

OpenTP1 Version 7
分散トランザクション処理機能

OpenTP1 クライアント使用の手引 TP1/Client/W,
TP1/Client/P 編

解説・手引・文法・操作書

3000-3-D58-50

前書き

■ 対象製品

- ・適用 OS : AIX V6.1, AIX V7.1, AIX V7.2
P-1M64-2541 uCosminexus TP1/Client/W 07-51
P-1M64-1521 uCosminexus TP1/Client/W(64) 07-51
- ・適用 OS : Red Hat Enterprise Linux Server 6 (32-bit x86)
P-8164-2511 uCosminexus TP1/Client/W 07-52
- ・適用 OS : Red Hat Enterprise Linux Server 6 (64-bit x86_64)
P-8164-2511 uCosminexus TP1/Client/W 07-52
P-8264-2511 uCosminexus TP1/Client/W(64) 07-52
- ・適用 OS : Red Hat Enterprise Linux Server 7 (64-bit x86_64)
P-8164-2511 uCosminexus TP1/Client/W 07-52
P-8264-2511 uCosminexus TP1/Client/W(64) 07-52
- ・適用 OS : Red Hat Enterprise Linux Server 8 (64-bit x86_64)
P-8164-2511 uCosminexus TP1/Client/W 07-52
P-8264-2511 uCosminexus TP1/Client/W(64) 07-52
- ・適用 OS : Windows 7, Windows 7 x64 Edition, Windows 8, Windows 8 x64 Edition, Windows 8.1, Windows 8.1 x64 Edition, Windows 10, Windows 10 x64 Edition, Windows 11, Windows Server 2008 R2, Windows Server 2012, Windows Server 2012 R2, Windows Server 2016, Windows Server 2019, Windows Server 2022
P-2464-2154 uCosminexus TP1/Client/P 07-51
- ・適用 OS : Windows 7 x64 Edition, Windows 8 x64 Edition, Windows 8.1 x64 Edition, Windows 10 x64 Edition, Windows 11, Windows Server 2008 R2, Windows Server 2012, Windows Server 2012 R2, Windows Server 2016, Windows Server 2019, Windows Server 2022
P-2964-2124 uCosminexus TP1/Client/P(64) 07-51
- ・適用 OS : Windows 10, Windows 10 x64 Edition, Windows 11, Windows Server 2016, Windows Server 2019, Windows Server 2022
P-2464-2164 uCosminexus TP1/Client/P 07-60

・適用 OS : Windows 10 x64 Edition, Windows 11, Windows Server 2016, Windows Server 2019, Windows Server 2022

P-2964-2134 uCosminexus TP1/Client/P(64) 07-60

これらのプログラムプロダクトのほかにも、このマニュアルをご利用になれる場合があります。詳細は「リリースノート」でご確認ください。

■ 輸出時の注意

本製品を輸出される場合には、外国為替及び外国貿易法の規制並びに米国輸出管理規則など外国の輸出関連法規をご確認の上、必要な手続きをお取りください。

なお、不明な場合は、弊社担当営業にお問い合わせください。

■ 商標類

HITACHI, CommuniNet, OpenTP1, uCosminexus, XDM は、株式会社 日立製作所の商標または登録商標です。

AIX は、世界の多くの国で登録された International Business Machines Corporation の商標です。

Azure は、マイクロソフト 企業グループの商標です。

IBM は、世界の多くの国で登録された International Business Machines Corporation の商標です。

Itanium は、Intel Corporation またはその子会社の商標です。

Linux は、Linus Torvalds 氏の日本およびその他の国における登録商標または商標です。

Microsoft は、マイクロソフト 企業グループの商標です。

Oracle および Java は、オラクルおよびその関連会社の登録商標です。

Red Hat is a registered trademark of Red Hat, Inc. in the United States and other countries.

Red Hat は、米国およびその他の国における Red Hat, Inc.の登録商標です。

Red Hat Enterprise Linux is a registered trademark of Red Hat, Inc. in the United States and other countries.

