

Tmax COBOLユーザガイド

Tmax v6.0



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

Copyright Notice

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

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

Restricted Rights Legend

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

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

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

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

Trademarks

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

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

Open Source Software Notice

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

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

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

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

文書情報

文書名: Tmax COBOLユーザガイド

発行日: 2016年8月5日

ソフトウェアバージョン: Tmax v6.0

ガイドバージョン: v2.1.1

目次

このガイドについて	vii
第1章 環境設定	1
1.1. ディレクトリー構造	1
1.2. 環境変数の設定	2
1.3. 環境ファイルの設定	2
1.4. Sampleファイルのインストール	4
第2章 プログラムの作成	7
2.1. データの格納	7
2.1.1. TPSTATUS	7
2.1.2. TPTYPE	8
2.1.3. TPSVCDEF	9
2.1.4. TPINFDEF	10
2.1.5. TPSVCRET	10
2.1.6. TPTRXDEF	11
2.1.7. TPCMTDEF	11
2.1.8. TPAUTDEF	11
2.1.9. TPPRIDEF	11
2.1.10. TPTRXLEV	12
2.1.11. TPBCTDEF	12
2.1.12. FDL-INFO	12
2.1.13. TPEVTDEF	13
2.1.14. TXSTATUS	14
2.1.15. TXINFDEF	15
2.2. 関数	16
2.2.1. FINIT	17
2.2.2. FVFTOS	18
2.2.3. FVSTOF	19
2.2.4. TP_SLEEP	21
2.2.5. TPACALL	22
2.2.6. TPCALL	24
2.2.7. TPCANCEL	28
2.2.8. TPFORWARD	29
2.2.9. TPGETLEV	31
2.2.10. TPGETRPLY	31
2.2.11. TPRETURN	34
2.2.12. TPSVCSTART	36
2.2.13. TPSVRDONE	37
2.2.14. TPSVRINIT	38
2.2.15. TXBEGIN	39
2.2.16. TXCOMMIT	40

2.2.17.	TXINFORM	42
2.2.18.	TXROLLBACK	43
2.2.19.	TXSETCOMMITRET	45
2.2.20.	TXSETTIMEOUT	47
2.2.21.	TXSETTRANCTL	48
第3章 例		51
3.1.	サービス・プログラム	51
3.1.1.	サンプル・プログラム	51
3.1.2.	Makefile	56
3.1.3.	サービスのコンパイルおよび起動	59
3.1.4.	サービスの呼び出しテスト	59
3.2.	FDLバッファ使用プログラム	60
3.2.1.	FDLの作成	60
3.2.2.	ビュー・ファイルの作成	61
3.2.3.	COBOL用のビューファイルの作成	61
3.2.4.	サンプル・プログラム	63
3.2.5.	サービスの呼び出しテスト	71
3.3.	グローバル・トランザクションのプログラム	72
3.3.1.	サンプル・プログラム	72
3.3.2.	サービスのコンパイルおよび起動	74
3.3.3.	サービスの呼び出しテスト	75
3.4.	TPSVRINIT/TPSVRDONEの使用	75
3.4.1.	TPSVRINIT	75
3.4.2.	TPSVRDONE	77
3.4.3.	サービスのコンパイルおよび起動	78
3.4.4.	サービスの呼び出しテスト	78
3.5.	サービスでのサービス呼び出しプログラム	79
3.5.1.	サンプル・プログラム	79
3.5.2.	サービスのコンパイルおよび起動	81
3.5.3.	サービスの呼び出しテスト	82
索引		83

このガイドについて

対象読者

本書は、Tmax[®](以下、Tmax)システムでCOBOLを使用してユーザー・プログラムを作成する開発者を対象としています。COBOLでプログラムを開発するための基本環境設定とCOBOLで作成されたサーバー・プログラムの構造とサーバー用のMakefile、およびサーバー・プログラムを起動、テスト、終了する方法について説明します。

前提知識

本章を読む前に、Tmaxの基本的な概念について熟知している必要があります。Tmaxエンジンは事前にインストールされていると想定しています。

ほとんどの例がOracleデータベースを使用しており、Oracleで提供するプリコンパイラーが32ビット版のみであるため、Tmaxエンジンも32ビット版を使用します。64ビット版のTmaxエンジンを使用する場合、実行ファイルにはsdlcの代わりにsdlc32を、64ビット版ライブラリーの代わりに32ビット版ライブラリーを使用してください。同書における例はTmax 3.8.15の32ビット・エンジンをベースにして作成しています。

本書では、作成されたサービス・プログラムの動作をテストするため、クライアント・プログラムを作成せずに、Tmaxが提供するtmdユーティリティを使用しています。tmdは、環境ファイルに指定された環境変数のTMAX_HOST_ADDRとTMAX_HOST_PORTを使用してTmaxに接続するので、環境ファイルにこの2つの環境変数を必ず設定してください。

制限事項

Tmaxシステムの基本的な用語(たとえば、ドメイン、ノード、サーバー・グループ、サーバー、サービスなど)は既に理解していると想定しています。

参考

Tmaxシステムの開発についての基本的な内容は、『Tmax 運用ガイド』および『Tmax アプリケーション開発ガイド』を、Tmaxが提供するコマンドとC APIについては、『Tmax リファレンスガイド』を参照してください。

本書の構成

本書は、計3章で構成されています。

各章の主な内容は以下のとおりです。

- 第1章: 環境設定

COBOLでプログラムを開発するための環境設定について説明します。

- 第2章: プログラムの作成

データを保存する構造体と関数について説明します。

- 第3章: 例

サービス・プログラム、グローバル・トランザクション・プログラムなどのサンプル・プログラムについて説明します。

表記上の規則

表記	意味
<AaBbCc123>	プログラム・ソースコードのファイル名
<Ctrl>+C	CtrlとC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、コマンド実行後の画面に出力された結果物、サンプル・コード

システム要件

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

関連文書

ガイド	説明
Tmax インストールガイド	Tmaxのインストール手順について説明しています
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 リファレンスガイド	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 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 環境設定

本章では、COBOLでプログラムを開発するための環境設定方法について説明します。

1.1. ディレクトリー構造

以下は、COBOL関連のディレクトリーについての説明です。

```
$TMAXDIR
|--lib
|--cobinc
|--config
|--ucblinc
+--sample
    |--cobserver
    |--cobclient
```

lib

COBOLライブラリー・ファイルが格納されます。

cobinc

COBOLヘッダーである*.cblファイルが格納されます。

config

COBOL sampleをテストできるTmax環境ファイルが格納されます。

ucblinc

フィールド・バッファーを使用するsampleのために必要なfield key、view fileなどが存在します。

sample

フォルダ	説明
cobserver	サービスの格納パスです
cobclient	クライアントの格納パスです

参考

Tmaxエンジンのインストールの詳細については『Tmax インストールガイド』を参照してください。

COBOLライブラリー・ファイル

\$TMAXDIR/libにはCOBOL用のライブラリー・ファイルが存在する必要があります。

```
$ ls -l $TMAXDIR/lib/libcbl*  
-rwx-----1 tmax dba 252461 4月22日 12:53 /user/tmax3815/lib/libcblc.so*  
-rw-----1 tmax dba 351420 4月22日 12:52 /user/tmax3815/lib/libcbls.a
```

1.2. 環境変数の設定

TmaxシステムにおいてCOBOLでサービスを作成するためには、Tmaxのための環境設定、COBOLを使用するための環境設定が必要です。

```
##### Tmax system Environment  
  
TMAXDIR=<Tmaxがインストールされたパス>  
TMAX_HOST_ADDR=<TmaxがインストールされたマシンのIPアドレス>  
TMAX_HOST_PORT=<Tmaxシステムが使用するポート>  
SDLFILE=$TMAXDIR/ucblinc/tmax.sdl  
FDLFILE=$TMAXDIR/sample/fdl/tmax.fdl  
PATH=$TMAXDIR/bin:$PATH  
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$TMAXDIR/lib:$TMAXDIR/tuxlib  
export TMAXDIR TMAX_HOST_ADDR TMAX_HOST_PORT  
export SDLFILE FDLFILE PATH LD_LIBRARY_PATH  
  
##### MF Cobol Environment  
  
COBCPY=$TMAXDIR/cobinc:$TMAXDIR/ucblinc  
COBDIR=<Cobol compilerがインストールされたディレクトリーのパス>  
PATH=$PATH$COBDIR/bin  
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$COBDIR/lib  
export COBCPY COBDIR PATH LD_LIBRARY_PATH
```

注

LD_LIBRARY_PATHは、IBM AIXでは「LIBPATH」で、UP-UXでは「SHLIB_PATH」で置き換えます。

1.3. 環境ファイルの設定

以下は、Tmax環境ファイルの作成手順です。

1. 環境ファイルを作成します。

```
*DOMAIN  
tmax1 SHMKEY =99990, TPORTNO=9389, BLOCKTIME=60,
```

```

MAXSVC=100, MAXSVR=50, MAXSPR=100, MAXTMS=10, IPCPERM=0666

*NODE
tmax5    TMAXDIR = "/user/tmax3815",
          APPDIR  = "/user/tmax3815/appbin",
          PATHDIR = "/user/tmax3815/path",
          TLOGDIR = "/user/tmax3815/log/tlog",
          ULOGDIR = "/user/tmax3815/log/ulog",
          SLOGDIR = "/user/tmax3815/log/slog"

*SVRGROUP
svg1     NODENAME = "edu25"

### tms for Oracle ###
svg2     NODENAME = "edu25",      DBNAME = ORACLE,
          OPENINFO = "Oracle_XA+Acc=P/scott/tiger+SesTm=60 ",
          TMSNAME  = tms_ora

*SERVER
carray_nxa    SVGNAME = svg1, MIN=1, MAX=1,
               CLOPT="-e $(SVR).err -o $(SVR).out -- scott tiger"
svctpcall     SVGNAME = svg1, MIN=1, MAX=1,
               CLOPT="-e $(SVR).err -o $(SVR).out"
fdlins        SVGNAME = svg1, MIN=1, MAX=1,
               CLOPT="-e $(SVR).err -o $(SVR).out"
carray_xa     SVGNAME = svg2, MIN=1, MAX=1,
               CLOPT="-e $(SVR).err -o $(SVR).out"

*SERVICE
CARRAYNXA     SVRNAME = carray_nxa           # carray buffer, non-xa
FDLINS        SVRNAME = fdlins                # field buffer, non-xa
CARRAYAUTO    SVRNAME = carray_xa,           # carray buffer, xa ,
               AUTOTRAN=Y                     autotran利用
CARRAYNAUTO   SVRNAME = carray_xa           # carray buffer, xa ,
               明示的なトランザクション
SVCTPCALL     SVRNAME = svctpcall            # carray buffer, use
                                               tpcall in service

```

2. 環境ファイルをコンパイルし、サービス表を作成します。

cflというコマンドを使用してテキスト形式のTmax環境ファイルをバイナリ形式で作成します。Tmax環境ファイルのバイナリは特定ディレクトリーとファイル名を指定していない場合、自動的に**\$TMAXDIR/config**ディレクトリーに**tmconfig**というファイル名で格納されます。

```

$ cfl -i cobol.m
$ls -l $TMAXDIR/config
-rw-r--r-- 1 tmax dba 793256 4月24日 09:50 cobol.m

```

```
-rw-r--r-- 1 tmax dba 793256 4月24日 09:50 tmconfig
$ gst
$ ls -l $TMAXDIR/svct
-rw-r--r-- 1 tmax dba 385 4月24日 17:50 carray_nxa_svctab.c
-rw-r--r-- 1 tmax dba 491 4月24日 17:50 carray_xa_svctab.c
-rw-r--r-- 1 tmax dba 373 4月24日 17:50 fdlins_svctab.c
-rw-r--r-- 1 tmax dba 385 4月24日 17:50 svctpcall_svctab.c
```

参考

TMSファイルの作成についての詳細は、『Tmax 運用ガイド』を参照してください。

3. エンジンを起動します。

```
$ tmboot -T
TMBOOT for node(tmax5) is starting:
TMBOOT: TMM is starting: Thu Apr 24 12:09:17 2003
TMBOOT: CLL is starting: Thu Apr 24 12:09:17 2003
TMBOOT: CLH is starting: Thu Apr 24 12:09:17 2003
TMBOOT: TMS(tms_ora) is starting: Thu Apr 24 12:09:17 2003
TMBOOT: TMS(tms_ora) is starting: Thu Apr 24 12:09:17 2003
```

1.4. Sampleファイルのインストール

ガイドで使用する例は**cobolsample.tar**というファイル名で提供されます。提供されるファイルには、環境ファイル、サーバー・プログラム、サーバーのMakefileおよびsampleで提供されるサービスが正常に作動するの
かをテストできるtmdモジュールが含まれています。COBOL用のサンプル・ファイルはTmaxエンジンが事前にインストールされているということを仮定して、COBOL用に必要な部分のみをコピーするため、インストール後にcobolsanmple.tarをシステムに必ずインストールします。

\$TMAXDIRの下で以下のコマンドを使用してサンプル・ファイルをインストールします。サンプル・ファイルはtarで結合されているため、「-xvf」オプションを利用してディレクトリー構造に圧縮を解凍します。

```
$ cd $TMAXDIR
$ tar -xvf cobolsample.tar
x ./config/cobol.m, 1709 bytes, 4 tape blocks
x ./ucblinc x ./ucblinc/viewdemo.v, 382 bytes, 1 tape blocks
x ./ucblinc/viewdemo.h, 190 bytes, 1 tape blocks
x ./ucblinc/VIEWDEMO.cbl, 671 bytes, 2 tape blocks .....
x ./lib/libcblc.so, 252805 bytes, 494 tape blocks
x ./lib/libcbls.a, 352024 bytes, 688 tape blocks
$ cd $TMAXDIR
$ tar -xvf cobolsample.tar
x ./config/cobol.m, 1709 bytes, 4 tape blocks
x ./ucblinc x ./ucblinc/viewdemo.v, 382 bytes, 1 tape blocks
```

```
x ./ucblinc/viewdemo.h, 190 bytes, 1 tape blocks
x ./ucblinc/VIEWDEMO.cbl, 671 bytes, 2 tape blocks
.....
x ./lib/libcblc.so, 252805 bytes, 494 tape blocks
x ./lib/libcbls.a, 352024 bytes, 688 tape blocks
```


第2章 プログラムの作成

本章では、データを保存する構造体と関数について説明します。

2.1. データの格納

以下は、データを格納するためのCOBOLプログラムの一覧です。

プログラム名	説明
TPSTATUS.cbl	ATMIルーチンのために定義します
TPTYPE.cbl	ユーザー・データを送受信するときに使用します
TPSVCDEF.cbl	Tmaxシステムとメッセージを送受信する際に使用します
TPINFDEF.cbl	Tmaxシステムと接続するためのTPINITIALIZE()で使用します
TPSVCRET.cbl	TPRETURN()で使用され、サービスの実行状態を示します
TPTRXDEF.cbl	トランザクション・タイムアウトを設定するためにTXBEGIN()で使用します
TPCMTDEF.cbl	コミットレベルの特性を設定するためにTPSCMT()で使用します
TPAUTDEF.cbl	認証(Authentication)が必要な際にTPCHKAUTH()で使用します
TPPRIDEF.cbl	メッセージの優先順位を調整するTPSPRIO()とTPGPRIOR()で使用します
TPTRXLEV.cbl	トランザクションレベルを設定するTPGETLEV()で使用します
TPBCTDEF.cbl	非要求メッセージを送信するTPNOTIFY()とTPBROADCAST()で使用します
FDLINFO.cbl	FDLバッファを運用するFINIT()、FVSTOF()、FVFTOS()で使用します
TPEVTDEF.cbl	TPPOST()、TPSUBSCRIBE()とTPUNSUBSCRIBE()などの非要求メッセージのイベント管理に使用します
TXSTATUS.cbl	TXルーチンによってリターンコードを管理するために使用します
TXINFDEF.cbl	TXINFORMを呼び出した際に結果が保存されるデータ構造体です

2.1.1. TPSTATUS

ATMIルーチンのために定義し、以下の戻り値が使用されます。

```
*****
* TPSTATUS.cbl
*****
05 TP-STATUS          PICS9(9) COMP-5.
```

88	TPOK	VALUE 0.
88	TPEABORT	VALUE 1.
88	TPEBADDESC	VALUE 2.
88	TPEBLOCK	VALUE 3.
88	TPEINVAL	VALUE 4.
88	TPELIMIT	VALUE 5.
88	TPENOENT	VALUE 6.
88	TPEOS	VALUE 7.
88	TPEPERM	VALUE 8.
88	TPEPROTO	VALUE 9.
88	TPESVCERR	VALUE 10.
88	TPESVCFAIL	VALUE 11.
88	TPESYSTEM	VALUE 12.
88	TPETIME	VALUE 13.
88	TPETRAN	VALUE 14.
88	TPEGOTSIG	VALUE 15.
88	TPERMERR	VALUE 16.
88	TPEITYPE	VALUE 17.
88	TPEOTYPE	VALUE 18.
88	TPERELEASE	VALUE 19.
88	TPEHAZARD	VALUE 20.
88	TPEHEURISTIC	VALUE 21.
88	TPEEVENT	VALUE 22.
88	TPEMATCH	VALUE 23.
88	TPEMAXVAL	VALUE 24.
05	TPEVENT	PICS9(9) COMP-5.
88	TPEV-NOEVENT	VALUE 0.
88	TPEV-DISCONIMM	VALUE 1.
88	TPEV-SENDONLY	VALUE 2.
88	TPEV-SVCERR	VALUE 3.
88	TPEV-SVCFAIL	VALUE 4.
88	TPEV-SVCSUCC	VALUE 5.
05	TPSVCTIMOUT	PICS9(9) COMP-5.
88	TPED-NOEVENT	VALUE 0.
88	TPEV-SVCTIMEOUT	VALUE 1.
88	TPEV-TERM	VALUE 2.
05	APPL-RETURN-CODE	PICS9(9) COMP-5.

2.1.2. TPTYPE

ユーザー・データを送受信するときに使用します。REC-TYPEは送信するデータタイプを指示します。LENは送受信するデータの長さを示します。

```
*****
*  TPTYPE.cbl
```



```

*****
05 REC-TYPE          PIC X(8).
88 X-OCTET           VALUE "X_OCTET".
88 X-COMMON           VALUE "X_COMMON".
05 SUB-TYPE          PIC X(16).
05 LEN               PIC S9(9) COMP-5.
88 NO-LENGTH         VALUE 0.
05 TPTYPE-STATUS     PICS 9(9) COMP-5.
88 TPTYPEOK           VALUE 0.
88 TPTRUNCATE         VALUE 1.

