、Tmaxcom.Txを項目に追加します
バッファタイプ	フィールド・キー・バッファ、フィールド・キー・ファイルをfdlcユーティリティでコンパイルして「fdl」ファイルを作成する必要があります
通信タイプ	OnTpcall()を利用して同期通信を行います
トランザクションの有無	照会、修正、削除、入力を実行する場合、すべてトランザクションで処理します
Tmax接続	各サービスを実行するたびに、接続後、サービスを完了する場合に接続を解除します

- サーバー・プログラム

機能	説明
サービス	SELECT、UPDATE、DELETE、INSERTを作成します
データベース指定	Oracleデータベースを使用します。システム構成ファイルのSVRGROUPにデータベース情報を指定します(XA方式)

7.6.3. クライアント・プログラム

メイン画面

プログラムを実行する前に、Windowsの実行画面にて<regsvr32 tmaxcomcli.dll>を実行します。

プログラムを実行すると下記のような画面が表示されます。ボタンは、[INSERT]、[UPDATE]、[DELETE]、[SELECT]で構成されています。

アカウント番号	<input type="text"/>
支社番号	<input type="text"/>
電話番号	<input type="text"/>
住所	<input type="text"/>
結果値	<input type="text"/>
<input type="button" value="INSERT"/> <input type="button" value="UPDATE"/> <input type="button" value="DELETE"/> <input type="button" value="SELECT"/>	

入出力管理

下記は、アカウント番号を利用した入出力管理の例です。

<Tmaxtest.asp>

```

<HTML>
<HEAD>
<TITLE> New Document </TITLE>
<META NAME="Generator" CONTENT="EditPlus">
<META NAME="Author" CONTENT="">
<META NAME="Keywords" CONTENT="">
<META NAME="Description" CONTENT="">
</HEAD>
<%
    Dim retStr
    Dim retAccount
    Dim retBranch
    Dim retPhone
    Dim retAddr
    retStr = request("retStr")
    retAccount = request("retAccount")
    retBranch = request("retBranch")
    retPhone = request("retPhone")
    retAddr = request("retAddr")
%>
<script language="javascript">
    function insertPage(){
        var accountValue = document.form1.taccount_id.value;
        var branchValue = document.form1.tbranch_id.value;
        var phoneValue = document.form1.tphone.value;
        var addrValue = document.form1.taddr.value;
        location.href="insert.asp?taccount_id="+accountValue+"&tbranch_id=
            "+branchValue+"&tphone="+phoneValue+"&taddr="+addrValue;
    }
    function updatePage(){
        var accountValue = document.form1.taccount_id.value;
        var branchValue = document.form1.tbranch_id.value;
        var phoneValue = document.form1.tphone.value;
        var addrValue = document.form1.taddr.value;
        location.href="update.asp?taccount_id="+accountValue+"&tbranch_id=
            "+branchValue+"&tphone="+phoneValue+"&taddr="+addrValue;
    }
    function deletePage(){
        var accountValue = document.form1.taccount_id.value;
        location.href="delete.asp?taccount_id="+accountValue;
    }
    function selectPage(){
        var accountValue = document.form1.taccount_id.value;
        location.href="select.asp?taccount_id="+accountValue;
    }
</script>
<BODY>

```

```

<form action="" name="form1" method="get">
<TABLE>
<TR>
    <TD colspan=2>アカウント番号</TD>
    <TD colspan=2><INPUT TYPE="text" NAME="taccount_id"></TD>
</TR>
<TR>
    <TD colspan=2>支社番号</TD>
    <TD colspan=2><INPUT TYPE="text" NAME="tbranch_id" value="<%if retBranch<
" "
        then response.write retBranch else response.write "" end if %>">

        </TD>
</TR>
<TR>
    <TD colspan=2>電話番号</TD>
    <TD colspan=2><INPUT TYPE="text" NAME="tphone" value="<%if retPhone<
        then response.write retPhone else response.write "" end if %>">
        </TD>
</TR>
<TR>
    <TD colspan=2>住所</TD>
    <TD colspan=2><INPUT TYPE="text" NAME="taddr" value="<%if retAddr<
        then response.write retAddr else response.write "" end if %>"></TD>
</TR>
<TR>
    <TD colspan=2>結果値</TD>
    <TD colspan=2><INPUT TYPE="text" NAME="trval" value="<%if retStr<
        then response.write retStr else response.write "" end if %>"></TD>
</TR>
<TR>
    <TD>
        <INPUT TYPE="button" NAME="insert" value="INSERT" onclick="insertPage();">

    </TD>
    <TD>
        <INPUT TYPE="button" NAME="update" value="UPDATE" onclick="updatePage();">

    </TD>
    <TD>
        <INPUT TYPE="button" NAME="delete" value="DELETE" onclick="deletePage();">

    </TD>
    <TD>
        <INPUT TYPE="button" NAME="select" value="SELECT" onclick="selectPage();">

    </TD>
</TR>