Red Hat Enterprise Linux は、米国およびその他の国における Red Hat, Inc.の登録商標です。

UNIX は、The Open Group の商標です。

Visual C++は、マイクロソフト 企業グループの商標です。

Visual Studio は、マイクロソフト 企業グループの商標です。

Windows は、マイクロソフト 企業グループの商標です。

Windows Server は、マイクロソフト 企業グループの商標です。

その他記載の会社名、製品名などは、それぞれの会社の商標もしくは登録商標です。

本書には、X/Open の許諾に基づき X/Open CAE Specification System Interfaces and Headers, Issue4, (C202 ISBN 1-872630-47-2) Copyright (C) July 1992, X/Open Company Limited の内容が含まれています；

なお、その一部は IEEE Std 1003.1-1990, (C) 1990 Institute of Electrical and Electronics Engineers, Inc.及び IEEE std 1003.2/D12, (C) 1992 Institute of Electrical and Electronics Engineers, Inc.を基にしています。

事前に著作権所有者の許諾を得ずに、本書の該当部分を複製、複写及び転記することは禁じられています。

本書には、X/Open の許諾に基づき X/Open Preliminary Specification Distributed Transaction Processing : The TxRPC Specification (P305 ISBN 1-85912-000-8) Copyright (C) July 1993, X/Open Company Limited の内容が含まれています；

事前に著作権所有者の許諾を得ずに、本書の該当部分を複製、複写及び転記することは禁じられています。

本書には、Open Software Foundation, Inc.が著作権を有する内容が含まれています。

This document and the software described herein are furnished under a license, and may be used and copied only in accordance with the terms of such license and with the inclusion of the above copyright notice. Title to and ownership of the document and software remain with OSF or its licensors.

■ 発行

2022年12月 3000-3-D58-50

■ 著作権

All Rights Reserved. Copyright (C) 2006, 2022, Hitachi, Ltd.

変更内容

変更内容 (3000-3-D58-50) uCosminexus TP1/Client/P 07-60, uCosminexus TP1/Client/P(64) 07-60

追加・変更内容	変更箇所
32bit 版で文字コード変換機能を使用する場合に必要な設定を追加した。	5.2.2(2)
TP1/Client/P の環境設定について説明する章を追加して、Windows の C ランタイムライブラリのインストールに関する説明を追加した。	7.

単なる誤字・脱字などはお断りなく訂正しました。

変更内容 (3000-3-D58-42) uCosminexus TP1/Client/W 07-52, uCosminexus TP1/Client/W 07-51, uCosminexus TP1/Client/P 07-51

追加・変更内容
マニュアル訂正の内容を反映した。

変更内容 (3000-3-D58-41) uCosminexus TP1/Client/W 07-52, uCosminexus TP1/Client/W 07-51, uCosminexus TP1/Client/P 07-51

追加・変更内容
認証 RPC の説明を削除した。

変更内容 (3000-3-D58-40) uCosminexus TP1/Client/W 07-52, uCosminexus TP1/Client/W 07-51, uCosminexus TP1/Client/P 07-51

追加・変更内容
TP1/Server Base のバージョンに関する記述を削除した。
ユーザ認証機能に関する説明を変更した。
常設コネクションに関する説明を追加した。
「一時的に格納したサービス情報の有効期間」を「一時的に格納したサービス情報の有効時間」に変更した。
ノード間負荷バランス機能で、サーバ側の判断で負荷分散を行う場合の説明を変更した。
ネームサービスを使用した RPC の説明を変更した。
窓口となる TP1/Server の説明を追加した。
dc_rpc_call_to_s 関数の説明を追加した。
トランザクション制御の説明を変更した。
トランザクションの処理時間に次の時間を追加した。

追加・変更内容
<ul style="list-style-type: none"> ・トランザクション同期点処理時の最大通信待ち時間 ・トランザクションブランチ最大実行可能時間
メッセージの組み立て機能と送達確認機能に関する説明を変更した。
TCP/IP 通信機能のユースケースの説明を変更した。
マルチスレッド機能に関する注意事項を追加した。
「CUP 実行ディレクトリ」を「カレントディレクトリ (CUP 実行ディレクトリ)」に変更した。
トラブルシュート機能に関する注意事項を追加した。
受信ポート固定機能を使用する場合の説明を変更した。
DLL と通常オブジェクトライブラリに関する記載を変更した。
翻訳と結合に関する説明を変更した。
クライアント環境定義 DCCLTONLYTHISNODE の指定値に関する説明を変更した。
X_OCTET に関する説明を削除した。
TP1/Client/P(64)の場合の形式を追加した。
次のオペランドの指定値を小文字から大文字に変更した。 <ul style="list-style-type: none"> ・dcclttxpsp ・dccltuttrcmt ・dcsockopenatrcv ・dccltdeliverycheck ・dccltloadbalance ・dcscddirect ・dccltdatacomp ・dccltauthent
dccltbacklogcount を追加した。
次のオペランドの説明を変更した。 <ul style="list-style-type: none"> ・DCCLTTRCPUTM ・DCCACHE ・DCCLTSERVICEGROUPLIST ・DCCLTDATACOMP ・DCEXTENDFUNCTION ・DCCLTINQUIRETIME ・DCHOSTCHANGE
-f または /f オプションの説明を変更した。
TP1/Client/P(64)および TP1/Client/W(64)の場合の出力形式を追加した。
記述形式の説明を変更した。