```

2.1.3. TPSVCDEF

Tmaxシステムとメッセージを送受信する際に使用します。

```

*****
* TPSVCDEF.cbl
*****
05 COMM-HANDLE       PIC S9(9) COMP-5.
05 TPBLOCK-FLAG      PIC S9(9) COMP-5.
88 TPBLOCK           VALUE 0.
88 TPNOBLOCK         VALUE 1.
05 TPTRAN-FLAG       PIC S9(9) COMP-5.
88 TPTRAN            VALUE 0.
88 TPNOTRAN          VALUE 1.
05 TPREPLY-FLAG      PIC S9(9) COMP-5.
88 TPREPLY           VALUE 0.
88 TPNOREPLY         VALUE 1.
05 TPACK-FLAG        PIC S9(9) COMP-5 REDEFINES TPREPLY-FLAG.
88 TPNOACK           VALUE 0.
88 TPACK             VALUE 1.
05 TPTIME-FLAG       PIC S9(9) COMP-5.
88 TPTIME            VALUE 0.
88 TPNOTIME          VALUE 1.
05 TPSIGRSTRT-FLAG   PIC S9(9) COMP-5.
88 TPNOSIGRSTRT      VALUE 0.
88 TPSIGRSTRT        VALUE 1.
05 TPGETANY-FLAG     PIC S9(9) COMP-5.
88 TPGETHANDLER      VALUE 0.
88 TPGETANY          VALUE 1.
05 TSENDRECV-FLAG    PIC S9(9) COMP-5.
88 TSENDONLY         VALUE 0.
88 TPRECVONLY        VALUE 1.
05 TPNOCHANGE-FLAG   PIC S9(9) COMP-5.
88 TPCHANGE          VALUE 0.
88 TPNOCHANGE        VALUE 1.

```

```

05 TPSERVICETYPE-FLAG    PIC S9(9) COMP-5.
88 TPREQRSP              VALUE IS 0.
88 TPCONV                VALUE IS 1.
*
05 APPKEY                PIC S9(9) COMP-5.
05 CLIENTID              OCCURS 4 TIMES PIC S9(9) COMP-5.
05 SERVICE-NAME          PIC X(15).

```

2.1.4. TPINFDEF

Tmaxシステムと接続するためのTPINITIALIZE()で使します。

```

*****
* TPINFDEF.cbl
*****
05 USRNAME                PIC X(30).
05 CLTNAME                PIC X(30).
05 PASSWD                 PIC X(30).
05 GRPNAME                PIC X(30).
05 NOTIFICATION-FLAG      PIC S9(9) COMP-5.
88 TPU-SIG                VALUE 1.
88 TPU-DIP                VALUE 2.
88 TPU-IGN                VALUE 3.
05 ACCESS-FLAG            PIC S9(9) COMP-5.
88 TPSA-FASTPATH          VALUE 1.
88 TPSA-PROTECTED         VALUE 2.
05 DATALEN PICS9(9)      COMP-5.

```

2.1.5. TPSVCRET

TPRETURN()で使され、サービス実行状態を示します。

```

*****
* TPSVCRET.cbl
*****
05 TP-RETURN-VAL          PIC S9(9) COMP-5.
88 TPSUCCESS             VALUE 0.
88 TPFAIL                 VALUE 1.
88 TPEXIT                 VALUE 2.
05 APPL-CODE              PIC S9(9) COMP-5.

```

2.1.6. TPTRXDEF

トランザクション・タイムアウトを設定するためにTXBEGIN()で使します。

```
*****
* TPTRXDEF.cbl
*****
05 T-OUT PICS9(9)      COMP-5 VALUE IS0.
05 TRANID              OCCURS 6 TIMES PICS9(9) COMP-5.
```

2.1.7. TPCMTDEF

コミットレベルの特性を設定するためにTPSCMT()で使します。

```
*****
* TPCMTDEF.cbl
*****
05 CMT-FLAG            PIC S9(9) COMP-5.
88 TP-CMT-LOGGED       VALUE 1.
88 TP-CMT-COMPLETE     VALUE 2.
05 PREV-CMT-FLAG      PIC S9(9) COMP-5.
88 PREV-TP-CMT-LOGGED  VALUE 1.
88 PREV-TP-CMT-COMPLETE VALUE 2.
```

2.1.8. TPAUTDEF

認証(Authentication)が必要な際にTPCHKAUTH()で使します。

```
*****
* TPAUTDEF.cbl
*****
05 AUTH-FLAG          PIC S9(9) COMP-5.
88 TPNOAUTH           VALUE 0.
88 TPSYSAUTH          VALUE 1.
88 TPAPPAUTH          VALUE 2.
```

2.1.9. TPPRIDEF

メッセージの優先順位を調整するTPSPRIO()とTPGPRI()で使します。

```
*****
* TPPRIDEF.cbl
*****
```

```

05 PRIORITY          PIC S9(9) COMP-5.
05 PRIO-FLAG         PIC S9(9) COMP-5.
88 TPABSOLUTE        VALUE 0.
88 TPRELATIVE        VALUE 1.

```

2.1.10. TPTRXLEV

トランザクションレベルを設定するTPGETLEV()で使します。

```

*****
* TPTRXLEV.cbl
*****
05 TPTRXLEV-FLAG     PIC S9(9) COMP-5.
88 TP-NOT-IN-TRAN    VALUE 0.
88 TP-IN-TRAN        VALUE 1.

```

2.1.11. TPBCTDEF

非要求メッセージを送信するTPNOTIFY()とTPBROADCAST()で使します。

```

*****
* TPBCTDEF.cbl
*****
05 TPBLOCK-FLAG     PIC S9(9) COMP-5.
88 TPBLOCK          VALUE 0.
88 TPNOBLOCK        VALUE 1.
05 TPTIME-FLAG      PIC S9(9) COMP-5.
88 TPTIME           VALUE 0.
88 TPNOTIME         VALUE 1.
05 TPSIGRSTRT-FLAG  PIC S9(9) COMP-5.
88 TPNOSIGRSTRT     VALUE 0.
88 TPSIGRSTRT       VALUE 1.
05 LMID             PIC X(30).
05 USERNAME         PIC X(30).
05 CLTNAME          PIC X(30).

```

2.1.12. FDL-INFO

FDLバッファを運用するFINIT()、FVSTOF()、FVFTOS()で使します。

```

*****
* FDLINFO.cbl
*****

```

```

05 FDL-STATUS          PIC S9(9) COMP-5.
88 FOK                 VALUE 0.
88 FALIGNERR          VALUE 1.
88 FNOTFLD            VALUE 2.
88 FNOSPACE           VALUE 3.
88 FNOTPRES           VALUE 4.
88 FBADFLD            VALUE 5.
88 FTYPERR            VALUE 6.
88 FEUNIX             VALUE 7.
88 FBADNAME           VALUE 8.
88 FMALLOC            VALUE 9.
88 FSYNTAX            VALUE 10.
88 FFTOPEN            VALUE 11.
88 FFTSYNTAX          VALUE 12.
88 FEINVAL            VALUE 13.
88 FBADTBL            VALUE 14.
88 FBADVIEW           VALUE 15.
88 FVFSYNTAX          VALUE 16.
88 FVFOPEN            VALUE 17.
88 FBADACM            VALUE 18.
88 FNOCNAME           VALUE 19.
*
05 FDL-LENGTH          PIC S9(9) COMP-5.
*
05 FDL-MODE            PIC S9(9) COMP-5.
88 FUPDATE            VALUE 1.
88 FCONCAT            VALUE 2.
88 FJOIN              VALUE 3.
88 FOJOIN             VALUE 4.
*
05 VIEWNAME            PIC X(33).

```

2.1.13. TPEVTDEF

TPPOST()、TPSUBSCRIBE()とTPUNSUBSCRIBE()などの非要求メッセージのイベント管理に使用します。

```

*****
* TPEVTDEF.cbl
*****
05 TPBLOCK-FLAG        PIC S9(9) COMP-5.
88 TPBLOCK             VALUE 0.
88 TPNOBLOCK           VALUE 1.
05 TPTRAN-FLAG         PIC S9(9) COMP-5.
88 TPTRAN              VALUE 0.

```

```

88 TPNOTRAN                VALUE 1.
05 TPREPLY-FLAG            PIC S9(9) COMP-5.
88 TPREPLY                  VALUE 0.
88 TPNOREPLY                VALUE 1.
05 TPTIME-FLAG              PIC S9(9) COMP-5.
88 TPTIME                    VALUE 0.
88 TPNOTIME                  VALUE 1.
05 TPSIGRSTRT-FLAG          PIC S9(9) COMP-5.
88 TPNOSIGRSTRT              VALUE 0.
88 TPSIGRSTRT                VALUE 1.
05 TPEV-METHOD-FLAG        PIC S9(9) COMP-5.
88 TPEVNOTIFY                VALUE 0.
88 TPEVSERVICE                VALUE 1.
88 TPEVQUEUE                  VALUE 2.
05 TPEV-PERSIST-FLAG        PIC S9(9) COMP-5.
88 TPEVNOPERSIST              VALUE 0.
88 TPEVPERSIST                VALUE 1.
05 TPEV-TRAN-FLAG            PIC S9(9) COMP-5.
88 TPEVNOTRAN                VALUE 0.
88 TPEVTRAN                  VALUE 1
*
05 EVENT-COUNT              PIC S9(9) COMP-5.
05 SUBSCRIPTION-HANDLE        PIC S9(9) COMP-5.
05 NAME-1                    PIC X(31).
05 NAME-2                    PIC X(31).
05 EVENT-NAME                PIC X(31).
05 EVENT-EXPR                PIC X(255).
05 EVENT-FILTER                PIC X(255).

```

2.1.14. TXSTATUS

TXルーチンによってリターンコードを管理するために使用します。

```

*****
* TXSTATUS.cbl
*****
05 TX-STATUS                PIC S9(9) COMP-5.
88 TX-NOT-SUPPORTED          VALUE 1.
* Normal execution
88 TX-OK                      VALUE 0.
* Normal execution
88 TX-OUTSIDE                 VALUE -1.
* Application is in an RM local transaction
88 TX-ROLLBACK                VALUE -2.
* Transaction was rolled back
88 TX-MIXED                   VALUE -3.

```

```

* Transaction was partially committed and partially
* rolled back
88 TX-HAZARD                VALUE -4.
* Transaction may have been partially committed and
* partially rolled back
88 TX-PROTOCOL-ERROR VALUE -5.
* Routine invoked in an improper context
88 TX-ERROR                 VALUE -6.
* Transient error
88 TX-FAIL                 VALUE -7.
* Fatal error
88 TX-EINVAL               VALUE -8.
* Invalid arguments were given
88 TX-COMMITTED            VALUE -9.
* The transaction was heuristically committed
88 TX-NO-BEGIN            VALUE -100.
* Transaction committed plus new transaction could not
* be started
88 TX-ROLLBACK-NO-BEGIN VALUE -102.
* Transaction rollback plus new transaction could not
* be started
88 TX-MIXED-NO-BEGIN      VALUE -103.
* Mixed plus new transaction could not be started
88 TX-HAZARD-NO-BEGIN    VALUE -104.
* Hazard plus new transaction could not be started.
88 TX-COMMITTED-NO-BEGIN VALUE -109.
* Heuristically committed plus transaction could not
* be started

```

2.1.15. TXINFDEF

TXINFORMを呼び出した際に結果が格納されるデータ構造体です。

```

*****
* TXINFDEF.cbl
*****
05 XID-REC.

* XID record
10 FORMAT-ID                PIC S9(9) COMP-5.

* A value of -1 in FORMAT-ID means that the XID is null
10 GTRID-LENGTH            PIC S9(9) COMP-5.
10 BRANCH-LENGTH          PIC S9(9) COMP-5.
10 XID-DATA                PIC X(128).
05 TRANSACTION-MODE        PIC S9(9) COMP-5.

```

```

* Transaction mode settings
88 TX-NOT-IN-TRAN          VALUE 0.
88 TX-IN-TRAN              VALUE 1.
05 COMMIT-RETURN          PIC S9(9) COMP-5.

* Commit_return settings
88 TX-COMMIT-COMPLETED    VALUE 0.
88 TX-COMMIT-DECISION-LOGGED VALUE 1.
05 TRANSACTION-CONTROL    PIC S9(9) COMP-5.

* TRANSACTION-CONTROL settings
88 TX-UNCHAINED            VALUE 0.
88 TX-CHAINED              VALUE 1.
05 TRANSACTION-TIMEOUT    PIC S9(9) COMP-5.

* Transaction_timeout value
88 NO-TIMEOUT              VALUE 0.
05 TRANSACTION-STATE      PIC S9(9) COMP-5.

* Transaction_state information
88 TX-ACTIVE               VALUE 0.
88 TX-TIMEOUT-ROLLBACK-ONLY VALUE 1.
88 TX-ROLLBACK-ONLY        VALUE 2.

```

2.2. 関数

以下は、COBOLプログラムで使用される関数一覧です。

関数名	説明
FINIT	フィールド化バッファを初期化します
FVFTOS	フィールド化バッファからCOBOL構造体にデータをコピーします
FVSTOF	C構造体からフィールド化バッファにデータを転送する際に使用します
TP_SLEEP	データの到着を秒単位で待機する関数で、サーバー/クライアントで使用します
TPACALL	非同期サービス要求を送信します
TPCALL	同期型サービス要求を送受信します
TPCANCEL	応答をキャンセルします
TPFORWARD	サービス要求を他のサービス・ルーチンに渡します
TPGETLEV	トランザクション・モードかどうかを確認します
TPGETRPLY	非同期サービス要求の応答を受信します
TPRETURN	Tmaxサービスを終了します

関数名	説明
TPSVCSTART	Tmaxサービスを開始します
TPSVRDONE	サーバー・プロセスを終了します
TPSVRINIT	Tmaxサーバーを初期化します
TXBEGIN	グローバル・トランザクションを開始します
TXCOMMIT	グローバル・トランザクションをコミットします
TXINFORM	グローバル・トランザクション情報を返します
TXROLLBACK	グローバル・トランザクションをロールバックします
TXSETCOMMITRET	commit-return特性を設定します
TXSETTIMEOUT	transaction-timeout特性を設定します
TXSETTRANCTL	transaction-control特性を設定します

2.2.1. FINIT

フィールド化バッファを初期化する関数です。

● 使用方法

```

01 FDL-BUFFER.
05 FDL-ALIGN      PIC S9(9) USAGE IS COMP.
05 FDL-DATA       PIC X(ユーザー指定の長さ).
01 FDL-REC
COPY FDLINFO.

CALL "FINIT"      USING FDL-BUFFER FDL-REC.
```

パラメータ	説明
FDL-BUFFER	フィールド化バッファのために使用されるレコードで、FDLバッファのためには4バイトが割り当てられる必要があります
FDL-LENGTH IN FDL-REC	FDLバッファで初期化するレコードの長さです

● 戻り値

実行結果	説明
成功	フィールド化バッファの初期化に成功した場合、FDL-STATUS IN FDL-REC がFOKに設定されます
失敗	FDL-STATUS IN FDL-RECが0でない値が設定されます

- エラー

関数の実行に失敗した場合、以下の値が設定されます。

エラーコード	説明
[FNOSPACE]	フィールド化バッファにデータを格納あるいはコピーする領域が不足する 合です

- 参照

FVFTOS

2.2.2. FVFTOS

フィールド化バッファからCOBOL構造体にデータをコピーする関数で、フィールド化バッファからCOBOLレコードにデータを転送する際に使用します。

- 使用方法

```
01 DATA-REC.  
COPY User data.  
01 FDL-BUFFER.  
05 FDL-ALIGN PIC S9(9) USAGE IS COMP.  
05 FDL-DATA PIC X(applen).  
01 FDL-REC COPY FDLINFO.  
  
CALL "FVFTOS" USING FDL-BUFFER DATA-REC FDL-REC.  
CALL "FVFTOS32" USING FDL-BUFFER DATA-REC FDL-REC.
```

パラメータ	説明
FDL-BUFFER	FINITによって初期化されたフィールド化バッファのポインタです
DATA-REC	C構造体のポインタです
VIEWNAME IN FDL-REC	COBOLレコードを説明するビューの名前です。 フィールドは、フィールド化バッファからVIEWNAMEにあるelement descriptionsをベースとした構造体にコピーされます。フィールド化バッファのフィールドがCOBOLレコードに相対するフィールドがない場合は無視されます。COBOLレコードにある特定のelementがフィールド化バッファに相対するフィールドがない場合はNULL値がelementにコピーされます。COBOLレコードにArray形式で格納するためには、レコードのelementはOCCURSに定義される必要があります。バッファがレコードのelementよりフィールド数が少ない場合、残ったelementはビュー・ファイルで指定したデフォルト値に設

パラメータ	説明
	定されます。反対に、バッファがレコードのelementよりフィールド数が多い場合、残りのoccurrencesは無視されます

- 戻り値

実行結果	説明
成功	FDL-STATUS IN FDL-RECはFOKに設定されます
失敗	FDL-STATUSに0ではない値が設定されます

- エラー

関数の実行に失敗した場合、以下の値が設定されます。

エラーコード	説明
[FBENOENT]	指定したフィールドキーがFDLFILEに定義されていないため、フィールド化バッファで使用できない場合に設定されます
[FBENOSPACE]	フィールド化バッファにデータを格納あるいはコピーする領域が不足する場合に設定されます
[FEINVAL]	入力パラメータに指定した値の中に有効でないパラメータが存在する場合に設定されます
[FBEBADFLD]	有効でないフィールドキーが使用されました。一般的にフィールドキーがfdlcユーティリティを使用してコンパイルされていないフィールドキーが使用された場合に設定されます
[FBEBADSTRUCT]	有効でない構造体が使用されました。一般的にTmaxシステムで認識できない形式の構造体を使用した場合や構造体定義ファイルが正しくコンパイルされなかった場合に設定されます

- 参照

FVSTOF

2.2.3. FVSTOF

C構造体からフィールド化バッファにデータを転送する際に使用します。

- 方法

```
01 DATA-REC.
COPY User data.
```

```

01 FDL-BUFFER.
05 FDL-ALIGN PIC S9(9) USAGE IS COMP.
05 FDL-DATA PIC X(applen).
01 FDL-REC
COPY FDLINFO.

CALL "FVSTOF" USING FDL-BUFFER DATA-REC FDL-REC.
CALL "FVSTOF32" USING FDL-BUFFER DATA-REC FDL-REC.