```

```
</TR>
</TABLE></form>

</BODY>
</HTML>
```

[INSERT]ボタンの実行スクリプト

下記は、[INSERT]ボタンの例です。

<insert.asp>

```
<%
    Dim file
    Dim label
    Dim tmp
    'Tmaxと通信するための変数
    Dim fbuf
    'INSERTサービスでfbufに格納した値を読み込むための変数
    Dim retStr
    'TmaxTestTx.aspで取得した値を格納する変数
    Dim taccount_id
    Dim tranch_id
    Dim tphone
    Dim taddr

    Dim rlen
    Dim rVal
    Dim errno

    'TmaxTestTx.aspで取得した値を設定します。
    taccount_id = request("taccount_id")
    tbranch_id = request("tbranch_id")
    tphone = request("tphone")
    taddr = request("taddr")

    'インターフェースのオブジェクトを作成します。
    Set objAtmi = Server.CreateObject("Tmaxcom.Atmi")
    Set objFdl = Server.CreateObject("Tmaxcom.Fdl")
    Set objEtc = Server.CreateObject("Tmaxcom.Etc")
    Set objTx = Server.CreateObject("Tmaxcom.Tx")
    ...
    file = "C:\\Inetpub\\wwwroot\\tmax.env"
    label = "TMAXC1"

    'Tmaxと接続するために、OnTmaxreadenvを利用して環境変数を読み込みます。
```

```

rVal = objEtc.OnTmaxreadenv (file, label)
if rVal < 0 then
    error processing
    ...
end if
'TmaxとOnTpstartで接続します。
tmp = ""
rVal = objAtmi.OnTpstart (tmp, tmp, tmp, tmp, CINT(0))
if rVal < 0 then
    error processing
    ...
end if
'バッファをOnTpallocで割り当てます。
'送信するバッファ/受信するバッファ
fbuf = objAtmi.OnTpalloc ("FIELD", "", CINT(0))
'フィールド・データを取得するバッファ
retStr = objAtmi.OnTpalloc ("STRING", "", CINT(0))

'送信するバッファにフィールド・データを格納します。
'INSERT_ID          ((FLDKEY)201327592) /* number: 1000 type: int */
'BRANCH_ID          ((FLDKEY)201327593) /* number: 1001 type: int */
'PHONE              ((FLDKEY)469763050) /* number: 1002 type: string */
'ADDRESS            ((FLDKEY)469763051) /* number: 1003 type: string */
rVal = objFdl.OnFbput (fbuf, "201327592", taccount_id, CINT("0"))
if rVal < 0 then
    error processing
    ...
end if
rVal = objFdl.OnFbput (fbuf, "201327593", tbranch_id, CINT("0"))
if rVal < 0 then
    error processing
    ...
end if
rVal = objFdl.OnFbput (fbuf, "469763050", tphone, CINT("0"))
if rVal < 0 then
    error processing
    ...
end if
rVal = objFdl.OnFbput (fbuf, "469763051", taddr, CINT("0"))
if rVal < 0 then
    error processing
    ...
end if

'トランザクションを開始します。
rVal = objTx.OnTx_begin()
if rVal < 0 then

```

```

        error processing
        ...
    end if

    'TmaxでINSERTサービスを要求します。
    rVal = objAtmi.OnTpcall ("INSERT", fbuf, CINT(0), fbuf, rlen, CINT("0"))
    if rVal < 0 then
        error processing
        ...
        rVal = objTx.OnTx_rollback()
        if rVal < 0 then
            error processing
            ...
        end if
        'Tmaxとの接続を解除します。
        ...
    else
        rVal = objTx.OnTx_commit()
        if rVal < 0 then
            error processing
            ...
        end if
    end if

    'OnTpcallの結果として取得したfbufからRESULTフィールドの内容を取得します。
    'RESULT ((FLDKEY)469764048) /* number: 2000 type: string */
    rVal = objFdl.OnFbget (fbuf, "469764048", retStr, rlen)
    if rVal < 0 then
        error processing
        ...
    end if

    'RESULTフィールドの内容が存在するバッファから実際の内容を取得します。
    rVal = objAtmi.OnTpget (retStr, tmp)
    if rVal < 0 then
        error processing
        ...
    end if

    'バッファを解除します。
    objAtmi.OnTpfree(fbuf)
    objAtmi.OnTpfree(retStr)

    'Tmaxとの接続を解除します。
    rVal = objAtmi.OnTpend()
    if rVal < 0 then
        error processing
    end if