追加・変更内容
KFCA02468-I の記述を変更した。
UNIX の CUP 実行プロセスでユーザが同時に使用できるファイル記述子の最大数の説明を追加した。

変更内容 (3000-3-D58-30) uCosminexus TP1/Client/W 07-50, uCosminexus TP1/Client/P 07-50

追加・変更内容
マルチスケジューラ機能を使用した RPC の説明を変更した。
窓口となる TP1/Server の切り替え機能の説明を変更した。
窓口となる TP1/Server の負荷分散の説明を変更した。
TCP/IP 通信機能の説明を追加した。
TCP/IP 通信機能のユースケースごとの設定方法を説明した項を追加した。
トラブルシュート機能を使用するときの注意事項を追加した。
UNIX 環境の場合の翻訳と結合に AIX と Linux についての説明を追加した。
コーディング例を修正した。
次に示す関数の説明を修正した。 <ul style="list-style-type: none"> • dc_rpc_call_to_s • DCRPC_DIRECT_SCHEDULE • dc_clt_cancel_notification_s
COBOL/2, COBOL85 に関する記述を削除した。
COBOL2002 の記述を追加した。
常設コネクション機能に次を追加した。 <ul style="list-style-type: none"> • 端末識別情報の設定
COBOL85 用の記述を COBOL2002 用の記述にした。
次の要求文のクライアント ID を設定するデータ型を COMP から COMP-X に変更した。 <ul style="list-style-type: none"> • CBLDCCLS('CLTIN ') • CBLDCCLS('EXCLTIN ') • CBLDCCLS('CLTOUT ') • CBLDCRPS('OPEN ') • CBLDCRPS('CLOSE ') • CBLDCRPS('CALL ') • CBLDCRPS('SETWATCH') • CBLDCRPS('GETWATCH') • CBLDCCLS('CONNECT ') • CBLDCCLS('DISCNCT ')

追加・変更内容

- CBLDCCLS('STRAPHST')
- CBLDCCLS('GTRAPHST')
- CBLDCCLS('STCONINF')
- CBLDCTRS('BEGIN ')
- CBLDCTRS('C-COMMIT')
- CBLDCTRS('C-ROLL ')
- CBLDCTRS('U-COMMIT')
- CBLDCTRS('U-ROLL ')
- CBLDCTRS('INFO ')
- CBLDCCLS('GETTRNID')
- CBLDCCLS('SEND ')
- CBLDCCLS('EXSEND ')
- CBLDCCLS('RECEIVE ')
- CBLDCCLS('RECEIVE2')
- CBLDCCLS('ASMSEND ')
- CBLDCCLS('ASMRECV ')
- CBLDCCLS('O-NOTIFY')
- CBLDCCLS('C-NOTIFY')
- CBLDCCLS('A-NOTIFY')
- CBLDCCLS('EXNCACPT')

次の要求文の説明を修正した。

- CBLDCRPS('CALL ')
- CBLDCCLS('GTRAPHST')
- CBLDCTRS('INFO ')
- CBLDCCLS('GETTRNID')
- CBLDCCLS('CANCEL ')
- CBLDCCLS('EXNCANCL')
- CBLDCCLS('A-NOTIFY')

export に指定するオペランドを変更した。

クライアント環境定義の設定方法の説明を追加した。

次のオペランドの説明を追加した。

- DCRCVPORT
- DCSOCKOPENATRCV

次のオペランドの説明を修正した。

- DCCLTSERVICEGROUPLIST
- DCSCDPORT
- DCSYSWATCHTIM
- DCSCDMULTICOUNT

追加・変更内容
<ul style="list-style-type: none"> • DCCLTOPTION • DCCLTBACKLOGCOUNT • DCUTOKEY • DCWATCHTIMRPCINHERIT
クライアント環境定義に関する注意事項を追加した。
cltdump（トレースの編集出力）出力形式の内容の記述を修正した。
次のメッセージの記述を修正した。 <ul style="list-style-type: none"> • KFCA02406-E • KFCA02450-W • KFCA02451-W