```

項目	説明
DATA-REC	COBOLレコードです
FDL-BUFFER	フィールド化バッファを含むレコードです
FDL-REC	VIEWNAMEはCOBOLレコードを指定するために使用したビューの名前です
FDL-MODE IN FDL-REC	<p>フィールド化バッファにデータを転送する方法を指定します。</p> <ul style="list-style-type: none"> – FUPDATE – FOJOIN – FJOIN – FCONCAT <p>モードは各C関数のFupdate(3)、Fojoin(3)、Fjoin(3)、Fconcatと同じ方式で動作します</p>

● 戻り値

実行結果	説明
成功	FDL-STATUS IN FDL-RECにはFOKが設定されます
失敗	FDL-STATUSには0ではない値が設定されます

● エラー

関数の実行に失敗した場合、以下の値が設定されます。

エラーコード	説明
[FBENOENT]	指定したフィールドがFDLFILEに定義されていないため、フィールド化バッファで使用できない場合に設定されます
[FBENOSPACE]	フィールド化バッファのデータを格納あるいはコピーする領域が不足する場合に設定されます
[FEINVAL]	入力パラメータに指定した値の中に有効でないパラメータが存在する場合に設定されます

エラーコード	説明
[FBEBADFLD]	有効でないフィールドキーが使用されました。一般的にフィールドキーがfdlcユーティリティを使用してコンパイルされていないフィールドキーが使用された場合に設定されます
[FBEBADSTRUCT]	有効でない構造体が使用されました。一般的にTmaxシステムで認識できない形式の構造体を使用した場合や構造体定義ファイルが正しくコンパイルされなかった場合に設定されます
[FBEMALLOC]	システム・エラーでフィールド化バッファをメモリーに割り当てることに失敗した場合に設定されます

- 参照

FVFTOS

2.2.4. TP_SLEEP

データの到着を秒単位で待機する関数で、サーバー/クライアントで使します。最大sec時間の間待機し、その間にデータが到着すれば即座に返します。

- 使用方法

```
int TP_SLEEP (int sec, cbl_tpstatus_t *status)
```

パラメータ	説明
sec	待機する時間を秒単位の正数値で入力します

- 戻り値

sec時間までデータが到着しなかった場合は0を返し、データが到着した場合は正数を返します。エラーが発生した場合は-1を返し、tperrno変数値が設定されます。

- エラー

以下の状況で、tp_sleep()は失敗し、tperrnoに以下の値のうち1つが設定されます。

エラーコード	説明
[TPESYSTEM]	Tmaxシステムにエラーが発生しました。詳細情報はログファイルに記録されます
[TPEOS]	OSにエラーが発生しました

- 参照

2.2.5. TPACALL

非同期サービス要求の送信を行う関数です。TPACALL関数は、SERVICE-NAME IN TPSVCDEF-RECに命令されたサービスに要求メッセージを送信します。TPACALL関数は、非同期通信でメッセージを送信後、結果を受信するまで待機せず、即座に返します。結果は後にTPGETRPLY関数を使用して応答を受けることもでき、TPCANCEL関数を使用して応答を取り消すこともできます。

サービス要求がトランザクション・モードで送信された場合、各要求に対する応答は受信される必要があります。

● 使用方法

```
01 TPSVCDEF-REC.  
COPY TPSVCDEF.  
01 TPTYPE-REC.  
COPY TPTYPE.  
01 DATA-REC.  
COPY User data.  
01 TPSTATUS-REC.  
COPY TPSTATUS.  
  
CALL "TPACALL" USING TPSVCDEF-REC TPTYPE-REC DATA-REC TPSTATUS-REC.
```

– DATA-RECは送信するデータであり、LEN IN TPTYPE-RECはDATA-RECに格納して送信されるデータの長さを指定します。DATA-RECが指定された長さを要求しないタイプのレコードの場合、LENは無視されます。(通常は1が使用されます)

– REC-TYPE IN TPTYPE-RECがSPACESの場合、DATA-RECとLENは無視され、データなしで送信されます。REC-TYPEがSTRINGで、LENが0の場合、データなしで送信されます。

REC-TYPEがSTRINGであり、かつLENが0の場合はデータなしで送信されます。DATA-RECのREC-TYPEとSUB-TYPEは、SERVICE-NAMEがサポートするタイプである必要があります。

– 以下は、TPSVCDEF-RECに設定できる値です。

設定値	説明
TPBLOCK	フラグなしでtpacall()関数が使用された場合、svclに呼び出されたサービスがない場合や正しくない結果が返された場合も、正常な結果が返されます。tpgetrply()関数を呼び出す際にエラーが返されます。フラグを利用してtpacall()関数を呼び出す際に、サービス状態が正常か否かを確認できます

設定値	説明
TPNOTRAN	<p>TPACALL関数の呼び出し元がトランザクション・モードにあり、この設定を使用してSERVICE-NAMEが要求された場合、SERVICE-NAME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。</p> <p>トランザクション・モードでトランザクションをサポートしていないSERVICE-NAMEを呼び出す場合はこの設定を必ず行います。トランザクション・モード内でTPACALL関数が呼び出された場合、TPNOTRANに設定されていても、依然としてトランザクション・タイムアウトに影響を受けます。TPNOTRANで呼び出されたサービスが失敗した場合、呼び出し元のトランザクションには影響を与えず、TPNOREPLYの応答を待機せず、即座に返します。サービスが正常に呼び出された場合はTPOKが返され、COMM-HANDLE IN TPSVCDEF-RECに0が設定されます。</p> <p>関数の呼び出し元がトランザクション・モードにある場合は、TPNOTRANと一緒に設定しなければTPNOREPLYを使用できません。TPNOREPLYの場合、サービス状態が正常か否かをチェックするためには、TPBLOCKと一緒に設定します。TPBLOCKを設定していない場合、サービスがNRDYの場合にもエラーを返しません</p>
TPNOBLOCK	<p>TPNOBLOCKを設定した状態でブロッキング状況が存在する場合（たとえば、内部バッファが送信するメッセージでいっぱいの場合）、サービスの要求は失敗します。</p> <p>TPNOBLOCKを設定せずにTPACALLが呼び出されると、ブロッキング状況が存在する場合は、関数の呼び出し元はブロッキング状況が消失するか、タイムアウト（トランザクション・タイムアウトまたはブロッキング・タイムアウト）が発生するまで待機します</p>
TPNOTIME	<p>TPNOTIMEは、関数の呼び出し元がブロッキング・タイムアウトを無視し、応答が受信されるまで無限に待機することを意味します。トランザクション・タイムアウトでTPACALLを行った場合は、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</p>
TPSIGRSTRT	<p>シグナルの割り込みを受け入れる場合に使用します。システム関数の呼び出しが妨害された場合、システム関数の呼び出しが再実行されます。このフラグが設定されていない状態でシグナルの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</p>

- 戻り値

実行結果	説明
成功	TP-STATUSはTPOKに設定されます
失敗	TP-STATUSはエラーコードが設定されます

- エラー

関数の実行に失敗した場合、TP-STATUSに以下の値が設定されます。

エラーコード	説明
[TPEINVAL]	パラメータが有効ではありません。たとえば、SERVICE-NAMEがSPACESである場合や、TPSVCDEF-RECが正常ではない場合です
[TPENOENT]	SERVICE-NAMEのサービスが存在しないため、サービスを要求できません
[TPEITYPE]	REC-TYPEとSUB-TYPEの構成がサポートしていないタイプです
[TPELIMIT]	処理されていない非同期サービス要求の数が最大数に達したため、呼び出し元のサービス要求が送信できませんでした
[TPETRAN]	SERVICE-NAMEはトランザクションをサポートしていないサービスで、この際にTPNOTRANが設定されませんでした
[TPETIME]	タイムアウトが発生しました。関数の呼び出し元がトランザクション・モードの場合はトランザクション・タイムアウトが発生し、トランザクションはロールバックされます。トランザクション・モードでない場合、TPNOTIMEとTPNOBLOCKが両方設定されていない状態でブロッキング・タイムアウトが発生します。トランザクション・タイムアウトが発生すると、トランザクションがロールバックされるまでは、新しいサービス要求を送信したり、未処理の応答を受信しようとしても、すべて[TPETIME]エラーで失敗します
[TPEBLOCK]	TPNOBLOCKが設定された状態でブロッキング状況が発生しました
[TPEGOTSIG]	TPSIGRSTRTが設定されていない状態でシグナルが受信されました
[TPEPROTO]	トランザクション・モードでのTPNOREPLYサービスが呼び出されると、TPNOTRANフラグを設定していない場合などの不適切な状況で発生します
[TPESYSTEM]	Tmaxシステムにエラーが発生しました
[TPEOS]	OSにエラーが発生しました

- 参照

TPCALL, TPCANCEL, TPGETRPLY

2.2.6. TPCALL

TPCALLは、同期型サービスで要求を送信し、応答を待つ関数です。この関数への呼び出しは、TPACALL()の呼び出し後に連続的にTPGETRPLYを呼び出すのと同じことです。TPCALLはTPSVCDEF-RECのSERVICE-NAMEでサービス要求を送受信します。

- 使用方法


```

01 TPSVCDEF-REC.
COPY TPSVCDEF.
01 IPTYPE-REC.
COPY TPTYPE.
01 IDATA-REC.
COPY User data.
01 OTPTYPE-REC.
COPY TPTYPE.
01 ODATA-REC.
COPY User data.
01 TPSTATUS-REC.
COPY TPSTATUS.
CALL "TPCALL" USING TPSVCDEF-REC IPTYPE-REC IDATA-REC OTPTYPE-REC
ODATA-REC TPSTATUS-REC.

```

- IPTYPE-RECでIDATA-RECは送信するデータであり、LENは送信されるデータの長さを指定します。IDATA-RECが指定された長さを要求しないタイプのレコードの場合、LENは無視されます(通常は1が使用されます)。

IPTYPE-RECのREC-TYPEがSPACESの場合、IDATA-RECとIPTYPE-RECのLENは無視され、データなしで要求されます。IPTYPE-RECがSTRINGでありながらIPTYPE-RECのLENが0の場合、データなしで要求されます。

IPTYPE-RECのREC-TYPEとSUB-TYPEは、SERVICE-NAMEがサポートするタイプである必要があります。応答を格納して送信するのに使用されるODATA-RECとODATA-RECに格納されるデータの最大長を考慮してODATA-RECのLENが指定される必要があります。データを送受信するのに同じレコードを使用する場合、ODATA-RECはIDATA-RECを再定義する必要があります。

- TPCALLが正常に実行されると、ODATA-RECにはOTPTYPE-RECのLEN分の応答が受信されます。OTPTYPE-RECのREC-TYPEとSUB-TYPEは、各応答タイプとサブタイプを含みます。応答がODAT-RECより大きい場合、ODATA-RECレコードに適切なデータ分のみ含むようになります。応答の残りは捨てられ、TPTRUNCATEが設定されます。正常な受信でOTPTYPE-RECのLENが0の場合、応答にはデータ部がなく、ODATA-RECはデータが受信されません。OTPTYPE-RECのLENに0が入ることはエラーです。

- 以下は、TPSVCDEF-RECに設定できる値です。

設定値	説明
TPNOTRAN	<p>呼び出し元がトランザクション・モードにあり、この設定を使用してSERVICE-NAMEサービスを呼び出した場合、トランザクション・モードから除外されて実行されます。</p> <p>トランザクション・モードでトランザクションをサポートしていないサービスを呼び出す場合は、この設定を使用する必要があります。トランザクション・モード内でTPNOTRANを設定してTPCALLを使用しても、依然としてトランザクション・タイム</p>

設定値	説明
	アウトに影響を受けます。TRNOTRANで呼び出されたサービスが失敗した場合、呼び出し元のトランザクションには影響を与えません
TPNOCHANGE	<p>一般的に、受信した応答バッファのODATA-RECが示すバッファのタイプが異なる場合、バッファのタイプは受信側が認識できる限り、受信した応答バッファのタイプに変わります。応答レコードのタイプとサブタイプは、REC-TYPE IN OTPTYPE-RECとSUB-TYPE IN OTPTYPE-RECが一致する必要があります。</p> <p>TPNOCHANGEが設定されている場合、ODATA-RECに設定されたタイプは変更されません</p>
TPNOBLOCK	<p>TPNOBLOCKを設定した状態でブロッキング状況が存在する場合（たとえば、内部バッファが送信するメッセージでいっぱいの場合）、サービス要求は失敗します。</p> <p>TPNOBLOCKを設定せずに呼び出すと、ブロッキング状況が存在する場合は、関数の呼び出し元はブロッキング状況が消失するか、またはタイムアウト（トランザクション・タイムアウトまたはブロッキング・タイムアウト）が発生するまで待機します</p>
TPNOTIME	TPNOTIMEは、関数の呼び出し元がブロッキング・タイムアウトを無視し、応答が受信されるまで無限に待機することを意味します。トランザクション・タイムアウトでTPCALLを行った場合は、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	<p>シグナルの割り込みを受け入れる場合に使用します。内部でシグナルの割り込みが発生してシステム関数の呼び出しが妨害された場合、システム関数の呼び出しが再実行されます。</p> <p>値が設定されていない状態でシグナルの割り込みが発生した場合、関数は失敗し、TPSTATUSにTPGOTSIGが設定されます</p>

- 戻り値

実行結果	説明
成功	<p>TP-STATUSにTPOKが設定されます</p> <p>TP-STATUSにTPOKかTPESVCFAILが設定されている場合、TPSTATUS-RECのAPPL-RETURN-CODEはTPRETURNの一部分として送信されるアプリケーション・プログラムが指定した値を含みます。受信したメッセージのサイズが指定されたLENより大きい場合はTPTRUNCATEが設定され、LEN分のデータのみODATA-RECに設定され、残りのデータは無視されます</p>

実行結果	説明
失敗	TP-STATUSはエラーコードが設定されます

● エラー

関数の実行に失敗した場合、TP-STATUSに以下の値が設定されます。

エラーコード	説明
[TPEINVAL]	引数が有効ではありません。たとえば、SERVICE-NAMEがSPACESであったり、TPSVCDEF-RECが正常ではありません
[TPENOENT]	SERVICE-NAMというサービスが存在しないため、サービスを要求できません
[TPEITYPE]	REC-TYPEとSUB-TYPE is SERVICE-NAMEがサポートしていないタイプです
[TPEOTYPE]	受信した応答バッファのタイプあるいは下位タイプが、呼び出し元が認識できないタイプです。フラグがTPNOCHANGEに設定されますが、ODATA-RECのREC-TYPEとSUB-TYPEが受信した応答バッファのタイプおよび下位タイプと合いません。この場合、OTPTYPE-RECのODATA-RECとLENの両方が変更されません。 呼び出し元がトランザクション・モードでサービスを要求した場合、そのトランザクションは応答が無視されるため、ロールバックされます
[TPETRAN]	SERVICE-NAMEはトランザクションをサポートしておらず、TPNOTRANが設定されていませんでした
[TPETIME]	タイムアウトが発生しました。関数の呼び出し元がトランザクション・モードの場合はトランザクション・タイムアウトが発生しました。トランザクション・モードではなく、TPNOTIMEとTPNOBLOCKのどちらも指定されていない場合は、ブロッキング・タイムアウトが発生しました。この2つの場合にOTPTYPE-RECのODATA-RECとLENは両方変更されません。 トランザクション・タイムアウトが発生した場合、トランザクションがロールバックされるまで、新しいサービス要求の送信や応答の受信は[TPETIME]エラーで失敗します
[TPESVCFAIL]	アプリケーション・プログラム上のエラーが発生して、サービス要求の応答を送信するサービス・ルーチンがTPFAILでTPRETURN()を呼び出しました。 サービスの応答が受信された場合、ODATA-RECの内容を利用できます。トランザクション・タイムアウトが発生するまでは、トランザクションがロールバックする前に他の通信が行われることがあります。そのような通信は、正常に処理されることもあれば、失敗することもあります。これが正常に実行されるためには、TPNOTRANが設定されている必要があります。呼び出し元のトランザクション・モードで実行された作業は、トランザクションの完了時にすべてロールバックされます

エラーコード	説明
[TPESVCERR]	サービス・ルーチンが実行中やTPRETURNの実行中に正しくない引数が渡された場合など、エラーが発生しました。エラーが発生した場合、応答データは返されず、OTPTYPE-RECのODATA-RECとLENの両方が変更されません。 Tmax環境ファイルにサービス別にSVCTIMEOUTを設定できますが、サービスの実行時間がこの時間を超過するとサービスは実行を止め、TPESVCERRを返します
[TPEBLOCK]	TPNOBLOCKが設定された状態でブロッキング状況が発生しました
[TPEGOTSIG]	TPSIGRSTRTが設定されていない状態でシグナルを受信しました
[TPEPROTO]	tpcall()が不適切な状況で呼び出されました
[TPESYSTEM]	Tmaxシステムのエラーが発生しました。詳細情報はログファイルに記録されます
[TPEOS]	OSにエラーが発生しました

2.2.7. TPCANCEL

応答をキャンセルする関数で、TPACALLが返した呼び出し記述子のCOMM-HANDLE IN TPSVCDEF-RECをキャンセルします。グローバル・トランザクションに関連するサービスはキャンセルできません。サービスの応答が正常にキャンセルされた場合、呼び出し記述子は無効になり、これを通じて受信した応答もすべて無視されます。

- 使用方法

```
01 TPSVCDEF-REC.
COPY TPSVCDEF.
01 TPSTATUS-REC.
COPY TPSTATUS.
CALL "TPCANCEL" USING TPSVCDEF-REC TPSTATUS-REC.
```

- 戻り値

実行結果	説明
成功	TP-STATUSはTPOKに設定されます
失敗	TP-STATUSはエラーコード値が設定されます

- エラー

関数の実行に失敗した場合、TP-STATUSに以下の値が設定されます。

エラーコード	説明
[TPEBADDESC]	COMM-HANDLEが無効な記述子です
[TPETRAN]	COMM-HANDLEが呼び出し元のトランザクションに関連しています。COMM-HANDLEは依然として有効で、呼び出し元の現在のトランザクションは影響を受けません
[TPEPROTO]	TPCANCELが不適切な状況で呼び出されました
[TPESYSTEM]	Tmaxシステムのエラーが発生しました。詳細情報はログファイルに記録されます
[TPEOS]	Tmaxシステムのエラーが発生しました。詳細情報はログファイルに記録されます。 または、OSにエラーが発生しました

- 参照

TPACALL

2.2.8. TPFORWARD

サービス要求を他のサービス・ルーチンに転送する関数です。

TPFORWARDは、自身のサービス処理を終了し、クライアントの要求を他のサービスに渡します。TPFORWARDはEXIT PROGRAMを含んでおり、TPRETURNのように作動します。TPRETURNと同様に、TPFORWARDがTmaxシステムに正常に返されるためには、TPFORWARDはTmaxシステムが制御するサービス・ルーチン内から呼び出される必要があります。会話型サービスからはTPFORWARDを呼び出すことはできません。

サービス・ルーチンがトランザクション・モードである場合、このトランザクションはトランザクション・コーディネーターがTXCOMMITまたはTXROLLBACKのうち1つを実行してトランザクションを完了する際、関数も完了します。トランザクションがそのサービス・ルーチン内でTXBEGINを使用して開始された場合、そのトランザクションはTPFORWARDの呼び出し前にTXCOMMITまたはTXROLLBACKのうちの1つで先に終了させる必要があります。TPFORWARDで接続されたすべてのサービスは、すべてがトランザクション・モードであるか、あるいはすべてがトランザクション・モードでない必要があります。