```



```

...
end if

'オブジェクトを解除します。
Set objAtmi = nothing
Set objFdl = nothing
Set objEtc = nothing
Set objTx = nothing

response.redirect "TmaxTestTx.asp?retStr=" & tmp
%>

```

[UPDATE]ボタンの実行スクリプト

下記は、[UPDATE]ボタンの例です。

<update.asp>

```

<%
    Dim file
    Dim label
    Dim tmp
    'Tmaxと通信するための変数
    Dim fbuf
    'UPDATEサービスでfbufに格納した値を読み込むための変数
    Dim retStr
    'TmaxTestTx.aspで取得した値を格納する変数
    Dim taccount_id
    Dim tranch_id
    Dim tphone
    Dim taddr

    Dim rlen
    Dim rVal
    Dim errno

    'TmaxTestTx.aspで取得した値を設定します。
    taccount_id = request("taccount_id")
    tbranch_id = request("tbranch_id")
    tphone = request("tphone")
    taddr = request("taddr")

    'インターフェースのオブジェクトを作成します。
    Set objAtmi = Server.CreateObject("Tmaxcom.Atmi")
    Set objFdl = Server.CreateObject("Tmaxcom.Fdl")
    Set objEtc = Server.CreateObject("Tmaxcom.Etc")
    Set objTx = Server.CreateObject("Tmaxcom.Tx")

```

```

%>
<script language="javascript">
    alert ("taccount_id[<%=taccount_id%>] tbranch_id[<%=tbranch_id%>]
           thphone[<%=tphone%>] taddr[<%=taddr%>]");
</script>
<%
    file = "C:\\Inetpub\\wwwroot\\tmax.env"
    label = "TMAXC1"

    'Tmaxと接続するために、OnTmaxreadenvを利用して環境変数を読み込みます。
    rVal = objEtc.OnTmaxreadenv (file, label)
    if rVal < 0 then
        error processing
        ...
    end if
    'TmaxとOnTpstartで接続します。
    tmp = ""
    rVal = objAtmi.OnTpstart (tmp, tmp, tmp, tmp, CINT(0))
    if rVal < 0 then
        error processing
        ...
    end if
    'バッファをOnTpallocで割り当てます。
    '送信するバッファ/受信するバッファ
    fbuf = objAtmi.OnTpalloc ("FIELD", "", CINT(0))
    'フィールド・データを取得するバッファ
    retStr = objAtmi.OnTpalloc ("STRING", "", CINT(0))

    '送信するバッファにフィールド・データを格納します。
    'UPDATE_ID ((FLDKEY)201327596) /* number: 1004 type: int */
    'BRANCH_ID ((FLDKEY)201327593) /* number: 1001 type: int */
    'PHONE ((FLDKEY)469763050) /* number: 1002 type: string */
    'ADDRESS ((FLDKEY)469763051) /* number: 1003 type: string */
    rVal = objFdl.OnFbput(fbuf, "201327596", taccount_id, CINT("0"))
    if rVal < 0 then
        error processing
        ...
    end if
    rVal = objFdl.OnFbput(fbuf, "201327593", tbranch_id, CINT("0"))
    if rVal < 0 then
        error processing
        ...
    end if
    rVal = objFdl.OnFbput(fbuf, "469763050", tphone, CINT("0"))
    if rVal < 0 then
        error processing
        ...

```

```

end if
rVal = objFdl.OnFbput(fbuf, "469763051", taddr, CINT("0"))
if rVal < 0 then
    error processing
    ...
end if

'トランザクションを開始します。
rVal = objTx.OnTx_begin()
if rVal < 0 then
    error processing
    ...
end if

'TmaxでUPDATEサービスを要求します。
rVal = objAtmi.OnTpcall("UPDATE", fbuf, CINT(0), fbuf, rlen, CINT("0"))
if rVal < 0 then
    error processing
    ...
    if rVal < 0 then
        errno = objAtmi.OnGettperrno()
        response.write("OnTx_rollback fail["&rVal&"] ["&errno&"]")
    end if
    ...
else
    rVal = objTx.OnTx_commit()
    if rVal < 0 then
        error processing
        ...
    end if
end if

'OnTpcallの結果として取得したfbufからRESULTフィールドの内容を取得します。
'RESULT ((FLDKEY)469764048) /* number: 2000 type: string */
rVal = objFdl.OnFbget(fbuf, "469764048", retStr, rlen)
if rVal < 0 then
    error processing
    ...
end if

'RESULTフィールドの内容が存在するバッファから実際の内容を取得します。
rVal = objAtmi.OnTpget (retStr, tmp)
if rVal < 0 then
    error processing
    ...
end if