変更内容 (3000-3-D58-20) uCosminexus TP1/Client/W 07-02, uCosminexus TP1/Client/P 07-02

追加・変更内容
<p>メッセージの組み立て機能、およびメッセージの送達確認機能を追加した。</p> <p>これに伴い、次の関数を追加した。</p> <ul style="list-style-type: none"> • dc_clt_assem_send_s • dc_clt_assem_receive_s • CBLDCCLS(`ASMSEND`) • CBLDCCLS(`ASMRECV`) <p>クライアント環境定義に次のオペランドを追加した。</p> <ul style="list-style-type: none"> • DCCLTDELIVERYCHECK <p>次のメッセージの説明を変更した。</p> <ul style="list-style-type: none"> • KFCA02447-E <p>次のメッセージを追加した。</p> <ul style="list-style-type: none"> • KFCA02485-E • KFCA02486-E
TP1/Server の性能検証用トレースに、TP1/Client が設定した性能検証用の識別情報を出力できるようにした。これに伴い、クライアント環境定義に DCCLTPRFINFOSEND オペランドを追加した。
TP1/Client の CUP の送信元ホストを指定できる機能を追加した。これに伴い、クライアント環境定義に DCCLTCUPSNDHOST オペランドを追加した。
受信ポート固定機能について、機能説明を追加した。
関数および要求文を使用するときの注意事項について、説明を追加した。
次に示す関数の引数 defpath にパス名を指定した場合の、ファイルの読み込み順序についての説明を変更した。 <ul style="list-style-type: none"> • dc_clt_cltin_s 関数 • dc_clt_accept_notification_s 関数

追加・変更内容
<ul style="list-style-type: none"> dc_clt_cancel_notification_s 関数 dc_clt_open_notification_s 関数
<p>次に示す要求文のデータ名にパス名を指定した場合の、ファイルの読み込み順序についての説明を変更した。</p> <ul style="list-style-type: none"> CBLDCCLS('CLTIN') CBLDCCLS('EXCLTIN') CBLDCCLS('NOTIFY') CBLDCCLS('EXNACPT') CBLDCCLS('CANCEL') CBLDCCLS('EXNCANCL') CBLDCCLS('O-NOTIFY')
<p>リンケージオプションに指定するライブラリ名の説明を変更した。</p>
<p>CBLDCRPS('GETWATCH')の値が返されるデータ領域の説明に、データ名 C を追加した。</p>
<p>端末識別情報を設定するための要求文として、CBLDCCLS(STCONINF')を追加した。</p>
<p>次の関数の注意事項の説明を変更した。</p> <ul style="list-style-type: none"> dc_clt_send_s 関数 CBLDCCLS('SEND') CBLDCCLS('EXSEND')
<p>CBLDCCLS('RECEIVE2')のデータ名 E の説明に、1 から 65535 の値を設定した場合の説明を追加した。</p>
<p>文字コード変換機能で提供する要求文は、マルチスレッド環境でも正しく動作する旨を追記した。</p>
<p>クライアント環境定義の一覧を追加した。</p>
<p>クライアント環境定義 DCSCDMULTICOUNT の指定範囲を((1~4096))に、デフォルト値を《1》に変更した。</p>
<p>次に示すバージョンの変更点を記載した。</p> <ul style="list-style-type: none"> TP1/Client/W 07-02 TP1/Client/P 07-02

変更内容 (3000-3-D58-10) uCosminexus TP1/Client/W 07-01, uCosminexus TP1/Client/P 07-01

追加・変更内容
<p>MHP からのメッセージを受信するために、CUP から実行する関数に dc_clt_receive2_s 関数を追加した。</p>
<p>マルチスレッドに対応しない関数の実行についての説明を追記した。</p>
<p>マルチスレッド環境では、XATMI インタフェース関数を使用できないようにした。</p>
<p>dc_clt_cltin_s 関数の次に示す引数の指定方法を変更した。</p> <ul style="list-style-type: none"> logname passwd