最終的に、TPFORWARDされたサービスがTPRETURNを使用して最初にサービスを要求したクライアントに応答を返します。TPFORWARDは応答を待機しているリクエストに応答を送信する役割を他のサーバー・プロセスに転送し、マルチ・ノード間でもサービスが行われます。TPFORWARDは、サービス・ルーチンが要求したすべてのサービスに対する応答を受け取った後に呼び出されなければなりません。受信されていない応答の記述子は無効になり、要求は転送されません。

サービス・ルーチンの作成者は、TPFORWARDの呼び出し後に再度制御権を取得することはできないため、TPSIGRSTRTが暗示的に定義された形式のブロッキング送信に使用します。TPFORWARDの実行中にシ

グナルが発生して実行が中止されても再実行され、ブロッキング状況が存在してもタイムアウトの発生までは待機して送信します。現在TPSVCDEF-RECは使用されておらず、指定された値は無視されます。

● 使用方法

```
01 TPSVCDEF-REC.  
COPY TPSVCDEF.  
01 TPTYPE-REC.  
COPY TPTYPE.  
01 DATA-REC.  
COPY User data.  
01 TPSTATUS-REC.  
COPY TPSTATUS.  
COPY TPFORWAR REPLACING TPSVCDEF-REC BY TPSVCDEF-REC  
TPTYPE-REC BY TPTYPE-REC  
DATA-REC BY DATA-REC  
TPSTATUS-REC BY TPSTAUS-REC
```

– DATA-RECに格納されたデータを使用して、SERVICE-NAMEで命令されたサービスに要求を渡します。要求を渡すサービス・ルーチンはいかなる応答も受信しません。要求が渡された後、サービス・ルーチンはTmaxシステムに返されます。そして、サーバーは自由に他の作業を実行できます。TPFORWARDは、リクエストからの応答は期待しないため、任意のサービスにエラーなしで転送できます。

– DATA-RECは送信されるレコードで、TPTYPE-RECのLENは送信されるデータの長さです。DATA-RECが長さの明示が必要ないバッファを指す場合、LENは無視されます（通常は0が使用されます）。

TPTPE-RECのREC-TYPEがSPACESの場合、DATA-RECとLENは無視され、データの長さが0の要求が送信されます。

● 戻り値

サービス・ルーチンは、呼び出し元であるTmaxシステムでいかなる値も返さず、TP-STATUSは設定されません。

● エラー

以下の状況でTPFORWARDは失敗し、tperrnoに以下の値のうち1つが設定されます。

エラーコード	説明
[TPESVCERR]	サービス・ルーチンの実行中、またはTPRETURNの実行中にエラーが発生しました。エラーが発生すると、いかなる応答データも返しません。この場合、OTPTYPE-RECのODATA-RECとLENの両方が変更されません
[TPETIME]	サービス・ルーチンの作業中、または要求の転送中にトランザクション・タイムアウト発生した場合に、TPETIMEエラーが返されます

- 参照

TPRETURN

2.2.9. TPGETLEV

TPGETLEVは、トランザクション・モードかどうかをチェックします。現在は、TP-NOT-IN-TRANとTP-IN-TRANの2つのトランザクション・モードが定義されています。

- 使用方法

```
01 TPTRXLEV-REC.  
COPY TPTRXLEV.  
01 TPSTATUS-REC.  
COPY TPSTATUS.  
  
CALL "TPGETLEV" USING TPTRXLEV-REC TPSTATUS-REC.
```

- 戻り値

成功すれば、TP-STATUSにはTPOK、TPTRXLEV-RECにはトランザクション・モードではない場合はTP-NOT-IN-TRAN、トランザクション・モードの場合はTP-IN-TRANが設定されます。

2.2.10. TPGETRPLY

TPGETRPLYは、以前にTPACALLで非同期的に要求したサービスに対する応答を受信する関数です。

関数は、COMM-HANDLEと一致する応答が受信されるまで、あるいはタイムアウトが発生するまで待機します。一般的に、COMM-HANDLEは応答が受信された後は無効になります。

- 使用方法

```
01 TPSVCDEF-REC.  
COPY TPSVCDEF.  
01 TPTYPE-REC.  
COPY TPTYPE.  
01 DATA-REC.  
COPY User data.  
01 TPSTATUS-REC.  
COPY TPSTATUS.  
  
CALL "TPGETRPLY" USING TPSVCDEF-REC TPTYPE-REC DATA-REC TPSTATUS-REC.
```

- データを正常に受信した場合、LENはDATA-RECに実際に格納されたデータの長さが入ります。

LENが0で返された場合、応答は何も受信されず、LENが指示するODATA-RECバッファに何の変化もありません。LENに0がINPUTで利用することはできません。応答がDATA-RECより大きい場合、DATA-RECにはこのサイズ分のデータのみが入ります。応答の残りは無視され、TPTRUNCATEが設定されます。

– TPTYPE-RECのREC-TYPEとSUB-TYPEには、送信された各データのタイプと下位タイプが入ります。

– 以下は、TPSVCDEF-RECに設定が可能な値です。

設定値	説明
TPGETANY	入力値で指定したCOMM-HANDLE in TPSVCDEF-RECを無視し、受信可能な応答を返します。 COMM-HANDLEは応答に対する呼び出しの記述子になります。応答がない場合、一般的にTPGETRPLYは応答の到着を待機します
TPNOCHANGE	受信した応答バッファとREC-TYPEが指すバッファのタイプが異なる場合、ODATA-RECのバッファのタイプは、受信側が認識できる限度内で受信された応答バッファのタイプに変更されます。フラグが設定されている場合、REC-TYPEが指すバッファのタイプは変更できません。受信された応答バッファのタイプおよび下位タイプは、指すバッファのタイプおよび下位タイプと一致する必要があります
TPNOBLOCK	TPNOBLOCKが設定されている場合、応答が到着するまで待機しません。受信可能な応答があれば、これを返します。TPNOBLOCKが指定されておらず、受信可能な応答がない場合、関数の呼び出し元は応答が到着するまで、あるいはタイムアウト(トランザクション・タイムアウトまたはブロッキング・タイムアウト)が発生するまで待機します
TPNOTIME	TPNOTIMEは、関数の呼び出し元がブロッキング・タイムアウトを無視し、応答が受信するまで無限で待機することを意味します。トランザクション・モードでTPGETRPLY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	シグナルの割り込みを受け入れる場合に使用します。システム関数の呼び出しが妨害された場合、システム関数の呼び出しが再実行されます

- 戻り値

実行結果	説明
成功	正常に処理された場合は1を返し、TPRETURNで渡されたAPPL-RETURN-CODEのグローバル変数は、TPGETRPLYが正常に返された場合、あるいはTP-STATUSが[TPESVCFAIL]の場合、アプリケーション・プログラムで定義した値が入ります

実行結果	説明
失敗	-1を返し、TP-STATUSにエラー状況に該当する値が設定されます

● エラー

関数の実行に失敗した場合、TP-STATUSに以下の値が設定されます。

エラーコード	説明
[TPEINVAL]	パラメータが有効ではありません。たとえば、SERVICE-NAMEがSPACESの場合や、TPSVCDEF-RECが正常でない場合です
[TPEBADDESC]	COMM-HANDLEが無効な記述子です
[TPEOTYPE]	受信した応答のタイプまたは下位タイプが、呼び出し元が認識できないタイプです。TPNOCHANGEが設定されていて、REC-TYPEとSUB-TYPEが受信したデータのタイプまたは下位タイプと一致しません。DATA-RECとLENは両方変更されません。応答が呼び出し元のトランザクション・モードで受信した場合、そのトランザクションは応答が無視されるためロールバックされます
[TPETIME]	<p>タイムアウトが発生しました。</p> <p>関数の呼び出し元がトランザクション・モードの場合、トランザクション・タイムアウトが発生し、そのトランザクションはロールバックします。トランザクション・モードではなく、TPNOTIMEとTPNOBLOCKのどちらも指定されていない場合、ブロッキング・タイムアウトが発生します。</p> <p>この2つの場合、DATA-RECの内容とLENは変更されません。トランザクション・タイムアウトが発生した場合、新しいサービスの要求を受信したり応答を待機したりすることは、トランザクションがロールバックするまで[TPETIME]エラーで失敗します</p>
[TPESVCFAIL]	<p>サービスの要求に対する応答を送信するサービス・ルーチンが、アプリケーション・プログラム上のエラーが発生し、TPFAILでTPRETURNを呼び出しました。サービスの応答を受信した場合、その内容はDATA-RECDPに指定され、使用されます。関数の呼び出し元がトランザクション・モードの場合、そのトランザクションはロールバックされます。</p> <p>トランザクション・タイムアウトが発生するまでは、トランザクションがロールバックする前に他の通信が行われることがあります。このような通信は通常に処理されることもあれば、失敗することもあります。実行するためには、TPNOTRANが設定されている必要があります。呼び出し元のトランザクション・モードで実行された作業は、トランザクションが完了した場合、すべてロールバックされます</p>
[TPESVCERR]	有効でない記述子を使用した場合、トランザクション・モードでXAオペレーションが失敗した場合に発生します(TXBEGIN、TXCOMMIT、TXROLLBACK)

エラーコード	説明
[TPEBLOCK]	TPNOBLOCKが設定された状態でブロッキング状況が発生した場合で、記述子は有効です
[TPGOTSIG]	TPSIGRSTRTが設定されていない状態でシグナルを受信しました
[TPEPROTO]	TPGETRPLYが不適切な状況で呼び出されました
[TPESYSTEM]	Tmaxシステムにエラーが発生しました
[TPEOS]	OSにエラーが発生しました

- 参照

TPACALL, TPCANCEL, TPRETURN

2.2.11. TPRETURN

Tmaxサービス終了の関数で、サービス・ルーチンの完了を意味します。TPRETURNはEXIT PROGRAM statementと同じ役割として、TPRETURNが呼び出された場合、サービス・ルーチンはTmaxシステムに戻されます。

Tmaxシステムに正しく返されるためには、TPRETURNはTmaxシステムが制御するサービス・ルーチン内で呼び出されることが望ましいです。サービス・ルーチンが自動的にトランザクション・モードにありますが、該当サービスを呼び出したクライアントが明示的にトランザクションを開始しないため、(つまり、TXBEGINを使用しなければ)TPRETURNはトランザクションの一部としてTPSUCCESSを使用するとコミット、TPFAILを使用するとロールバックされます。サービスは同じトランザクション(Global Transaction)の一部として、複数回呼び出されることもあります。そのため、TXBEGINを使用したトランザクションの開始側がTXCOMMITまたはTXROLLBACKのうち1つを呼び出し、トランザクションを完了するまでは完全にコミットまたはロールバックされません。

TPRETURNは、サービスの応答メッセージを送信します。応答を受信するプログラムがTPCALL、TPGETRPLY、またはTPRECVで応答を待機している場合、その応答はTPRETURNの呼び出しが成功した後に、受信側のバッファーを通じて渡されます。呼び出しに失敗した場合、サービスの特性に従って、[TPESVCERR]エラーやTPEV_SVCERRイベントがサービス・ルーチンと通信するプログラムに戻されます。

- 使用方法

```
01 TPSVCRET-REC.
COPY TPSVCRET.
01 TPTYPE-REC.
COPY TPTYPE.
01 DATA-REC.
COPY User data.
01 TPSTATUS-REC.
```

```

COPY TPSTATUS.
COPY TPRETURN REPLACING TPSVCRET-REC BY TPSVCRET-REC
TPTYPE-REC BY TPTYPE-REC
DATA-REC BY DATA-REC
TPSTATUS-REC BY TPSTATUS-REC.

```

- 以下は、TP-RETURN-VALの引数として設定が可能な値です。以下に存在しないTP-RETURN-VAL値はすべてTPFAILと見なされます。

設定値	説明
TPSUCCESS	<p>サービスが正常に終了しました。データが存在し、TPRETURNの実行中にエラーが発生しなければ、データは送信されます。</p> <p>呼び出し元がトランザクション・モードの場合、このトランザクションの一部をコミット可能な状態に決定します。トランザクションが最終的に完了した場合、トランザクションに属する残りのサービスがすべて正常に完了し、コミット可能な状態の場合はコミット、1つでも失敗した場合はロールバックされます。TPRETURNの呼び出しは、必ずしもトランザクション全体の終了にはつながりません。</p> <p>呼び出し元がTPSUCCESSでTPRETURNを行っても、応答や会話型サービスが存在する場合や、あるいはサービスで実行された作業がトランザクションをロールバックすると、サービス失敗としてメッセージが送信されます。応答の受信側は、[TPESVCERR]表示あるいはTPEV_SVCERRイベントを受信します。サービス・ルーチン内でトランザクションがロールバックされると、TP-RETURN-VALはTPFAILに設定されます。</p> <p>会話型サービスでTPSUCCESSで返されると、TPEV_SVCSUCCイベントが発生します</p>
TPFAIL	<p>サービスがアプリケーション・プログラムの失敗で終了しました。応答を受け取るプログラムにエラーが返されます。応答を受け取る呼び出しが失敗し、受信側は[TPSVCFAIL]値やTPEV_SVCFAILイベントを受け取ります。</p> <p>呼び出し元がトランザクション・モードであり、自動トランザクションの場合、tpreturn()はトランザクションをロールバックします。そのトランザクションがすでにロールバック状態で決定されていることもあります。このパラメータはデータを送信できません</p>
TPEXIT	<p>サービスの呼び出し後に返す場合、サーバー・プロセスを強制終了する際に使用します。TPEXITで終了したプロセスは、RESTART=Yが設定されている場合、TMMIによって再度自動的に起動します</p>
TPDOWN	<p>TPEXITと似ていますが、TPDOWNで終了したプロセスはTMMIによって再度起動しません</p>

- アプリケーション・プログラムでユーザーによって定義されたリターンコード値のAPPL-CODEは、サービスの応答を受け取るプログラムに送信されます。TP-RETURN-VALの値と関係なく、応答がクライア

ントに送信できる場合、つまり受信の呼び出しが正常に行われた場合か、[TPSVCFAIL]で返された場合、あるいはTPEV_SVCSUCCまたはTPEV_SVCFAILイベントのうち1つを受信した場合に送信されます。APPL-CODEから転送されるデータは、TPSTATUS-RECのAPPL-RETURN-CODEに転送されて使用できます。

- DATA-RECは送信されるデータで、LENは送信されたデータの長さです。データがNULLの場合、LENは無視されます。この場合、サービスを呼び出したプログラムが応答を期待していれば、データがない応答が送信されます。応答を期待していない場合、TPRETURNはいかなるデータも削除し、送信する応答なしで返します。

DATA-RECが長さの明示が必要ないバッファを指す場合、LENは無視され、通常0を使用します。REC-TYPEがSPACESの場合、DATA-RECとLENは無視されます。DATA-RECが長さの明示が必要なバッファを指す場合、LENは0になることはできません。REC-TYPEがSTRINGでありLENが0の場合、要求はデータなしで送信されます。

- 戻り値

サービス・ルーチンは、呼び出し元のTmaxシステムにいかなる値も返しません。サービス・ルーチンはTPRETURN(EXIT PROGRAMを含む)を使用して終了することが原則です。

サービス・ルーチンがTPRETURNを使用せずに、COBOLのEXIT PROGRAM文を使用して返す場合、サーバーはサービスのリクエストにサービス・エラーを返します。TPRETURNがサービス・ルーチン外部で使用された場合、たとえばサービスではないルーチンで使用された場合、何もせずに返します。サーバーがトランザクション・モードの場合、そのトランザクションはロールバックが表示されます。

- エラー

TPRETURNがサービス・ルーチンを終了させるため、パラメータを処理中にエラーが発生すると、呼び出し元のサービス・ルーチンが知ることはできません。

同期と非同期通信(TPCALLまたはTPGETRPLYでサービス結果を受信するプログラム)に対しては、TP-STATUSが[TPESVCERR]になります。対話型通信(TPSENDやTPRECVを使用するプログラム)に対しては、TPEV_SVCERRイベントが発生します。

- 参照

TPCALL, TPGETRPLY

2.2.12. TPSVCSTART

Tmaxサービスを開始する関数で、サービス・ルーチンを作成する際に最初に使用されるTmaxシステム・ルーチンです。

サービス・ルーチンでTPSVCSTARTを呼び出す前に他のことを実行すると、エラーになります。TPVCSTARTは、サービスのパラメータとデータを取得するのに使用します。このルーチンは、TPCALLやTPACALLを使用して受信した要求をサービスで使用できるようにします。

- 使用方法

```
01 TPSVCDEF-REC.  
COPY TPSVCDEF.  
01 TPTYPE-REC.  
COPY TPTYPE.  
01 DATA-REC.  
COPY User data.  
01 TPSTATUS-REC.  
COPY TPSTATUS.  
CALL "TPSVCSTART" USING TPSVCDEF-REC TPTYPE-REC DATA-REC TPSTATUS-REC.
```

- 戻り値

実行結果	説明
成功	TP-STATUSにTPOKを設定します、受信されたメッセージが定義されたLENより大きい場合はTPTRUNCATEが設定され、LEN分のデータのみDATA-RECに移動され、残りは無視されます
失敗	TP-STATUSにエラーコード値が設定されます

- エラー

関数の実行に失敗した場合、TP-STATUSに以下の値が設定されます。

エラーコード	説明
[TPEINVAL]	パラメータが有効ではありません
[TPEPROTO]	TPSVCSTARTが不適切な状況で呼び出されました
[TPESYSTEM]	Tmaxシステムにエラーが発生しました
[TPEOS]	OSにエラーが発生しました

- 参照

TPSVRINIT, TPSVRDONE, TXBEGIN, TPCALL

2.2.13. TPSVRDONE

サーバー・プロセスを終了する関数で、Tmaxのアプリケーション・サーバーでサービスの要求処理をすべて終え、プロセスが終了する前に呼び出されます。このルーチンが実行される際、そのサーバー・プロセスは

依然としてシステムの一部ではありますが、サービスはサポートしません。そのため、TPSVRDONEルーチン内でTmax通信が実行される場合や、トランザクションが定義される場合もあります。

アプリケーション・プログラムでTPSVRDONEルーチンを提供していない場合、Tmaxが提供するデフォルト・ルーチンが代わりに呼び出されます。デフォルトのTPSVRDONEはトランザクションを処理するサーバー・グループに含まれているサーバーで、TXCLOSEとUSERLOGを呼び出して、サーバーが終了することを通知します。TPRETURNかTPFORWARDのうち1つがTPSVRDONE内で呼び出された場合、ルーチンは何も作動せずに、単純に返します。

TPSVRDOENが会話型接続を維持している場合や、非同期応答を依然として待機している場合や、トランザクション・モードにある間に返す場合、Tmaxは会話型接続を終了し、待機していた非同期応答などを無視し、トランザクションを中止して、サーバーは即座に終了します。

- 使用方法

```
01 TPSTATUS-REC.  
COPY TPSTATUS.  
  
PROCEDURE DIVISION.  
* User code  
EXIT PROGRAM.
```

- 参照

TPSVRINIT

2.2.14. TPSVRINIT

Tmaxサーバーの初期化関数で、Tmaxシステムはサーバーの初期化過程でTPSVRINITを呼び出します。

プロセスが起動後、いかなるサービス要求も処理される前に呼び出されます。そのため、TPSVRINITルーチン内でTmax通信が実行される場合や、トランザクションが定義される場合もあります。アプリケーション・プログラムでTPSVRINITルーチンを提供していない場合、Tmaxが提供するデフォルト・ルーチンが代わりに呼び出されます。基本的なTPSVRINITは、トランザクションを処理するサーバー・グループに含まれたサーバーの場合、TXOPENとUSERLOGを呼び出し、サーバーが正常に開始したことを通知します。

アプリケーション・プログラムのコマンド・ライン・オプション(CLOPT)は、ARGCとARGVを使用して渡されます。ARGCにはパラメータの数が渡され、ARGVはSPACE文字(1つ)で区切られた引数を含みます。

TPRETURNかTPFORWARDのうち1つがTPSVRDONE内で呼び出された場合、これらのルーチンは何も作動せずに、単純に返します。

- 使用方法

```
LINKAGE SECTION.  
01 CMD-LINE.
```

```

05 ARGV.
10 ARGV PIC X OCCURS 0 TO 9999 DEPENDING ON ARGV.
01 TPSTATUS-REC.
COPY TPSTATUS.

PROCEDURE DIVISION USING CMD-LINE TPSTATUS-REC.
* User code
EXIT PROGRAM.