```

```

'バッファを解除します。
objAtmi.OnTpfree(fbuf)
objAtmi.OnTpfree(retStr)

'Tmaxとの接続を解除します。
rVal = objAtmi.OnTpend()
if rVal < 0 then
    error processing
    ...
end if

'オブジェクトを解除します。
Set objAtmi = nothing
Set objFdl = nothing
Set objEtc = nothing
Set objTx = nothing

response.redirect "TmaxTestTx.asp?retStr=" & tmp
%>

```

[DELETE]ボタンの実行スクリプト

下記は、[DELETE]ボタンの例です。

<delete.asp>

```

<%
    Dim file
    Dim label
    Dim tmp
    'Tmaxと通信するための変数
    Dim fbuf
    'DELETEサービスでfbufに格納した値を読み込むための変数
    Dim retStr
    'TmaxTestTx.aspで取得した値を格納する変数
    Dim taccount_id

    Dim rlen
    Dim rVal
    Dim errno

    'TmaxTestTx.aspで取得した値を設定します。
    taccount_id = request("taccount_id")

    'インターフェースのオブジェクトを作成します。
    Set objAtmi = Server.CreateObject("Tmaxcom.Atmi")

```

```

Set objFdl = Server.CreateObject("Tmaxcom.Fdl")
Set objEtc = Server.CreateObject("Tmaxcom.Etc")
Set objTx = Server.CreateObject("Tmaxcom.Tx")
%>
<script language="javascript">
    alert("taccount_id[<%=taccount_id%>]");
</script>
<%
    file = "C:\\Inetpub\\wwwroot\\tmax.env"
    label = "TMAXC1"

    'Tmaxと接続するために、OnTmaxreadenvを利用して環境変数を読み込みます。
    rVal = objEtc.OnTmaxreadenv (file, label)
    if rVal < 0 then
        error processing
        ...
    end if
    'TmaxとOnTpstartで接続します。
    tmp = ""
    rVal = objAtmi.OnTpstart (tmp, tmp, tmp, tmp, CINT(0))
    if rVal < 0 then
        error processing
        ...
    end if
    'バッファをOnTpallocで割り当てます。
    '送信するバッファ/受信するバッファ
    fbuf = objAtmi.OnTpalloc ("FIELD", "", CINT(0))
    'フィールド・データを取得するバッファ
    retStr = objAtmi.OnTpalloc ("STRING", "", CINT(0))

    '送信するバッファにフィールド・データを格納します。
    'DELETE_ID ((FLDKEY)201327597) /* number: 1005 type: int */
    rVal = objFdl.OnFbput (fbuf, "201327597", taccount_id, CINT("0"))
    if rVal < 0 then
        error processing
        ...
    end if

    'トランザクションを開始します。
    rVal = objTx.OnTx_begin()
    if rVal < 0 then
        error processing
        ...
    end if

    'TmaxでINSERTサービスを要求します。
    rVal = objAtmi.OnTpcall ("DELETE", fbuf, CINT(0), fbuf, rlen, CINT("0"))

```

```

if rVal < 0 then
    error processing
    ...
    rVal = objTx.OnTx_rollback()
    if rVal < 0 then
        errno = objAtmi.OnGettperrno()
        response.write("OnTx_rollback fail["&rVal&"] ["&errno&"]")
    end if
    ...
else
    rVal = objTx.OnTx_commit()
    if rVal < 0 then
        error processing
        ...
    end if
end if

'OnTpcallの結果として取得したfbufからRESULTフィールドの内容を取得します。
'RESULT ((FLDKEY)469764048) /* number: 2000 type: string */
rVal = objFdl.OnFbget (fbuf, "469764048", retStr, rlen)
if rVal < 0 then
    error processing
    ...
end if

'RESULTフィールドの内容が存在するバッファから実際の内容を取得します。
rVal = objAtmi.OnTpget (retStr, tmp)
if rVal < 0 then
    error processing
    ...
end if

'バッファを解除します。
objAtmi.OnTpfree(fbuf)
objAtmi.OnTpfree(retStr)

'Tmaxとの接続を解除します。
rVal = objAtmi.OnTpend()
if rVal < 0 then
    error processing
    ...
end if

'オブジェクトを解除します。
Set objAtmi = nothing
Set objFdl = nothing
Set objEtc = nothing

```

```

        Set objTx = nothing

        response.redirect "TmaxTestTx.asp?retStr=" & tmp
    %>