```

- 戻り値

実行結果	説明
成功	TP-STATUSはTPOKを渡し、サービスは正常に要求を受け取ることができるようになります
失敗	TP-STATUSに-1を渡し、サーバーは終了し、いかなるサービスの要求も受け取ることができません

- 参照

TPSVRDONE

2.2.15. TXBEGIN

グローバル・トランザクションを開始する関数で、関数の呼び出し元はトランザクション・モードになります。

Tmaxシステムでトランザクションはすべて成功あるいはすべて失敗するようにする1つの論理的な作業単位を定義するのに使用します。呼び出しプロセスは、トランザクションを開始する前に、TXOPENでサポート・マネージャーと接続されている必要があります。TXBEGIN関数は、呼び出し元がすでにトランザクション・モードである場合、あるいはTXOPENが呼び出されていない場合に失敗し、[TX-PROTOCOL-ERROR]を返します。

トランザクションが開始されると、呼び出しプロセスは現在のトランザクションを完了させるために、TPCOMMITまたはTPROLLBACKを呼び出す必要があります。トランザクションを開始するためには、必ずしもTXBEGINを明示的に呼び出す必要がない連鎖(chaining)トランザクションの場合も存在します。詳細については、TPCOMMITあるいはTPROLLBACKを参照してください。

- 使用方法

```

01 TPTRXDEF-REC.
COPY TPTRXDEF.
CALL "TXBEGIN" USING TPTRXDEF-REC.

```

- 戻り値

実行結果	説明
成功	TX-STATUSはTX-OKに設定されます
失敗	TX-STATUSにエラーコード値が設定されます

- エラー

関数の実行に失敗した場合、TX-STATUSに以下の値が設定されます。

エラーコード	説明
[TX-OUTSIDE]	現在呼び出しプロセスが外部のトランザクションに参加しているため、トランザクション・マネージャーがグローバル・トランザクションを開始できません。このような作業がすべて完了してからでなければ、グローバル・トランザクションを開始できません。参加しているトランザクションは影響を与えません
[TX-PROTOCOL-ERROR]	関数が不適切な状況で呼び出されました。たとえば、呼び出し元がすでにトランザクション・モードにある場合です。現在のトランザクションには影響を与えません
[TX-ERROR]	トランザクション・マネージャーまたはリソース・マネージャーがトランザクションの開始において一時的にエラーを検出しました。このエラーが返された場合、呼び出し元はトランザクション・モードにありません。エラーの正確な原因は製品の特性によって決定されます
[TX-FAIL]	致命的なエラーが発生して、トランザクション・マネージャーあるいはリソース・マネージャーは、今後アプリケーション・プログラムのために作業を行うことができません。このエラーが返された場合、呼び出し元はトランザクション・モードにありません。エラーの正確な原因は製品の特性によって異なります

- 参照

TPROLLBACK, TPCOMMIT, TPGETLEV

2.2.16. TXCOMMIT

グローバル・トランザクションをコミットする関数です。

transaction-control特性がTX-UNCHAINEDの場合、TXCOMMITが返す際に呼び出し元はトランザクション・モードでなくなります。transaction-control特性がTX-CHAINEDの場合、TXCOMMITが返される際、呼び出し元は新しいトランザクションのためにトランザクション・モードのままになります。正常に完了すると、TX-RETURN-STATUSが戻り値に使用されます。

- 使用方法


```

DATA DIVISION.
* Include TX definitions.
01 TX-RETURN-STATUS.
COPY TXSTATUS.

PROCEDURE DIVISION.
CALL "TXCOMMIT" USING TX-RETURN-STATUS.

```

- 戻り値

実行結果	説明
成功	TXCOMMITは負数ではないTX-OKを返します
失敗	TX-STATUSにはエラーコードの負数値を返します

- エラー

関数の実行に失敗した場合、TX-STATUSに以下の値が設定されます。

エラーコード	説明
[TX-NO-BEGIN]	トランザクションが正常にコミットされました。 新しいトランザクションは開始できず、呼び出し元はトランザクション・モードではなくなりました。戻り値はtransaction-control特性がTX-CHAINEDである場合のみ発生します
[TX-ROLLBACK]	トランザクションがロールバックされました。transaction-control特性がTX-CHAINEDの場合、新しいトランザクションが開始されます
[TX-ROLLBACK-NO-BEGIN]	トランザクションがロールバックされました。新しいトランザクションは開始できず、呼び出し元はトランザクション・モードではなくなりました。戻り値はtransaction-control特性がTX-CHAINEDである場合のみ発生します
[TX-MIXED]	トランザクションが一部はコミットされ、一部はロールバックされました。transaction-control特性がTX-CHAINEDの場合、新しいトランザクションが開始されます
[TX-MIXED-NO-BEGIN]	トランザクションが一部はコミットされ、一部はロールバックされました。新しいトランザクションは開始できず、呼び出し元はトランザクション・モードではなくなりました。戻り値はtransaction-control特性がTX-CHAINEDである場合のみ発生します
[TX-HAZARD]	エラーのために、トランザクションが一部はコミットされましたが、一部はロールバックされた可能性があります。transaction-control特性がTX-CHAINEDの場合、新しいトランザクションが開始されます

エラーコード	説明
[TX-HAZARD-NO-BEGIN]	エラーのために、トランザクションの一部はコミットされましたが、一部はロールバックされた可能性があります。新しいトランザクションは開始できず、呼び出し元はトランザクション・モードではなくなりました。戻り値はtransaction-control特性がTX-CHAINEDの場合のみ発生します
[TX-PROTOCOL-ERROR]	関数が不適切な状況で呼び出されました。たとえば、呼び出し元がトランザクション・モードにない場合です。トランザクションに関連する呼び出し元の状態は変更されません
[TX-FAIL]	致命的なエラーが発生し、トランザクション・マネージャーあるいはリソース・マネージャーがアプリケーション・プログラムのために作業を行うことができなくなりました。エラーの正確な原因は製品の特性によって異なります。トランザクションに関連する呼び出し元の状態は不明です

- 参照

TXBEGIN, TXSETCOMMITRET, TXSETTRANCTL, TXSETTIMEOUT

2.2.17. TXINFORM

TX-INFO-AREAを使用してグローバル・トランザクション情報を返す関数です。

トランザクション・モードでTXINFORMが実行されると、TX-IN-TRANが設定され、XID-RECは現在のトランザクションのブランチ識別子になり、TRANSACTION-STATEは現在のトランザクションの状態になります。呼び出し元がトランザクション・モードでない場合、TX-NOT-IN-TRANが設定され、XID-RECはNULL XIDが入ります。

また、呼び出し元がトランザクション・モードにあるかどうかに関係なく、COMMIT-RETURN、TRANSACTION-CONTROL、そして秒単位のTRANSACTION-TIMEOUT値を含みます。

返されたトランザクション・タイムアウト値は、次のトランザクションの開始時から使用します。それは、呼び出し元の現在のグローバル・トランザクションのタイムアウト値ではないこともあります。現在のトランザクションが開始後に行われたTXSETTIMEOUTの呼び出しによって、TRANSACTION-TIMEOUT値を変更されていることがあるためです。

- 使用方法

```
DATA DIVISION.
* Include TX definitions.
01 TX-RETURN-STATUS.
COPY TXSTATUS.
01 TX-INFO-AREA.
COPY TXINFDEF.
```

```
PROCEDURE DIVISION.  
CALL "TXINFORM" USING TX-INFO-AREA, TX-RETURN-STATUS.
```

- 戻り値

実行結果	説明
成功	TXINFOは負数でないTX-OKを返します。呼び出し元がトランザクション・モードにある場合は1を返し、呼び出し元がトランザクション・モードにない場合は0を返します
失敗	以下の負数値を返します

- エラー

関数の実行に失敗した場合、以下の値が設定されます。

エラーコード	説明
[TX-PROTOCOL-ERROR]	関数が不適切な状況で呼び出されました。たとえば、呼び出し元がまだTXOPENを呼び出していない場合です
[TX-FAIL]	致命的なエラーが発生し、トランザクション・マネージャーはアプリケーション・プログラムのための作業を実行できなくなりました。エラーの正確な原因は製品の特性によって異なります

- 参照

TXOPEN, TXSETCOMMITRET, TXSETTRANCTL, TXSETTIMEOUT

2.2.18. TXROLLBACK

グローバル・トランザクションをロールバックします。transaction-control特性(TXSETTRANSACTIONCONTROL)がTX-UNCHAINEDの場合、TXROLLBACKが返されると、呼び出し元はトランザクション・モードではなくなります。ただし、transaction-control特性がTX-CHAINEDの場合、TXROLLBACKが返されると、呼び出し元は新しいトランザクションのためにトランザクション・モードのままになります。

- 使用方法

```
01 TPTRXDEF-REC.  
COPY TPTRXDEF.  
01 TPSTATUS-REC.  
COPY TPSTATUS.  
CALL "TXROLLBACK" USING TPTRXDEF-REC.
```

- 戻り値

実行結果	説明
成功	TX-STATUSはTP-OKに設定されます
失敗	TX-STATUSにはエラーコード負数値が設定されます

- エラー

関数の実行に失敗した場合、TX-STATUSに以下の値が設定されます。

エラーコード	説明
[TX-NO-BEGIN]	トランザクションがロールバックされました。新しいトランザクションは開始できず、呼び出し元はトランザクションモードではなくなります。戻り値はtransaction-control特性がTX-CHAINEDの場合にのみ発生します
[TX-MIXED]	トランザクションが一部はコミットされ、一部はロールバックされました。transaction-control特性がTX-CHAINEDの場合、新しいトランザクションを開始します
[TX-MIXED-NO-BEGIN]	トランザクションが一部はコミットされ、一部はロールバックされました。新しいトランザクションは開始できず、呼び出し元はトランザクション・モードではなくなります。戻り値はtransaction-control特性がTX-CHAINEDの場合にのみ発生します
[TX-HAZARD]	エラーにより、トランザクションが一部はコミットされ、一部はロールバックされた可能性があります。transaction-control特性がTX-CHAINEDの場合、新しいトランザクションが開始されます
[TX-HAZARD-NO-BEGIN]	エラーにより、トランザクションが一部はコミットされ、一部はロールバックされた可能性があります。新しいトランザクションは開始されず、呼び出し元はトランザクション・モードではなくなります。戻り値はtransaction-control特性がTX-CHAINEDの場合にのみ発生します
[TX-COMMITTED]	トランザクションが動的にコミットされました。transaction-control特性がTX-CHAINEDの場合、新しいトランザクションが開始されます
[TX-COMMITTED-NO-BEGIN]	トランザクションがヒューリスティックにコミットされました。新しいトランザクションは開始できず、呼び出し元はトランザクション・モードではなくなります。戻り値はtransaction-control特性がTX-CHAINEDの場合のみ発生します
[TX-PROTOCOL-ERROR]	関数が不適切な状況で呼び出されました。たとえば、呼び出し元がトランザクション・モードではなくなります。トランザクションに関連する呼び出し元の状態は変化がありません
[TX-FAIL]	致命的なエラーが発生し、トランザクション・マネージャーやリソース・マネージャーはアプリケーション・プログラムのための作業を実行できません

エラーコード	説明
	せん。エラーの性格な原因は製品の特性によって異なります。トランザクションに関連する呼び出し元の状態は不明です

- 参照

TXBEGIN, TPCOMMIT, TPGETLEV, TPACALL

2.2.19. TXSETCOMMITRET

commit-return特性をCOMMIT-RETURNに設定する関数です。TXCOMMITが関数の呼び出し元に制御権を返す方式を決定します。TXSETCOMMITRETは関数の呼び出し元がトランザクション・モードであることに関係なく呼び出しが可能です。TXSETCOMMITRETの再呼び出しによって変更されるまで設定は有効です。

- 使用方法

```
DATA DIVISION.
* Include TX definitions.
01 TX-RETURN-STATUS.
COPY TXSTATUS.
01 TX-INFO-AREA.
COPY TXINFDEF.

PROCEDURE DIVISION.
CALL "TXSETCOMMITRET" USING TX-INFO-AREA TX-RETURN-STATUS.
```

以下は、commit-return特性としてCOMMIT-RETURNに設定が可能な値です。

設定値	説明
TX-COMMIT-DECISION-LOGGED	<p>TXCOMMITが2PC(Two-Phase Commit)プロトコルのうち1相目でロギングされた後、2相目は完了する前に返すようにします。</p> <p>TXCOMMITが呼び出し元に速く応答できます。しかし、トランザクションがヒューリスティックな結果を得る恐れがあります。その場合、TXCOMMITから返されたコードでは呼び出し元が状況を正確に把握することはできません。正常な場合、1相目でトランザクションをコミットすることにしたトランザクションの参加者は、2相目で正確にコミットすることになります。ネットワークやノード障害が長時間継続するなど正常でない場合は2相目が完了しないので、ヒューリスティックな結果が発生することもあります。</p>

設定値	説明
	トランザクション・マネージャーはオプションでこの特性をサポートするように選択でき、この際にこの値をサポートしないことを示す[TX-NOT-SUPPORTED]値で変換します
TX-COMMIT-COMPLETED (デフォルト値)	2PCプロトコルが完全に終了後、TXCOMMITが返されます。 トランザクションがヒューリスティックな結果を得るか、またはその可能性を知らせるリターンコードをTXCOMMITの呼び出し元に渡します。トランザクション・マネージャーはオプションでこの特徴をサポートしないよう選択することができます。この際、この値をサポートしないことを示す[TX-NOT-SUP-PORTED]値で返します

- 戻り値

作業が正常に完了した場合、TXSETCOMMITRETは負数ではないTX-OKを返します。

COMMIT-RETURNがTX-COMMIT-COMPLETEDまたはTX-COMMIT-DECISION-LOGGEDに設定されていない場合、関数は負数ではない値で[TX-NOT-SUPPORTED]を返します。commit-return特性は現在適用されている値が有効です。トランザクション・マネージャーはCOMMIT-RETURNを少なくともTX-COMMIT-COMPLETEDまたはTX-COMMIT-DECISION-LOGGEDのうち1つには設定する必要があります。

- エラー

以下のような状況で、TXSETCOMMITRETはcommit-return特性の設定を変更せず、以下のいずれかの負数の値を返します。

エラーコード	説明
[TX-EINVAL]	COMMIT-RETURNがTX-COMMIT-COMPLETEDあるいはTX-COMMIT-DECISION-LOGGEDに設定されていません
[TX-PROTOCOL-ERROR]	関数が不適切な状況で呼び出されました。たとえば、TXOPENがまだ呼び出されていない状態です

- 参照

TXCOMMIT, TXOPEN, TXINFORM, TXGBEGIN, TXROLLBACK

2.2.20. TXSETTIMEOUT

TXSETTIMEOUTは、transaction-control特性をタイムアウト値として設定する関数です。この値はトランザクション・タイムアウトが発生する前にトランザクションを完了しなければならない時間で、TXBEGINとTXCOMMITまたはTXBEGINとTXROLLBACKの間の時間になります。

TXSETTIMEOUTは関数の呼び出し元がトランザクション・モードであるか否かに関係なく呼び出しが可能です。TXSETTIMEOUTがトランザクション・モードで呼び出された場合、新しいタイムアウト値は次のトランザクションから適用されます。TRANSACTION-TIMEOUTの初期値は0で、タイムアウトの制限がないことを意味します。

TRANSACTION-TIMEOUTは、トランザクション・タイムアウトが発生するまでに許容された時間を秒単位の数字で指定します。システム別に定義されたS9(9) COMP-5タイプの最大値まで設定可能です。

● 使用方法

```
DATA DIVISION.  
* Include TX definitions.  
01 TX-RETURN-STATUS.  
COPY TXSTATUS.  
*  
01 TX-INFO-AREA.  
COPY TXINFDEF.  
  
PROCEDURE DIVISION.  
CALL "TXSETTIMEOUT" USING TX-INFO-AREA TX-RETURN-STATUS.
```

● 戻り値

実行結果	説明
成功	TXSETTIMEOUTに負数でないTX-OKを返します
失敗	TXSETTIMEOUTは既存のTRANSACTION-TIMEOUT値の変更なく、エラーコードの負数値のうち1つを返します

● エラー

関数の実行に失敗した場合、TXSETTIMEOUTに以下の値が設定されます。

エラーコード	説明
[TX-EINVAL]	指定されたタイムアウト値が無効です
[TX-PROTOCOL-ERROR]	関数が不適切な状況で呼び出されました。たとえば、tx-open()がまだ呼び出されていません
[TX-FAIL]	致命的なエラーが発生し、トランザクション・マネージャーがアプリケーション・プログラムのための作業を実行できなくなります。エラーの正確な原因は製品の特性によって異なります

- 参照

TXBEGIN, TXCOMMIT, TXOPEN, TXROLLBACK, TXINFORM

2.2.21. TXSETTRANCTL

TXSETTRANCTLは、transaction-control特性をTRANSACTION-CONTROLに設定された値で設定する関数です。

TXCOMMITとTXROLLBACKが呼び出し元に返す前に新しいトランザクションを開始するか否かを決定します。TXSETTRANCTL関数はアプリケーション・プログラムがトランザクション・モードであるか否かに関係なく呼び出しが可能です。この設定はTXSETTRANCTLが再呼び出しによって変更されるまで有効です。

- 使用方法

```
DATA DIVISION.
* Include TX definitions.
01 TX-RETURN-STATUS.
COPY TXSTATUS.
*
01 TX-INFO-AREA.
COPY TXINFDEF.

PROCEDURE DIVISION.
CALL "TXSETTRANCTL" USING TX-INFO-AREA TX-RETURN-STATUS.
```

以下は、transaction-control特性としてTRANSACTION-CONTROLに設定可能な値です。

設定値	説明
TX-UNCHAINED(デフォルト値)	TXCOMMITとTXROLLBACKが呼び出し元に返す前に新しいトランザクションを開始しないようにします。呼び出し元は、新しいトランザクションを開始するにはTXBEGINを実行します
TX-CHAINED	TXCOMMITとTXROLLBACKが呼び出し元に返す前に新しいトランザクションを開始します

- 戻り値

実行結果	説明
成功	TXSETTRANCTLは負数ではないTX-OKを返します
失敗	TXSETTRANCTLは既存のtransaction_control特性を変更することなく、エラーコードの負数値を返します

- エラー

関数の実行に失敗した場合、TXSETTRANCTLに以下の値が設定されます。

エラーコード	説明
[TX-EINVAL]	指定されたタイムアウト値が有効ではありません
[TX-PROTOCOL-ERROR]	関数が不適切な状況で呼び出されました。たとえば、呼び出し元ではないtx-open()を呼び出しました
[TX-FAIL]	致命的なエラーが発生し、トランザクション・マネージャーがアプリケーション・プログラムのための作業を実行できません。エラーの正確な原因は製品の特性によって異なります

- 参照

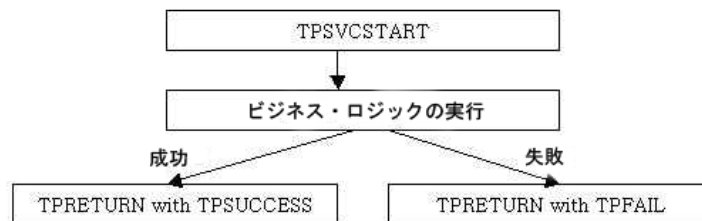
TXBEGIN, TXCOMMIT, TXOPEN, TXROLLBACK, TXINFORM

第3章 例

本章では、サービス・プログラム、グローバル・トランザクション・プログラムなどのサンプル・プログラムについて説明します。

3.1. サービス・プログラム

以下は、サービス・プログラムの作成手順です。



3.1.1. サンプル・プログラム

以下は、サービス・プログラムの例です。

参考

サンプル・プログラムで説明するサービスは、XAサーバーグループに属するサーバーとしてTmaxのAUTOTRAN機能を利用するため、Tmax環境ファイルで該当サービスに対して“AUTOTRAN=Y”設定が必要です。

<\$TMAXDIR/sample/cobserver/carrayauto.pco>

```
*****
* XAでありながらCARRAYバッファを使用するサービス      *
*****

IDENTIFICATION          DIVISION.
PROGRAM-ID.             CARRAYAUTO.
AUTHOR.                 TMAX DEVELOPMENT.
ENVIRONMENT             DIVISION.
CONFIGURATION           SECTION.

DATA                    DIVISION.
FILE                   SECTION.
```

WORKING-STORAGE SECTION.

* TMAX ATMI関数のためのdefinitions

01 TPSVCRET-REC.

COPY TPSVCRET.

*

01 TPTYPE-REC.

COPY TPTYPE.

*

01 TPSTATUS-REC.

COPY TPSTATUS.

*

01 TPSVCDEF-REC.

COPY TPSVCDEF.

*

01 TX-RETURN-STATUS.

COPY TXSTATUS.

* Log message definitions

01 LOGMSG.

05 LOGMSG-SVC PIC X(16).

05 FILLE PIC X(3) VALUES " : ".

05 LOGMSG-TEXT PIC X(80).

01 LOGMSG-LEN PIC S9(9) COMP-5.

* User defined data records

01 RECV-STRING PIC X(100).

01 SEND-STRING PIC X(100).

* ユーザ一定義ORACLE Host変数の宣言

EXEC SQL BEGIN DECLARE SECTION END-EXEC.

01 S-SRC.

05 S-TEMP PIC X(10) VARYING.

EXEC SQL END DECLARE SECTION END-EXEC.

EXEC SQL INCLUDE SQLCA END-EXEC.

```

LINKAGE                                SECTION.
*
PROCEDURE                              DIVISION.
*
000-MAIN.

PERFORM 100-SVCSTART      THRU 100-EXIT.
PERFORM 200-DBINSERT      THRU 200-EXIT.
PERFORM 900-TPRETURNSCS  THRU 900-TPRETURNSCS-X.

000-EXIT.

*****
* クライアントから送信されたデータを受信
*****

100-SVCSTART.
    MOVE LENGTH OF RECV-STRING TO LEN.
    CALL "TPSVCSTART" USING TPSVCDEF-REC
                                TPTYPE-REC
                                RECV-STRING
                                TPSTATUS-REC.

    IF NOT TPOK
        MOVE "TPSVCSTART Fail" TO LOGMSG-TEXT
        PERFORM 800-USERLOG
        PERFORM 900-TPRETURNFAIL THRU 900-TPRETURNFAIL-X
    END-IF.

    IF TPTRUNCATE
        MOVE "Received data was truncated" TO LOGMSG-TEXT
        PERFORM 800-USERLOG
        PERFORM 900-TPRETURNFAIL THRU 900-TPRETURNFAIL-X
    END-IF.

    DISPLAY "-----".

    MOVE SERVICE-NAME TO LOGMSG-SVC.
    MOVE "サービス開始 " TO LOGMSG-TEXT.
    PERFORM 800-USERLOG.
    100-EXIT.

200-DBINSERT.
    MOVE RECV-STRING(1:LEN) TO S-TEMP-ARR.
    MOVE LENGTH OF S-TEMP-ARR TO S-TEMP-LEN.

* DB INSERT処理
    EXEC SQL INSERT INTO TEMP (NAME,SAL)

```

```

VALUES (RTRIM(:S-TEMP),150)
END-EXEC.
IF SQLCODE NOT = 0
    PERFORM 800-SQLERROR
    PERFORM 800-USERLOG
    PERFORM 900-TPRETURNFAIL THRU 900-TPRETURNFAIL-X
END-IF.

STRING "Inserted data : " S-TEMP-ARR
    DELIMITED BY SIZE INTO LOGMSG-TEXT.
PERFORM 800-USERLOG.

MOVE "正常処理" TO SEND-STRING.

200-EXIT.

800-SQLERROR.
* LOGメッセージの内容を構成してロギング処理
    STRING "800-SQLERROR : " SQLERRMC
    DELIMITED BY SIZE INTO LOGMSG-TEXT.
    MOVE SQLCODE TO APPL-CODE.

*****
* Write out a log err messages
*****
800-USERLOG.
DISPLAY LOGMSG.

*****
* サービスが正常終了
*****
900-TPRETURNSCS.
    MOVE "サービスが正常終了" TO LOGMSG-TEXT.
    PERFORM 800-USERLOG.

SET TPSUCCESS TO TRUE.
MOVE LENGTH OF SEND-STRING TO LEN.
COPY TPRETURN REPLACING
DATA-REC BY SEND-STRING.

900-TPRETURNSCS-X.

*****
* サービスが異常終了
*****
900-TPRETURNFAIL.
    MOVE "サービスが異常終了" TO LOGMSG-TEXT.

```

```

        PERFORM 800-USERLOG.

        SET TPFAIL TO TRUE.
        MOVE "処理不可" TO SEND-STRING.
        MOVE LENGTH OF SEND-STRING TO LEN.
        COPY TPRETURN REPLACING

        DATA-REC BY SEND-STRING.
        900-TPRETURNFAIL-X.

```

以下は、各段階の説明です。

1. サービス名を設定します。サービスの呼び出し元がTPCALLで使用するサービス名はPROGRAM-IDで定義します。

```
PROGRAM-ID. CARRAYAUTO.
```

2. バッファを定義します。

クライアントとデータの送受信に使用されるバッファを定義します。サービスの呼び出し元とデータを送受信する際に使用するバッファを定義します。

```

*****
* User defined data records
*****
01 RECV-STRING          PIC X(100).
    01 SEND-STRING      PIC X(100).

```

3. サービスを開始します。サービスを開始しながら、サービスの呼び出し元が送信したデータを「バッファ定義」で定義したバッファに取得します。

例では、RECV-STRINGにサービスの呼び出し元が送信したデータが格納されます。エラーが発生すると「IF NOT TPOK」でチェックされ、サービスの呼び出し元が送信したデータよりバッファ(RECV-STRING)のサイズが小さい場合はTPTRUNCATEエラーが発生します。

```

MOVE LENGTH OF RECV-STRING TO LEN.
CALL "TPSVCSTART" USING TPSVCDEF-REC
                        TPTYPE-REC
                        RECV-STRING
                        TPSTATUS-REC.

IF NOT TPOK
    MOVE "TPSVCSTART Fail" TO LOGMSG-TEXT
    PERFORM 800-USERLOG
    PERFORM 900-TPRETURNFAIL THRU 900-TPRETURNFAIL-X
END-IF.

```

```

IF TPTRUNCATE
    MOVE "Received data was truncated" TO LOGMSG-TEXT
    PERFORM 800-USERLOG
    PERFORM 900-TPRETURNFAIL THRU 900-TPRETURNFAIL-X
END-IF.

```

4. サービスが正常終了します。

```

SET TPSUCCESS TO TRUE.
MOVE LENGTH OF SEND-STRING TO LEN.
COPY TPRETURN REPLACING
DATA-REC BY SEND-STRING.

```

5. ロジックのエラーで失敗を返します。

ビジネス・ロジックの終了後にサービスの呼び出し元に応答を渡すにはTPRETURNを使用します。サービスの実行が失敗した場合はTPFAILを使用します。TPFAILを使用してTPRETURNを行う場合は、サービスの呼び出し元はエラーコードとして、C言語ではtperrno=11(TPESVCFAIL)、COBOL言語ではTP-STATUS=11(TPESVCFAIL)が返されます。

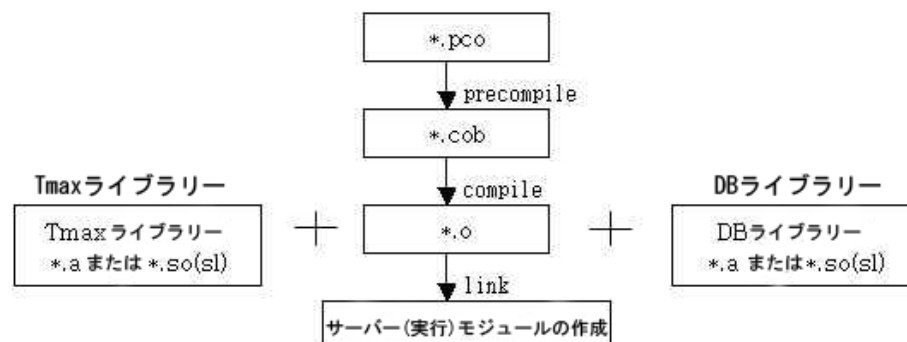
```

SET TPFALL TO TRUE.
MOVE "処理不可" TO SEND-STRING.
MOVE LENGTH OF SEND-STRING TO LEN.
COPY TPRETURN REPLACING
DATA-REC BY SEND-STRING.

```

3.1.2. Makefile

ユーザーが作成したサービス・プログラムとTmaxで提供するライブラリー、DBベンダーで提供するライブラリーが結合して1つの実行ファイルを作成します。以下は、実行ファイルを作成する手順です。



1. *.pco → *.cob

ProCOBOLプログラムをプリコンパイルします。

2. *.cob → *.o

COBOLプログラムをコンパイルしてオブジェクト・ファイルを作成します。

3. *.o + *.a → 実行ファイル

Tmaxで提供するライブラリー、DBベンダーで提供するライブラリーをリンクして実行ファイルを作成します。

COBOLで作成したサービスが1つであり、1つのサーバーが1つのサービスで構成される場合は、上記の3段階を1つのMakefileで処理します。ただし、一般的に1つのサーバーには複数のサービスが存在します。つまり、*.pco→.cob, *.cob → *.oの段階は、各サービス別に構成される必要があり、*.o+ライブラリー・ファイルをまとめてサーバーを作成するプロセスは1回のみ実行されます。

*.pco→ *.oはシェル・プログラムで処理し、実行ファイルの作成はMakefileで処理します。

```
$ cat carray_xa.sh
echo "-----> carrayauto.pco"
procobl8 sqlcheck=full userid=scott/tiger mode=ansi release_cursor=no
hold_cursor=yes maxliteral=160 ireclen=132 iname=carrayauto.pco

cob -C IBMCOMP -C NESTCALL -cx carrayauto.cob
echo "-----> carraynauto.pco"
procobl8 sqlcheck=full userid=scott/tiger mode=ansi release_cursor=no
hold_cursor=yes maxliteral=160 ireclen=132 iname=carraynauto.pco

cob -C IBMCOMP -C NESTCALL -cx carraynauto.cob
```

以下は、各段階の説明です。

1. COBOLのオブジェクト・ファイルをシェルで作成します。

```
sh carray_xa.sh
-----> carrayauto.pco
Pro*COBOL: Release 1.8.76.0.0 - Production on Wed May 7 23:43:42 2003
(c) Copyright 2001 Oracle Corporation. All rights reserved.
System default option values taken from:
/database/ora9/precomp/admin/pcccob.cfg
Precompiling carrayauto.pco
-----> carraynauto.pco
Pro*COBOL: Release 1.8.76.0.0 - Production on Wed May 7 23:43:42 2003
(c) Copyright 2001 Oracle Corporation. All rights reserved.
System default option values taken from:
/database/ora9/precomp/admin/pcccob.cfg
Precompiling carraynauto.pco
```

注

1. procob18またはcobのcompile optionはOS別に異なるため、確認してから作業します。Procob18で使用するoptionは\$ORACLE_HOME/precomp/demo/procob/demo_procob.mkファイルを参照します。
 2. carray_xa.shではcarrayauto.pco、carraynauto.pcoをプリコンパイル、コンパイルを経るため、サンプル・プログラムでcarrayauto.pco、carraynauto.pcoがすべて必要です。
-

2. 実行ファイルを作成するためのMakefileを作成します。

```
#Makefile

TARGET = carray_xa                -----(1)

TMAXCBLSTUB = cblstub             -----(2)
TMAXCBLSTUBOBJ = $(TMAXCBLSTUB).o

COBOBJS = carrayauto.o carraynauto.o -----(3)
OBS      = $(TMAXCBLSTUBOBJ) $(SDLOBJ) $(SVCTOBJ)
SVCTOBJ  = $(TARGET)_svctab.o
SDLOBJ   = $(TMAXDIR)/lib/sdl.o

CFLAGS   = -O -q32 -g -D_CBL_MODULE -I$(TMAXDIR) -I$(TMAXDIR)/usrinc
LDFLAGS  = -brtl
COBFLAGS = -C IBMCOMP -C NESTCALL

APPDIR   = $(TMAXDIR)/appbin
SVCTDIR  = $(TMAXDIR)/svct
LIBDIR   = $(TMAXDIR)/lib
LIBS     = -loras -lcbls -----(4)

ORALIBD  = $(ORACLE_HOME)/lib32
ORALIBS  = $(ORACLE_HOME)/precomp/lib32/cobsqlintf.o -lcblntsh -lld -lm
          `cat $(ORACLE_HOME)/lib32/sysliblist` -lm -lc_r -lpthreads
#
.SUFFIXES : .c
.c.o:
    $(CC) $(CFLAGS) -c $<
# server compile
#
$(TARGET): $(OBS)
    cob -Q "-brtl" $(COBFLAGS) -g -o $(TARGET) -L$(LIBDIR) -L$(ORALIBD)
          $(OBS) $(COBOBJS) $(LIBS) $(ORALIBS)
    mv -f $(TARGET) $(APPDIR)/.
    rm -f $(TARGET).o $(SVCTOBJ) $(TMAXCBLSTUBOBJ) $(COBOBJS)
$(TMAXCBLSTUBOBJ):
```

```

cc $(CFLAGS) $(LDFLAGS) -c $(TMAXDIR)/cobinc/$(TMAXCBLSTUB).c

$(SVCTOBJ):$(SVCTDIR)/$(TARGET)_svctab.c
cc $(CFLAGS) -c $(SVCTDIR)/$(TARGET)_svctab.c

#####

clean:
rm -f $(OBSJS) core $(TARGET) $(TARGET).lis $(TARGET).cob

```

項目	説明
(1) TARGET	サーバー(実行ファイル)名を設定します
(2) TMAXCBLSTUB	Tmaxで提供するstubファイル名を設定します
(3) COBOBJS	COBOLで作成したサービスのobjectファイル名を設定します
(4) LIBS	Tmaxで提供するライブラリーのファイル名を設定します

3.1.3. サービスのコンパイルおよび起動

Makefileの作成が完了すると、コンパイルします。

```
$ make -f carray_xa.mk
```

コンパイルされたサーバー・プログラムを起動します。

```
$ tmboot -s carray_xa
```

3.1.4. サービスの呼び出しテスト

tmdユーティリティを利用して、クライアント・プログラムを作成せずにサービスをテストします。

\$TMAXDIR/cobclientディレクトリーに含まれているtmd_CARRAYAUTOファイルを利用して、CARRAYAUTOサービスをテストします。

```
$ cd $TMAXDIR/cobclient
$ tmd -i t md_CARRAYAUTO
```

```
output: 正常処理
```

3.2. FDLバッファ使用プログラム

COBOLサービス・プログラムでFDLバッファを直接使用できないため、構造体の構造との相互変換作業が要求されます。そして、FDLバッファとビュー(フィールドと構造体をマッピングさせたファイル)を使用するには、事前作業が必要です。

cobolsample.tarの圧縮を解凍すると、ビューの例はucblincディレクトリーのviewdemo.vファイルで、FDLの例はfdldemo.fファイルです。

```
Fdl   : $TMAXDIR/ucblinc/fdldemo.f
View  : $TMAXDIR/ucblinc/viewdemo.v
```

3.2.1. FDLの作成

クライアントまたはサーバーで使用するFIELD情報を作成します。

< fdldemo.f >

# name	number	type	flags	comments
# string type				
FDL_S1	1001	string	-	-
FDL_S2	1002	string	-	-
# integer type				
FDL_I1	2001	int	-	-
FDL_I2	2002	int	-	-
# long type				
FDL_L1	3001	long	-	-
FDL_L2	3002	long	-	-
# float type				
FDL_F1	4001	float	-	-
FDL_F2	4002	float	-	-

outputで作成されたtmax.fdlファイルは、環境変数に指定した\$FDLFILEに指定されたディレクトリーに位置させます。

cobolsample.tarファイルを使用する場合は、\$TMAXDIR/ucblincディレクトリーにfdldemo.fファイルが作成されるため、このディレクトリーでfdlc作業を実行し、\$FDLFILEに\$TMAXDIR/ucblincが指定されるようにします。

```
$fdlc -c -i fdldemo.f -o tmax.fdl
```