```

[SELECT]ボタンの実行スクリプト

下記は、[SELECT]ボタンの例です。

<select.asp>

```

<%
    Dim file
    Dim label
    Dim tmp
    'Tmaxと通信するための変数
    Dim fbuf
    'DELETEサービスでfbufに格納した値を読み込むための変数
    Dim str
    Dim branch
    Dim phone
    Dim addr
    'TmaxTestTx.aspで取得した値を格納する変数
    Dim taccount_id
    'TmaxTestTx.aspに値を引き渡すための変数
    Dim retStr
    Dim retBranch
    Dim retPhone
    Dim retAddr

    Dim rlen
    Dim rVal
    Dim errno

    'TmaxTestTx.aspで取得した値を設定します。
    taccount_id = request("taccount_id")

    'インターフェースのオブジェクトを作成します。
    Set objAtmi = Server.CreateObject("Tmaxcom.Atmi")
    Set objFdl = Server.CreateObject("Tmaxcom.Fdl")
    Set objEtc = Server.CreateObject("Tmaxcom.Etc")
    Set objTx = Server.CreateObject("Tmaxcom.Tx")
%>
<script language="javascript">
    alert("taccount_id[<%=taccount_id%>]");
</script>

```

```

<%
file = "C:\\Inetpub\\wwwroot\\tmax.env"
label = "TMAXC1"

'Tmaxと接続するために、OnTmaxreadenvを利用して環境変数を読み込みます。
rVal = objEtc.OnTmaxreadenv (file, label)
if rVal < 0 then
    error processing
    ...
end if
'TmaxとOnTpstartで接続します。
tmp = ""
rVal = objAtmi.OnTpstart (tmp, tmp, tmp, tmp, CINT(0))
if rVal < 0 then
    error processing
    ...
end if
'バッファをOnTpallocで割り当てます。
'送信するバッファ/受信するバッファ
fbuf = objAtmi.OnTpalloc ("FIELD", "", CINT(0))
'フィールド・データを取得するバッファ
str = objAtmi.OnTpalloc ("STRING", "", CINT(0))
branch = objAtmi.OnTpalloc ("STRING", "", CINT(0))
phone = objAtmi.OnTpalloc ("STRING", "", CINT(0))
addr = objAtmi.OnTpalloc ("STRING", "", CINT(0))

'送信するバッファにフィールド・データを格納します。
'SELECT_ID ((FLDKEY)201327598) /* number: 1006 type: int */
rVal = objFdl.OnFbput (fbuf, "201327598", taccount_id, CINT("0"))
if rVal < 0 then
    error processing
    ...
end if

'トランザクションを開始します。
rVal = objTx.OnTx_begin()
if rVal < 0 then
    error processing
    ...
end if

'TmaxでSELECTサービスを要求します。
rVal = objAtmi.OnTpcall ("SELECT", fbuf, CINT(0), fbuf, rlen, CINT("0"))
if rVal < 0 then
    error processing
    ...
    rVal = objTx.OnTx_rollback()

```



```

    if rVal < 0 then
        errno = objAtmi.OnGettperrno()
        response.write("OnTx_rollback fail[" &rVal&"] ["&errno&"]")
    end if
    ...
else
    rVal = objTx.OnTx_commit()
    if rVal < 0 then
        error processing
        ...
    end if
end if

'OnTpcallの結果として取得したfbufからRESULTフィールドの内容を取得します。
'RESULT ((FLDKEY)469764048) /* number: 2000 type: string */
rVal = objFdl.OnFbget (fbuf, "469764048", str, rlen)
if rVal < 0 then
    error processing
    ...
end if

'RESULTフィールドの内容が存在するバッファから実際の内容を取得します。
rVal = objAtmi.OnTpget (str, retStr)
if rVal < 0 then
    error processing
    ...
end if

'BRANCH_ID ((FLDKEY)201327593) /* number: 1001 type: int */
'PHONE ((FLDKEY)469763050) /* number: 1002 type: string */
'ADDRESS ((FLDKEY)469763051) /* number: 1003 type: string */
rVal = objFdl.OnFbget (fbuf, "201327593", branch, rlen)
rVal = objAtmi.OnTpget(branch, retBranch)
rVal = objFdl.OnFbget (fbuf, "469763050", phone, rlen)
rVal = objAtmi.OnTpget(phone, retPhone)
rVal = objFdl.OnFbget (fbuf, "469763051", addr, rlen)
rVal = objAtmi.OnTpget(addr, retAddr)

'バッファを解除します。
objAtmi.OnTpfree(fbuf)
objAtmi.OnTpfree(str)
objAtmi.OnTpfree(branch)
objAtmi.OnTpfree(phone)
objAtmi.OnTpfree(addr)

'Tmaxとの接続を解除します。
rVal = objAtmi.OnTpend()
if rVal < 0 then
    error processing

```

```
...
end if

'オブジェクトを解除します。
Set objAtmi = nothing
Set objFdl = nothing
Set objEtc = nothing
Set objTx = nothing

response.redirect "TmaxTestTx.asp?retStr=" & retStr & "&retBranch=" & retBranch & "&
retPhone=" & retPhone & "&retAddr=" & retAddr

%>
```