参考

フィールド・バッファの使用に関する詳細は、『Tmax FDLリファレンスガイド』を参照してください。

3.2.2. ビュー・ファイルの作成

COBOLでは、フィールド・バッファを使用できないため、ビュー（構造体）を作成して、リクエストから渡された（または渡す）フィールド・バッファの内容を特定関数を使用して変換する際に使用します。

< viewdemo.v >

```
VIEW viewdemo
#type      cname      Fldkey      count      flag      size      null
string     sdata1     FDL_S1      1           -         20        "@@"
string     sdata2     FDL_S2      5           -         20        "@@"
int        idata1     FDL_I1      1           -         -         0
int        idata2     FDL_I2      5           -         -         0
long       ldata1     FDL_L1      1           -         -         0
long       ldata2     FDL_L2      5           -         -         0
float      fdata1     FDL_F1      1           -         -         0
float      fdata2     FDL_F2      5           -         -         0
double     ddata1     FDL_D1      1           -         -         0
double     ddata2     FDL_D2      5           -         -         0
END
```

outputで作成されたtmax.sdlファイルは、環境変数に指定した\$SDLFILEに定義したディレクトリーに位置させます。

cobolsample.tarファイルを使用する場合は、\$TMAXDIR/ucblincディレクトリーにviewdemo.vファイルが作成されるため、このディレクトリーでsdlc作業を実行して\$SDLFILEに\$TMAXDIR/ucblincが指定されるようにします。

```
$cd $TMAXDIR/ucblinc
$ sdlc -c -v viewdemo.v -o tmax.sdl
$ ls -l
-rw-r-xr-- 1 tmax dba 382 Apr 28 17:27 viewdemo.v*
-rw-r--r-- 1 tmax dba 976 May 7 18:17 tmax.sdl
```

3.2.3. COBOL用のビューファイルの作成

COBOL用のビューファイルを作成します。

```
$ sdlc -C -v viewdemo.v
$ls -l total 48
total 48
```

```
drwxr-xr-x 2 tmax dba 512 Apr 30 14:48 ./
drwxr-xr-x 7 tmax dba 512 Apr 14 11:50 ../
-rw-r--r-- 1 tmax dba 671 Apr 30 14:48 VIEWDEMO.cbl
-rw-r-xr-- 1 tmax dba 382 Apr 28 17:27 viewdemo.v*
```

< VIEWDEMO.cbl >

```
*          VIEWNAME: "viewdemo"
05 SDATA1                                PIC X(20).
05 SDATA2 OCCURS 5 TIMES                  PIC X(20).
05 IDATA1                                PIC S9(9)      USAGE IS COMP-5.
05 IDATA2 OCCURS 5 TIMES                  PIC S9(9) USAGE IS COMP-5.
05 LDATA1                                PIC S9(9) USAGE IS COMP-5.
05 LDATA2 OCCURS 5 TIMES                  PIC S9(9) USAGE IS COMP-5.
05 FDATA1                                USAGE IS COMP-1.
05 FDATA2 OCCURS 5 TIMES                  USAGE IS COMP-1.
05 DDATA1                                USAGE IS COMP-2.
05 DDATA2 OCCURS 5 TIMES                  USAGE IS COMP-2.
```

このファイルは、COBOLでCOPYコマンドを使用してソース内で使用する必要があるため、\$COBCPYに指定されたディレクトリーに位置させます。cobolsample.tarファイルを使用する場合は、\$TMAXIDIR/ucblincディレクトリーにこのファイルが位置されます。

注

効率的に管理するために、Tmaxで提供するCOBOL用のヘッダーファイルと分離し、他のディレクトリーに格納することを推奨します。

"export COBCPY=\$TMAXDIR/cobinc:\$TMAXDIR/ucblinc"で環境を設定し、ユーザーが作成したビューファイルは\$TMAXDIR/ucblincに格納します。

\$TMAXDIR/ucblincには以下のファイルが格納されます。

```
$ ls -l
-rw-r--r-- 1 tmax dba 671 May 7 18:17 VIEWDEMO.cbl
-rwxr-xr-- 1 tmax dba 381 Apr 30 15:10 fdldemo.f*
-rw-r--r-- 1 tmax dba 755 May 6 23:59 fdldemo_fdl.h
-rw-r--r-- 1 tmax dba 3852 May 6 23:59 tmax.fdl
-rw-r--r-- 1 tmax dba 976 May 7 20:13 tmax.sdl
-rw-r--r-- 1 tmax dba 996 May 7 00:56 viewdemo.sdl
-rw-r-xr-- 1 tmax dba 385 May 7 18:17 viewdemo.v*
-rw-r--r-- 1 tmax dba 574 May 7 20:13 viewdemo_sdl.h
-rw-r--r-- 1 tmax dba 571 Apr 30 14:53 viewdemo_sdl.h
```

3.2.4. サンプル・プログラム

以下は、フィールド・バッファーを使用してサービス・プログラムを作成する例です。

<\$TMAXDIR/sample/cobserver/fdlins.pco>

```
*****
* FDLバッファーを使用するサービス
*****

IDENTIFICATION          DIVISION.
    PROGRAM-ID.          FDLINS.
    AUTHOR.              TMAX DEVELOPMENT.
ENVIRONMENT             DIVISION.
    CONFIGURATION        SECTION.

DATA                    DIVISION.
    FILE                 SECTION.

WORKING-STORAGE         SECTION.

*****
* TMAX ATMI関数のためのdefinitions
*****

01 TPSVCRET-REC.
COPY TPSVCRET.
*
01 TPTYPE-REC.
COPY TPTYPE.
*
01 TPSTATUS-REC.
COPY TPSTATUS.
*
01 TPSVCDEF-REC.
COPY TPSVCDEF.
*
01 FML-REC.
COPY FMLINFO.

*****
* Log message definitions
*****

01 LOGMSG.
    05 LOGMSG-SVC PIC X(16).
    05 FILLER PIC X(3) VALUES " : ".
    05 LOGMSG-TEXT PIC X(50).
01 LOGMSG-LEN PIC S9(9) COMP-5.
```

```

*****
* User defined data records
*****

01 RECV-BUFFER.
    03 RECV-ALIGN          pic S9(9) COMP-5.
    03 RECV-DATA           pic x(1000).

01 SEND-BUFFER.
    03 SEND-ALIGN          pic S9(9) COMP-5.
    03 SEND-DATA           pic x(1000).

01 VIEWDEMO.
COPY VIEWDEMO.

LINKAGE                SECTION.
*
PROCEDURE                DIVISION.
*

000-MAIN.

    PERFORM 100-SVCSTART      THRU 100-EXIT.
    PERFORM 200-FDL2VIEW      THRU 200-EXIT.
    PERFORM 300-VIEW2FDL      THRU 300-EXIT.
    PERFORM 900-TPRETURNSCS   THRU 900-TPRETURNSCS-X.

000-EXIT.

*****
* クライアントから送信されたデータを受信
*****

100-SVCSTART.
    MOVE LENGTH OF RECV-BUFFER TO LEN.
    CALL "TPSVCSTART" USING TPSVCDEF-REC
                        TPTYPE-REC
                        RECV-BUFFER
                        TPSTATUS-REC.

    IF NOT TPOK
        MOVE "TPSVCSTART Fail" TO LOGMSG-TEXT
        PERFORM 800-USERLOG
        PERFORM 900-TPRETURNFAIL THRU 900-TPRETURNFAIL-X
    END-IF.

    IF TPTRUNCATE
        MOVE "Received data was truncated" TO LOGMSG-TEXT
        PERFORM 800-USERLOG
        PERFORM 900-TPRETURNFAIL THRU 900-TPRETURNFAIL-X

```



```

END-IF.

DISPLAY "-----".

MOVE SERVICE-NAME TO LOGMSG-SVC.
MOVE "サービスの開始" TO LOGMSG-TEXT.
PERFORM 800-USERLOG.
100-EXIT.

*****
* FIELDバッファの内容をstructに移動
*****
200-FDL2VIEW.
* viewname, FML-LENGTHを指定する必要があります。
MOVE "viewdemo" TO VIEWNAME.
MOVE LEN TO FML-LENGTH.

* FIELD bufferのデータをCOBOL structureに渡す
CALL "FVFTOS" USING RECV-BUFFER VIEWDEMO FML-REC.
IF NOT FOK
    MOVE "FVFTOS FAILED " TO LOGMSG-TEXT
    DISPLAY "FML-STATUS : " FML-STATUS
    PERFORM 800-USERLOG
    PERFORM 900-TPRETURNFAIL THRU 900-TPRETURNFAIL-X
END-IF.

200-EXIT.

*****
* structの内容をFIELDバッファに移動
*****
300-VIEW2FDL.
* FML-LENGTHを必ず使用する。
MOVE LENGTH OF SEND-BUFFER TO FML-LENGTH.
CALL "FINIT" USING SEND-BUFFER FML-REC.
IF NOT FOK
    MOVE "FINIT FAILED " TO LOGMSG-TEXT
    DISPLAY "FML-STATUS : " FML-STATUS
    PERFORM 800-USERLOG
    PERFORM 900-TPRETURNFAIL THRU 900-TPRETURNFAIL-X
END-IF.

MOVE "TEST STRING" TO SDATA1.
MOVE "TEST STRING1" TO SDATA2(1).
MOVE "TEST STRING2" TO SDATA2(2).
MOVE "TEST STRING3" TO SDATA2(3).

```

```

compute idata1 = idata1 + 10.
compute idata2(1) = idata2(1) + 10.
compute idata2(2) = idata2(2) + 10.
compute idata2(3) = idata2(3) + 10.
compute idata2(4) = idata2(4) + 10.

compute ldata1 = ldata1 + 10.
compute ldata2(1) = ldata2(1) + 10.
compute ldata2(2) = ldata2(2) + 10.
compute ldata2(3) = ldata2(3) + 10.
compute ldata2(4) = ldata2(4) + 10.

compute fdata1 = fdata1 + 10.
compute fdata2(1) = fdata2(1) + 10.
compute fdata2(2) = fdata2(2) + 10.
compute fdata2(3) = fdata2(3) + 10.
compute fdata2(4) = fdata2(4) + 10.

compute ddata1 = ddata1 + 10.
compute ddata2(1) = ddata2(1) + 10.
compute ddata2(2) = ddata2(2) + 10.
compute ddata2(3) = ddata2(3) + 10.
compute ddata2(4) = ddata2(4) + 10.

* COBOL structureのデータをFDL bufferに渡す

MOVE "viewdemo" TO VIEWNAME.
SET FUPDATE      TO TRUE.

CALL "FVSTOF" USING SEND-BUFFER VIEWDEMO FML-REC.
IF NOT FOK
    MOVE "FVSTOF FAILED " TO LOGMSG-TEXT
    DISPLAY "FML-STATUS : " FML-STATUS
    PERFORM 800-USERLOG
    PERFORM 900-TPRETURNFAIL THRU 900-TPRETURNFAIL-X
END-IF.
300-EXIT.

*****
* Write out a log err messages
*****
800-USERLOG.
DISPLAY LOGMSG.

*****
* サービスが正常終了

```

```

*****
900-TPRETURNSCS.
    MOVE "サービスが正常終了" TO LOGMSG-TEXT.
    PERFORM 800-USERLOG.
    SET TPSUCCESS TO TRUE.
    MOVE LENGTH OF SEND-BUFFER TO LEN.
    COPY TPRETURN REPLACING
        DATA-REC BY SEND-BUFFER.
900-TPRETURNSCS-X.

*****
* サービスが異常終了
*****

900-TPRETURNFAIL.
    MOVE "サービスが異常終了" TO LOGMSG-TEXT.
    PERFORM 800-USERLOG.

    SET TPFAIL TO TRUE.
    MOVE "処理不可" TO SDATA1.
    MOVE LENGTH OF SEND-BUFFER TO LEN.
    COPY TPRETURN REPLACING
        DATA-REC BY SEND-BUFFER.
900-TPRETURNFAIL-X.

```

以下は、各段階の説明です。

1. フィールドを宣言します。

RECV-BUFFER/SEND-BUFFERは、サービスの呼び出し元から送信されたデータを受信あるいは送信する際に使用するバッファで、VIEWDEMOはフィールド・バッファとビューを相互転換する際に使用する構造体です。

```

01 FML-REC.
COPY FMLINFO.

*****
* User defined data records
*****

01 RECV-BUFFER.
    03 RECV-ALIGN          pic S9(9) COMP-5.
    03 RECV-DATA           pic x(1000).

01 SEND-BUFFER.
    03 SEND-ALIGN          pic S9(9) COMP-5.
    03 SEND-DATA           pic x(1000).

01 VIEWDEMO.

```

```

COPY VIEWDEMO.

*      VIEWFILE: "viewdemo.v"
*      VIEWNAME: "viewdemo"
*
*      05 SDATA1                      PIC X(20).
*      05 SDATA2 OCCURS 5 TIMES      PIC X(20)
*      05 IDATA1                      PIC S9(9) USAGE IS COMP-5.
*      05 IDATA2 OCCURS 10 TIMES     PIC S9(9) USAGE IS COMP-5.
*      05 LDATA1                      PIC S9(9) USAGE IS COMP-5.
*      05 LDATA2 OCCURS 5 TIMES      PIC S9(9) USAGE IS COMP-5.
*      05 FDATA1                      USAGE IS COMP-1.
*      05 FDATA2 OCCURS 10 TIMES     USAGE IS COMP-1.
*      05 DDATA1                      USAGE IS COMP-2.
*      05 DDATA2 OCCURS 10 TIMES     USAGE IS COMP-2.

```

2. クライアントからフィールド・バッファを受信します。

サービスの呼び出し元から送信されたデータを受信するためには、サービスが削除される際、最初にTPSVCSTARTを実行する必要があります。

```

*****
* クライアントから送信されたデータを受信
*****

100-SVCSTART.
    MOVE LENGTH OF RECV-BUFFER TO LEN.
    CALL "TPSVCSTART" USING TPSVCDEF-REC
                                TPTYPE-REC
                                RECV-BUFFER
                                TPSTATUS-REC.

    IF NOT TPOK
        MOVE "TPSVCSTART Fail" TO LOGMSG-TEXT
        PERFORM 800-USERLOG
        PERFORM 900-TPRETURNFAIL THRU 900-TPRETURNFAIL-X
    END-IF.

    IF TPTRUNCATE
        MOVE "Received data was truncated" TO LOGMSG-TEXT
        PERFORM 800-USERLOG
        PERFORM 900-TPRETURNFAIL THRU 900-TPRETURNFAIL-X
    END-IF.

    DISPLAY "-----".

    MOVE SERVICE-NAME TO LOGMSG-SVC.
    MOVE "サービス開始" TO LOGMSG-TEXT.

```

```

        PERFORM 800-USERLOG.
100-EXIT.

```

3. FIELDバッファを構造体に変換します。

サービスの呼び出し元から受信したフィールド・バッファのデータをCOBOLで使用する構造体に変換する際に使用します。サービスの呼び出し元がフィールド・バッファに送信したデータがFVFTOSを使用してCOBOLの構造体に格納されます。arrayを使用する際、送信されたデータの数arrayの数に足りない場合は、ビュー・ファイルで定義したデフォルト値が設定されます。

```

*****
* FIELDバッファの内容を構造体へ移動
*****
        200-FDL2VIEW.
* viewname, FML-LENGTHを必ず指定します。
        MOVE "viewdemo" TO VIEWNAME.
        MOVE LEN TO FML-LENGTH.

* FIELD bufferのデータをCOBOL structureに渡す
        CALL "FVFTOS" USING RECV-BUFFER VIEWDEMO FML-REC.
        IF NOT FOK
            MOVE "FVFTOS FAILED " TO LOGMSG-TEXT
            DISPLAY "FML-STATUS : " FML-STATUS
            PERFORM 800-USERLOG
            PERFORM 900-TPRETURNFAIL THRU 900-TPRETURNFAIL-X
        END-IF.

200-EXIT.

```

4. COBOLソースでビジネス・ロジックを実行するデータを操作します。

```

        MOVE "TEST STRING" TO SDATA1.
        MOVE "TEST STRING1" TO SDATA2(1).
        MOVE "TEST STRING2" TO SDATA2(2).
        MOVE "TEST STRING3" TO SDATA2(3).
        .....
        compute ddata1 = ddata1 + 10.
        compute ddata2(1) = ddata2(1) + 10.
        compute ddata2(2) = ddata2(2) + 10.
        compute ddata2(3) = ddata2(3) + 10.
        compute ddata2(4) = ddata2(4) + 10.

```

5. 送信する際に使用するフィールド・バッファを作成します。

サービスの呼び出し元にデータを送信する際に使用するバッファは、受信したバッファをそのまま使用しても、新しいバッファを作成して使用しても構いません。新しいバッファを使用する場合は、FINITを使用します。

```

*****
* structの内容をFIELDバッファに移動
*****
300-VIEW2FDL.
* FML-LENGTHを必ず使用する。
    MOVE LENGTH OF SEND-BUFFER TO FML-LENGTH.
    CALL "FINIT" USING SEND-BUFFER FML-REC.
IF NOT FOK
    MOVE "FINIT FAILED " TO LOGMSG-TEXT
    DISPLAY "FML-STATUS : " FML-STATUS
    PERFORM 800-USERLOG
    PERFORM 900-TPRETURNFAIL THRU 900-TPRETURNFAIL-X
END-IF.

```

6. 構造体をフィールド・バッファに変換します。

サービスの呼び出し元にフィールド・タイプのデータを送信するには、COBOLで使用した構造体データをフィールド・タイプのデータに変換します。

```

MOVE "viewdemo" TO VIEWNAME.
    SET FUPDATE TO TRUE.

CALL "FVSTOF" USING SEND-BUFFER VIEWDEMO FML-REC.
    IF NOT FOK
        MOVE "FVSTOF FAILED " TO LOGMSG-TEXT
        DISPLAY "FML-STATUS : " FML-STATUS
        PERFORM 800-USERLOG
        PERFORM 900-TPRETURNFAIL THRU 900-TPRETURNFAIL-X
    END-IF.
300-EXIT.

```

7. クライアントにデータを送信します。

サービスが正常に実行された場合、TPRETURNを行う際にTUSUCCESS値を使用します。

```

*****
* サービスが正常終了
*****
900-TPRETURNSCS.
    MOVE "サービスが正常終了" TO LOGMSG-TEXT.
    PERFORM 800-USERLOG.

    SET TPSUCCESS TO TRUE.
    MOVE LENGTH OF SEND-BUFFER TO LEN.
    COPY TPRETURN REPLACING
        DATA-REC BY SEND-BUFFER.
900-TPRETURNSCS-X.