第8章 PHPインターフェース

本章では、PHPでTmaxクライアントの機能が使用できる拡張モジュールの使用方法と例について説明します。

8.1. 使用方法

PHPでは、Cライブラリーを呼び出せる拡張モジュールの作成方法を提供しています。

Tmaxは、PHP開発者がTmaxクライアントの機能を使用できるように、clientライブラリーの関数を呼び出せる拡張モジュールを作成および使用する方法を提供します。

8.1.1. 開発環境

以下は、拡張モジュールをビルドできる環境です。

1. 4.0.0以上のphp-develがインストールされている必要があります。
2. システム環境変数のPHP_TMAX_LIB_BITSを、32ビットでは「32」、64ビットでは「64」に設定します。

8.1.2. 拡張モジュールの作成手順

以下は、拡張モジュールを作成する手順です。

1. Tmaxディレクトリー配下のusrinc/webscript/php_tmaxディレクトリーをphp拡張ディレクトリー内にコピーします。このディレクトリーは通常、phpソースの場所の配下のextディレクトリーです。

例: /php-5.5.8/ext

2. extディレクトリー配下のphp_tmaxディレクトリーに移動した後、phpizeを実行します。configureファイルが作成されます。
3. ./configureを実行します。メイクファイルが作成されます。
4. makeを実行します。配下にあるmodulesディレクトリー配下にphp_tmax.soファイルが作成されます。

5. php_tmax.soファイルをphp拡張モジュールが存在するディレクトリーにコピーします。このディレクトリーは通常、/usr/lib/php/modules/に設定され、環境設定ファイルの /etc/php.iniで「extension_dir=」の値を設定して指定することができます。

8.1.3. 拡張モジュールの使用方法

作成された拡張モジュールは、以下のように使用します。

1. Tmaxディレクトリー配下のusrinc/webscript/ の下にあるtmax.phpファイルを、ユーザーが実行するphpファイルと同じディレクトリーに置きます。
2. phpファイルに次のステートメントを含めます。

```
include "tmax.php";  
dl("php_tmax.so");
```

dl問合せの結果はTRUEである必要があります。FALSEである場合、php_tmax.soのロードに失敗したことを意味します。

使用可能な関数

PHP拡張モジュールにより使用可能な関数は以下のとおりです。

PHP関数	対応するTmaxクライアント関数
int php_tmaxreadenv(string file, string label)	int tmaxreadenv (char *file, char *label)
string php_tpgetenv(string name)	char *tpgetenv(char *name)
int php_tpputenv(string env_string)	int tpputenv(char *string)
int php_gettperrno()	int gettperrno(void)
int php_gettpurcode()	long gettpurcode(void)
string php_tpstrerror(int tperrno)	char *tpstrerror (int tperrno)
int php_tpstart(Tpstart tpinfo)	int tpstart (TPSTART_T *tpinfo)
int php_tpend()	int tpend (void)
CallResult php_tpcall(string svcname, string idata, int flags)	int tpcall (char *svc, char *idata, long ilen, char **odata, long *olen, long flags)
int php_tpacall(string svcname, string idata, int flags)	int tpacall (char *svc, char *data, long len, long flags)
CallResult php_tpgetrply(int cd, int flags)	int tpgetrply(int *cd, char **data, long *len, long flags)

PHP関数	対応するTmaxクライアント関数
int php_tpcancel(int cd)	int tpcancel (int cd)
int php_tpset_timeout(int sec)	int tpset_timeout (int sec)
int php_tpget_timeout()	int tpget_timeout(void)
int php_tx_begin()	int tpget_timeout(void)
int php_tx_commit()	int tx_commit(void)
int php_tx_rollback()	int tx_rollback(void)
int php_tx_set_transaction_timeout(int timeout)	int tx_rollback(void)
int php_tx_set_transaction_control(int control)	i n t tx_set_transaction_control(TRANSACTION_CONTROL control)
int php_tx_set_commit_return(int when_return)	int tx_set_commit_return (COMMIT_RETURN when_return)
TxInfo php_tx_info()	int tx_info (TXINFO *info)

ほとんどの関数は、接頭辞「php_」を抜いた同名のTmaxクライアント関数と使用方法が同じですが、次の点に注意が必要です。

- php_tpcall、php_tpacallはSTRING型データのみ処理するので、ilenパラメータを受ける必要がありません。
- php_tpstartの引数として使用されるクラスTpstart型のデータはtmax.phplに定義されており、各フィールドの意味はtpinfoと同じです。
- php_tpcall、php_tpgetrplyの戻り値はクラスCallResult型のデータです。各フィールドの意味は以下のとおりです。

区分	説明
ret	tpcall、tpgetrplyの戻り値です
rcvbuf	受信したデータです
rcvlen	受信したデータの長さです
tperrno	エラーの発生時に設定されたtperrnoの値です
tpurcode	urcodeを設定時に指定されたurcodeの値です