```

サービスが異常実行された場合、TPRETURNを行う際にTUFAIL値を使用します。

```
*****
* サービスが異常終了
*****

900-TPRETURNFAIL.
    MOVE "サービスが異常終了" TO LOGMSG-TEXT.
    PERFORM 800-USERLOG.

    SET TPFAIL TO TRUE.
    MOVE "処理不可 " TO SDATA1.
    MOVE LENGTH OF SEND-BUFFER TO LEN.
    COPY TPRETURN REPLACING
        DATA-REC BY SEND-BUFFER.
900-TPRETURNFAIL-X.
```

3.2.5. サービスの呼び出しテスト

tmdユーティリティを使用して、クライアント・プログラムを作成せずにサービスを呼び出します。

```
$ cd TMAXDIR/sample/cobclient
$ tmd -i tmd_FDLINS

fkey = 469763049, fname = FDL_S1, type = string, value = TEST STRING

fkey = 469763050, fname = FDL_S2, type = string, value = TEST STRING1

fkey = 469763050, fname = FDL_S2, type = string, value = TEST STRING2

fkey = 469763050, fname = FDL_S2, type = string, value = TEST STRING3


fkey = 201328593, fname = FDL_I1, type = int, value = 20
fkey = 201328594, fname = FDL_I2, type = int, value = 30
fkey = 201328594, fname = FDL_I2, type = int, value = 10
fkey = 201328594, fname = FDL_I2, type = int, value = 10
fkey = 201328594, fname = FDL_I2, type = int, value = 10
fkey = 268438457, fname = FDL_L1, type = long, value = 10010
fkey = 268438458, fname = FDL_L2, type = long, value = 100010
fkey = 268438458, fname = FDL_L2, type = long, value = 200010
fkey = 268438458, fname = FDL_L2, type = long, value = 300010
fkey = 268438458, fname = FDL_L2, type = long, value = 10
fkey = 335548321, fname = FDL_F1, type = float, value = 20.123451
fkey = 335548322, fname = FDL_F2, type = float, value = 30.123449
fkey = 335548322, fname = FDL_F2, type = float, value = 10.000000
fkey = 335548322, fname = FDL_F2, type = float, value = 10.000000
fkey = 335548322, fname = FDL_F2, type = float, value = 10.000000
```

```
fkey = 402658185, fname = FDL_D1, type = double, value = 100010.123450
fkey = 402658186, fname = FDL_D2, type = double, value = 200010.123450
fkey = 402658186, fname = FDL_D2, type = double, value = 10.000000
fkey = 402658186, fname = FDL_D2, type = double, value = 10.000000
fkey = 402658186, fname = FDL_D2, type = double, value = 10.000000
```

注

Tmax環境ファイルを読み込まないため、ユーザー・プロファイルに格納されたTMAX_HOST_PORT、TMAX_HOST_ADDR、FDLFILEの値を確認します。

3.3. グローバル・トランザクションのプログラム

3.3.1. サンプル・プログラム

以下は、グローバル・トランザクションを使用するサービス・プログラムの例です。

<\$TMAXDIR/sample/cobserver/carraynauto.pco>

```
*****
*   XAであり、CARRAYバッファを使用するサービス           *
*****
IDENTIFICATION          DIVISION.
PROGRAM-ID.             CARRAYNAUTO.

.....
WORKING-STORAGE         SECTION.
*****
*   Tmax ATMI関数のためのdefinitions
*****
01  TPSVCRET-REC.
COPY TPSVCRET.
*
01  TPTYPE-REC.
COPY TPTYPE.
*
01  TPSTATUS-REC.
COPY TPSTATUS.
*
01  TPSVCDEF-REC.
COPY TPSVCDEF.
*
01  TX-RETURN-STATUS.
COPY TXSTATUS.
.....
```



```

PROCEDURE                                DIVISION.
000-MAIN.

    PERFORM 100-SVCSTART                THRU 100-EXIT.
    PERFORM 200-DBINSERT                THRU 200-EXIT.
    PERFORM 900-TPRETURNSCS            THRU 900-TPRETURNSCS-X.

000-EXIT.

200-DBINSERT.
    MOVE RECV-STRING(1:LEN) TO S-TEMP-ARR.
    MOVE LENGTH OF S-TEMP-ARR TO S-TEMP-LEN.

    CALL "TXBEGIN" USING TX-RETURN-STATUS.
    IF NOT TPOK
        MOVE "FAIL TO TXBEGIN" TO LOGMSG-TEXT
        PERFORM 800-USERLOG
        PERFORM 900-TPRETURNFAIL THRU 900-TPRETURNFAIL-X
    END-IF.

* DB INSERT处理
EXEC SQL INSERT INTO TEMP (NAME,SAL)
        VALUES (RTRIM(:S-TEMP),150)
END-EXEC.
IF SQLCODE NOT = 0
    PERFORM 800-SQLError
    PERFORM 800-USERLOG
    CALL "TXROLLBACK" USING TX-RETURN-STATUS
    IF NOT TPOK
        MOVE "txrollback ERROR " TO LOGMSG-TEXT
        PERFORM 800-USERLOG
    END-IF
    PERFORM 900-TPRETURNFAIL THRU 900-TPRETURNFAIL-X
END-IF.

CALL "TXCOMMIT" USING TX-RETURN-STATUS.
IF NOT TPOK
    MOVE "TXCOMMIT ERROR" TO LOGMSG-TEXT
    PERFORM 800-USERLOG
    PERFORM 900-TPRETURNFAIL THRU 900-TPRETURNFAIL-X
END-IF.

STRING "Inserted data : " S-TEMP-ARR
        DELIMITED BY SIZE INTO LOGMSG-TEXT.
PERFORM 800-USERLOG.
MOVE "正常处理" TO SEND-STRING.
200-EXIT.

```

以下は、各段階の説明です。

1. グローバル・トランザクションを開始します。

```
CALL "TXBEGIN" USING TX-RETURN-STATUS.  
IF NOT TPOK  
    MOVE "FAIL TO TXBEGIN" TO LOGMSG-TEXT  
    PERFORM 800-USERLOG  
    PERFORM 900-TPRETURNFAIL THRU 900-TPRETURNFAIL-X  
END-IF.
```

2. グローバル・トランザクションの成功を処理します。

```
CALL "TXCOMMIT" USING TX-RETURN-STATUS.  
IF NOT TPOK  
    MOVE "TXCOMMIT ERROR" TO LOGMSG-TEXT  
    PERFORM 800-USERLOG  
    PERFORM 900-TPRETURNFAIL THRU 900-TPRETURNFAIL-X  
END-IF.
```

3. グローバル・トランザクションを取り消します。

```
CALL "TXROLLBACK" USING TX-RETURN-STATUS  
IF NOT TPOK  
    MOVE "txrollback ERROR " TO LOGMSG-TEXT  
    PERFORM 800-USERLOG  
END-IF.
```

明示的トランザクションを開始するサービスは、Tmax環境ファイルで**"AUTOTRAN=N"** (Tmax3.8.9以降、デフォルト)と設定されている必要があります。Tmax環境ファイルに"AUTOTRAN=Y"と設定されている場合、サービスを開始しながら自動的にグローバル・トランザクションを開始することになるため、サービスで明示的にTXBEGINを使用する際、TPEPROTOエラーが返されます。

注

"AUTOTRAN=N"に設定された場合も、クライアントで明示的トランザクションを開始し、サービスでも明示的トランザクションを開始すると、サービスで実行したTXBEGINはTPEPROTOエラーが返されるため、注意してください。

3.3.2. サービスのコンパイルおよび起動

サービスをコンパイルします。

```
$ sh carray_xa.sh
$ make -f carray_xa.mk
```

コンパイルされたサービスを起動します。

```
$ tmboot -s carray_xa
```

3.3.3. サービスの呼び出しテスト

サービスの呼び出しをテストします。

```
$ cd $TMAXDIR/sample/cobclient
$ tmd -i t md_CARRAYNAUTO
```

output: 正常処理

3.4. TPSVRINIT/TPSVRDONEの使用

3.4.1. TPSVRINIT

TPSVRINITはTmaxサーバーを初期化する関数です。Tmaxサーバーが初めて起動する際に1回のみ実行されます。一般的に、非XAサーバーグループに属するサーバーがDBと接続するロジックを実行します。tpsvrinit.pcoという名前でcobolsample.tarに含まれており、コンパイル過程(.oを作成)を経て、サーバー・プログラムのオブジェクトと一緒にリンクされます。

<\$TMAXDIR/sample/cobserver/tpsvrinit.pco>

```
*****
*   NXAサーバーでDB(Resource Manager)と接続を行う用途で使用
*****
IDENTIFICATION          DIVISION.
PROGRAM-ID.             TPSVRINIT.
AUTHOR.                 TMAX DEVELOPMENT.
ENVIRONMENT             DIVISION.
CONFIGURATION           SECTION.

DATA                    DIVISION.
FILE                   SECTION.

WORKING-STORAGE         SECTION.

EXEC SQL BEGIN DECLARE SECTION END-EXEC.
01  USERNAME            PIC X(10) VARYING.
```

```

01 PASSWD          PIC X(10) VARYING.
EXEC SQL END DECLARE SECTION END-EXEC.

EXEC SQL INCLUDE SQLCA END-EXEC.

LINKAGE            SECTION.
01 CMD-LINE.
    05 ARGC PIC 9(4) COMP-5.
    05 ARGV.
        10 ARGS PIC X OCCURS 0 TO 9999 DEPENDING ON ARGC.

01 SERVER-INIT-STATUS.
COPY TPSTATUS.
*
PROCEDURE          DIVISION USING CMD-LINE SERVER-INIT-STATUS.
*
000-TPSVRINIT-START.

*****
* DB error check
*****
EXEC SQL WHENEVER SQLERROR
    DO PERFORM SQL-ERROR
END-EXEC.

DISPLAY "ARGC : " ARGC, " ARGV: " ARGV.

*****
* DB connect
*****

MOVE "SCOTT" TO USERNAME-ARR.
MOVE 5 TO USERNAME-LEN.

MOVE "TIGER" TO PASSWD-ARR.
MOVE 5 TO PASSWD-LEN.
EXEC SQL
    CONNECT :USERNAME IDENTIFIED BY :PASSWD
END-EXEC.

DISPLAY "CONNECTED TO ORACLE as User : " USERNAME-ARR.

SET TPOK IN SERVER-INIT-STATUS TO TRUE.
EXIT PROGRAM.

SQL-ERROR.
EXEC SQL WHENEVER SQLERROR CONTINUE END-EXEC.

```

```

        DISPLAY " ".
        DISPLAY "ORACLE ERROR DETECTED:".
        DISPLAY SQLERRMC.
        EXEC SQL ROLLBACK WORK RELEASE END-EXEC.

        SET TPEINVAL IN SERVER-INIT-STATUS TO TRUE.
EXIT PROGRAM.

```

3.4.2. TPSVRDONE

TPSVRDONEはTmaxサーバーを終了する関数です。Tmaxサーバーが終了する際に最後の1回のみ実行されます。一般的に、非XAサーバーグループに属するサーバーがDB接続を解除するロジックを実行します。tpsvrdone.pcoという名前でもcobolsample.tarに含まれており、コンパイル過程(.oを作成)を経て、サーバープログラムのオブジェクトと一緒にリンクされます。

<\$TMAXDIR/sample/cobserver/tpsvrdone.pco>

```

*****
* NXAサーバーでDB(Resource Manager)との接続を解除する用途で使用
*****
IDENTIFICATION          DIVISION.
PROGRAM-ID.             TPSVRDONE.
AUTHOR.                 TMAX DEVELOPMENT.
ENVIRONMENT             DIVISION.
CONFIGURATION           SECTION.
DATA                   DIVISION.
FILE                   SECTION.
WORKING-STORAGE        SECTION.
01 SERVER-DONE-STATUS.
COPY TPSTATUS.

        EXEC SQL INCLUDE SQLCA END-EXEC.
*
        PROCEDURE          DIVISION .
*
000-TPSVRDONE-START.

*****
* DB error check
*****
        EXEC SQL WHENEVER SQLERROR
                DO PERFORM SQL-ERROR END-EXEC.

*****
* DB disconnect
*****

```

```

EXEC SQL COMMIT WORK RELEASE END-EXEC.
DISPLAY " ".
DISPLAY "DISCONNECTED TO ORACLE ".

SET TPOK IN SERVER-DONE-STATUS TO TRUE.
EXIT PROGRAM.

SQL-ERROR.
EXEC SQL WHENEVER SQLERROR CONTINUE END-EXEC.
DISPLAY " ".
DISPLAY "ORACLE ERROR DETECTED :".
DISPLAY " ".
DISPLAY SQLERRMC.
EXEC SQL ROLLBACK WORK RELEASE END-EXEC.

SET TPEINVAL IN SERVER-DONE-STATUS TO TRUE.
EXIT PROGRAM.

```

3.4.3. サービスのコンパイルおよび起動

サービスをコンパイルします。

```

$ sh carray_nxa.sh
$ make -f carray_nxa.mk

```

コンパイル後、サーバーのcarray_nxaを起動します。

```

$ tmboot -s carray_nxa

```

userlogファイルの\$TMAXDIR/log/ulog/carray_nxa.outでサーバーの動作を確認できます。

```

ARGC : 00023 ARGV: carray_nxa scott tiger CONNECTED TO ORACLE as User : SCOTT

```

3.4.4. サービスの呼び出しテスト

サービスの呼び出しをテストします。

```

$ cd $TMAXDIR/sample/cobclient
$ tmd -i tmd_CARRAYNXA

```

```

output: 正常処理 from CARRAYNXA

```

3.5. サービスでのサービス呼び出しプログラム

3.5.1. サンプル・プログラム

以下は、サービスでサービスの呼び出しを行うプログラムについての例です。

< \$TMAXDIR/sample/cobserver/svctpcall.pco >

```
*****
* DBを使用せずにCARRAYバッファーを使用するサービス      *
*****

IDENTIFICATION          DIVISION.
PROGRAM-ID.             SVCTPCALL.
AUTHOR.                 TMAX DEVELOPMENT.
ENVIRONMENT             DIVISION.
CONFIGURATION           SECTION.

DATA                    DIVISION.
FILE                   SECTION.

WORKING-STORAGE        SECTION.
*****
* TMAX ATMI関数のためのdefinitions
*****

01  TPSVCRET-REC.
COPY TPSVCRET.
*

01  TPTYPE-REC.
COPY TPTYPE.
*

01  TPSTATUS-REC.
COPY TPSTATUS.
*

01  TPSVCDEF-REC.
COPY TPSVCDEF.

*****
* Log message definitions
*****

01  LOGMSG.
      05  LOGMSG-SVC  PIC X(16).
      05  FILLER      PIC X(3) VALUES " : ".
      05  LOGMSG-TEXT PIC X(50).
01  LOGMSG-LEN        PIC S9(9)  COMP-5.
```

```

*****
* User defined data records
*****

01 RECV-STRING          PIC X(100).
01 SEND-STRING          PIC X(100).

77 NUM-TPSTATUS         PIC 99.

PROCEDURE                DIVISION.
*
000-MAIN.

        PERFORM 100-SVCSTART      THRU 100-EXIT.
        PERFORM 200-TPCALL        THRU 200-EXIT.
        PERFORM 900-TPRETURNSCS   THRU 900-TPRETURNSCS-X.

*****
* クライアントから送信されたデータを受信
*****

100-SVCSTART.
.....
100-EXIT.

*****
* tpcall in service
*****

200-TPCALL.

        MOVE "CARRYNXA" TO SERVICE-NAME.
        MOVE LENGTH OF SEND-STRING TO LEN IN TPTYPE-REC.

        CALL "TPCALL" USING TPSVCDEF-REC
                                TPTYPE-REC
                                RECV-STRING
                                TPTYPE-REC
                                SEND-STRING
                                TPSTATUS-REC .

        IF NOT TPOK
            MOVE "Service failed" TO LOGMSG-TEXT
            MOVE TP-STATUS TO NUM-TPSTATUS
            STRING "TPCALL call failed : " NUM-TPSTATUS
                DELIMITED BY SIZE INTO LOGMSG-TEXT
            PERFORM 800-USERLOG
            PERFORM 900-TPRETURNFAIL THRU 900-TPRETURNFAIL-X
        END-IF.

200-EXIT.

```



```

*****
* Write out a log err messages
*****

    800-USERLOG.
        DISPLAY LOGMSG.

*****

* サービスが正常終了
*****

    900-TPRETURNSCS.
        .....
    900-TPRETURNSCS-X.

*****

* サービスが異常終了
*****

    900-TPRETURNFAIL.
        .....
    900-TPRETURNFAIL-X.

```

サービスの呼び出し元は、Tmaxクライアントまたはサービスです。Tmaxサービスでも他のTmaxサービスを呼び出すことができるという意味です。

上記例のように、他のサービスを呼び出すためにはTPCALLで送受信するバッファを使用します。

```

MOVE "CARRAYNXA" TO SERVICE-NAME.
MOVE LENGTH OF SEND-STRING TO LEN IN TPTYPE-REC.

CALL "TPCALL" USING TPSVCDEF-REC
                    TPTYPE-REC
                    RECV-STRING
                    TPTYPE-REC
                    SEND-STRING
                    TPSTATUS-REC

```

3.5.2. サービスのコンパイルおよび起動

サービスをコンパイルします。

```
$ make -f svctpcall.mk
```

コンパイルされたサービスを起動します。

```
$ tmboot -s svctpcall
```

3.5.3. サービスの呼び出しテスト

サービスの呼び出しをテストします。

```
$ cd $TMAXDIR/sample/cobclient  
$ tmd -i tmd_SVCTPCALL
```

```
output: 正常処理 from CARRAYNXA
```

索引

F

FDLバッファ使用プログラム例, 60

M

Makefile, 56

T

TPSVRDONEの使用例, 77

TPSVRINITの使用例, 75

TPTYPE, 31

か

環境設定

COBOLライブラリー・ファイル, 2

環境ファイルの設定, 2

環境変数の設定, 2

関数

FINIT, 17

FVFTOS, 18

FVSTOF, 19

TP_SLEEP, 21

TPACALL, 22

TPCALL, 24

TPCANCEL, 28

TPFORWARD, 29

TPGETLEV, 31

TPGETRPLY, 31

TPRETURN, 34

TPSVCSTART, 36

TPSVRDONE, 37

TPSVRINIT, 38

TXBEGIN, 39

TXCOMMIT, 40

TXINFORM, 42

TXROLLBACK, 43

TXSETCOMMITRET, 45

TXSETTIMEOUT, 47

TXSETTRANCTL, 48

グローバル・トランザクションのプログラム例, 72

さ

サービスの呼び出しプログラムの例, 79

サービス・プログラムの例, 51

た

ディレクトリー構造, 1

データ格納cbl

FDL-INFO, 12

TPAUTDEF, 11

TPBCTDEF, 12

TPCMTDEF, 11

TPEVTDEF, 13

TPINFDEF, 10

TPPRIDEF, 11

TPSTATUS, 7

TPSVCDEF, 9

TPSVCRET, 10

TPTRXDEF, 11

TPTRXLEV, 12

TPTYPE, 8

TXINFDEF, 15

TXSTATUS, 14