- php_tx_info()の戻り値はクラスTxInfo型のデータです。このデータは、when_return、transaction_control、transaction_timeout、transaction_stateフィールドを持ちます。各フィールドの意味は /usrinc/tx.hのTXINFOと同じです。

8.1.4. サンプル・プログラム

以下は、アカウント番号による入出力管理の例です。

<php_tmax_test.php>

```
<html>
<head>
<title>memman test</title>
</head>
<body>
new page <p/>
<?php
    echo '<p>Hello World</p>';
    include "tmax.php";
    $a = new TpstartT;
    $b = new TpstartT;
    $a -> username = "tmax";
    $a -> dompwd = "tmax1234";
    $a -> usrpwd = "tmax1234";
    $a -> cltname = "TP1";
    $a -> flags = TPNOFLAGS;

    if (!dl("php_tmax.so")) {
        echo "<p>\n\n failures detected in \"tmaxreadenv\" \n\n</p>";
        exit;
    }

    else {
        echo '<p>success</p>'.php_tmaxreadenv("tmax.env","DOM1_NA");
    }
    $ret = php_tpstart(NULL);
    if($ret < 0) {
        echo "<p>\n\n failures detected in \"tpstart\" \n\n</p>";
        exit;
    }
    $errno = php_gettperrno();
    echo "<p>tperrstr : ".php_tpstrerror($errno)."</p>";
    echo "<p>tpset_timeout : ".php_tpset_timeout(60)."</p>";
    echo "<p>tpget_timeout : ".php_tpget_timeout()."</p>";
    echo "<p>tx_begin : ". php_tx_begin() ."</p>";
    echo "<p>tx_set_transaction_timeout : ".php_tx_set_transaction_timeout(50).
"</p>";
    echo "<p>tx_set_transaction_control :
.php_tx_set_transaction_control(TX_CHAINED). "</p>";
    echo "<p>tx_set_commit_return : ".php_tx_set_commit_return(TX_COMMIT_COMPLETED).
"</p>";
```

```

$retval = php_tpcall("TOUPPER", "bbb", 0);
if($retval -> rcvbuf < 0) {
    echo "<p>\n failures detected in \"tpcall\" \n</p>";
    exit;
}
$cd1 = php_tpacall("TOUPPER","aaa",0);
if($cd1 < 0) {
    echo "<p>\n failures detected in \"tpacall_1\" \n</p>";
    exit;
}
$cd2 = php_tpacall("TOUPPER","aaa",0);
if($cd2 < 0) {
    echo "<p>\n failures detected in \"tpacall_2\" \n</p>";
    exit;
}
$ret = php_tpcancel($cd1);
if($cd1 < 0) {
    echo "<p>\n failures detected in \"tpcancel\" \n</p>";
    exit;
}
$retval1 = php_tpgetrply($cd1, 0);
if($retval1 -> ret < 0) {
    echo "<p>\n failures detected in \"tpgetrply_1\" \n</p>";
    exit;
}
var_dump($retval1);
$retval2 = php_tpgetrply($cd2, 0);
if($retval2 -> ret < 0) {
    echo "<p>\n failures detected in \"tpgetrply_2\" \n</p>";
    exit;
}
var_dump($retval1);
echo "<p>tpgetrply : ". $retval2 -> rcvbuf . "</p>";
$ret = php_tx_commit();
if($ret < 0) {
    echo "<p>\n failures detected in \"tx_commit(\".$ret.\")\" \n</p>";
    exit;
}
$retval = php_tpcall("TOERROR", "bbb", 0);
var_dump($retval);
$ret = php_gettpurcode();
if($ret != 2) {
    echo "<p>\n failures detected in \"gettpurcode\" \n</p>";
    exit;
}
$ret = php_tpend();
if($ret < 0) {

```

```
        echo "<p>\n1 failures detected in \"tpend\"\\n</p>";
        exit;
    }
    echo "<p>\n";
    echo "No errors detected in \"php_tmax_test\"";
    echo "\\n</p>";
?>
</body>
</html>
```


索引

A

ASPインターフェース, 365

C

C# .netインターフェース, 331

D

Delphiインターフェース, 165

E

ErrorMsg, 201, 267

F

f_data, 4

f_datadel, 5

f_fdata, 6

f_fdatadel, 7

f_form, 8

f_getdata, 9

FdlErrorMsg, 114, 202, 268

FilltpstartBuf, 115, 203, 269

G

GETCAR, 116, 204, 270

GETCAR2, 205, 271

GETCAR3, 206, 272

GETCAR_BA, 117

GETCHR, 119, 207, 273

GETDOUBLE, 120, 208, 274

GETFLOAT, 121, 209, 275

GETINT, 122, 210, 276

GETLONG, 123, 211, 277

GETSHORT, 212, 278

GETSTR, 123

GETVAR, 124, 213, 279

I

isblocked, 10

O

OnFballoc, 393

OnFbcalcsiz, 394

OnFbdelall, 394

OnFbdelete, 395

OnFbfldcount, 396

OnFbfree, 396

OnFbget, 397

OnFbget_fbsize, 397

OnFbget_fldkey, 398

OnFbget_fldname, 398

OnFbget_fldtype, 399

OnFbget_strfldtype, 400

OnFbget_unused, 400

OnFbget_used, 401

OnFbgetc, 401

OnFbgetf, 403

OnFbgetnth, 403

OnFbgetntht, 404

OnFbgetval, 405

OnFbgetvali, 406

OnFbgetvals, 407

OnFbgetvalt, 407

OnFbinit, 408

OnFbinsert, 409

OnFbisfbuf, 410

OnFbispres, 410

OnFbkeynm_unload, 411

OnFbkeyoccur, 411

OnFbmake_fldkey, 412

OnFbnmkey_unload, 413

OnFbput, 413

OnFbputt, 414

OnFbrealloc, 415

OnFbsterror, 415

OnFbtypecvt, 416

OnFbupdate, 417

OnGetfberrno, 417

OnGetfberror, 418

OnGettperrno, 369
OnGettpurcode, 370
OnTmax_chk_conn, 382
OnTmaxreadenv, 382
OnTp_sleep, 383
OnTp_usleep, 383
OnTpacall, 370
OnTpalloc, 371
OnTpcall, 371
OnTpcancel, 372
OnTpconnect, 373
OnTpdeq, 384
OnTpdiscon, 374
OnTpend, 374
OnTpenq, 385
OnTperrordetail, 386
OnTpextsvcinfo, 386
OnTpextsvcname, 387
OnTpfree, 375
OnTpget, 375
OnTpgetenv, 387
OnTpgetrply, 376
OnTpput, 377
OnTpputenv, 388
OnTpqstat, 389
OnTpqsvcstat, 389
OnTprealloc, 378
OnTprecv, 378
OnTpreissue, 390
OnTpreset, 391
OnTpsend, 379
OnTpset_timeout, 391
OnTpstart, 380
OnTpsubqname, 392
OnTptobackup, 393
OnTptypes, 381
OnTx_begin, 418
OnTx_commit, 419
OnTx_rollback, 419
OnTx_set_transaction_timeout, 419

P

pb_etpcall, 11
pb_getunsold, 13
pb_IsBlocked, 14
pb_reset, 15
pb_tmaxreadenv, 16
pb_tpacall, 18
pb_tpalloc, 19
pb_tpbroadcast, 21
pb_tpcall, 22
pb_tpcallw, 25
pb_tpcancel, 27
pb_tpcommit, 29
pb_tpconnect, 30
pb_tpconv, 32
pb_tpdiscon, 33
pb_tpend, 34
pb_tpfcall, 35
pb_tpfree, 37
pb_tpget, 38
pb_tpgetrply, 39
pb_tpgetunsol, 41
pb_tpput, 42
pb_tprecv, 43
pb_tpreset, 46
pb_tpsend, 47
pb_tpset_timeout, 49
pb_tpsetunsol, 51
pb_tpstart, 52
pb_tptobackup, 54
pb_tx_begin, 55
pb_tx_commit, 56
pb_tx_info, 62
pb_tx_rollback, 57
pb_tx_set_commit_return, 58
pb_tx_set_transaction_control, 59
pb_tx_set_transaction_timeout, 60
pb_uotmax_ver, 61
PHPの拡張モジュール, 437
PowerBuilder, 1
PUTCAR, 125, 214, 280
PUTCAR2, 215, 281

PUTCAR3, 216, 282
PUTCAR_BA, 126
PUTCHR, 128, 218, 284
PUTDOUBLE, 129, 219, 285
PUTFLOAT, 130, 220, 286
PUTINT, 131, 221, 287
PUTLONG, 132, 222, 288
PUTSHORT, 223, 289
PUTSTR, 133
PUTVAR, 134, 224, 290

T

TmaxError, 135
tp_abort, 63
tp_acall, 64
tp_begin, 65
tp_broadcast, 66
tp_call, 67
tp_cancel, 69
tp_commit, 70
tp_connect, 71
tp_conv, 73
tp_discon, 75
tp_fcall, 77
tp_getrply, 78
tp_init, 79
tp_recv, 80
tp_send, 83
tp_subscribe, 85
tp_term, 86
tp_unsubscrib, 86
tpbegin, 87
tuxcall, 88
tuxedo_compat, 90
tuxreadenv, 92

V

Visual Basic .netインターフェース, 199, 265
Visual Basicインターフェース, 113
Visual Basicインターフェース・モジュール, 113

は

ブロッキング状態, 14