追加・変更内容

次に示す関数は、関数の呼び出しごとにクライアント環境定義を定義できる旨を追記した。

- dc_clt_cltin_s 関数
- dc_clt_accept_notification_s 関数
- dc_clt_cancel_notification_s 関数
- dc_clt_open_notification_s 関数

TP1/Client/W の COBOL 言語用テンプレートの格納ディレクトリを変更した。

TP1/Server Base との間でのデータ送信遅延を回避できるように、クライアント環境定義に次のオペランドを追加した。

- DCCLTRECVBUFFSIZE
- DCCLTSEENDBUFFSIZE
- DCCLTTCPNODELAY

DCCLTSERVICEGROUPLIST オペランドの説明に、dc_rpc_call_s 関数で呼び出したサービスグループ名が DCCLTSERVICEGROUPLIST オペランドに指定したファイルに定義されていない場合、クライアント環境定義 DCCLTNOSERVER の指定によって動作が異なる旨を追記した。

次に示すオペランドの説明に、接続先が rap サーバの場合は rap リスナーサービス定義のオペランドの指定に従う旨を追記した。

- DCCLTTREXPTM
- DCCLTTREXPSP
- DCCLTTRCPUTM
- DCCLTINQUIRETIME
- DCCLTTRSTATISITEM
- DCCLTTROPTIITEM
- DCCLTTRWATCHTIME
- DCCLTTRRBINFO
- DCCLTTRLIMITTIME
- DCCLTTRRBRCV
- DCCLTTRRECOVERYTYPE

次に示す関数は、引数 defpath に指定されたファイルを参照して定義を読み込む旨を追記した。

- dc_clt_cltin_s 関数
- dc_clt_accept_notification_s 関数
- dc_clt_cancel_notification_s 関数
- dc_clt_open_notification_s 関数

メッセージを追加した。

バージョンアップ時の、関数、定義およびメッセージの変更を記載した。

はじめに

このマニュアルは、次に示すプログラムプロダクトの機能と使い方について説明したものです。本文中に記載されている製品のうち、このマニュアルの対象製品ではない製品については、OpenTP1 Version 7 対応製品の発行時期をご確認ください。

- P-1M64-2541 uCosminexus TP1/Client/W
- P-1M64-1521 uCosminexus TP1/Client/W(64)
- P-8164-2511 uCosminexus TP1/Client/W
- P-8264-2511 uCosminexus TP1/Client/W(64)
- P-2464-2154 uCosminexus TP1/Client/P
- P-2964-2124 uCosminexus TP1/Client/P(64)
- P-2464-2164 uCosminexus TP1/Client/P
- P-2964-2134 uCosminexus TP1/Client/P(64)

■ 対象読者

システム管理者をはじめとして、システム設計者、プログラマ、オペレータの方を対象としています。

このマニュアルの記述は、マニュアル「OpenTP1 解説」の知識があることを前提としていますので、あらかじめお読みいただくことをお勧めします。

■ マニュアルの構成

このマニュアルは、次に示す章と付録から構成されています。

第1章 概要

OpenTP1 のクライアント機能の特長について説明しています。

第2章 機能

OpenTP1 のクライアント機能について説明しています。

第3章 ユーザアプリケーションプログラムの作成 (C 言語編)

C 言語でのユーザアプリケーションプログラムインタフェースについて説明しています。

第4章 TP1/Client で使用できる関数 (C 言語編)

TP1/Client で使用できる関数について説明しています。

第 5 章 ユーザアプリケーションプログラムの作成 (COBOL 言語編)

COBOL 言語でのユーザアプリケーションプログラムインタフェースについて説明しています。

第 6 章 TP1/Client で使用できる要求文 (COBOL 言語編)

TP1/Client で使用できる要求文について説明しています。

第 7 章 環境設定

TP1/Client/P の環境設定とその手順について説明します。

第 8 章 定義

クライアント環境定義について説明しています。

第 9 章 運用コマンド

クライアント機能使用時に使用する運用コマンドの入力方法、記述形式、および詳細内容について説明しています。

第 10 章 障害対策

障害が発生した場合の対処方法について説明しています。

第 11 章 メッセージ

出力されるメッセージについて説明しています。

付録 A コード変換の仕様

文字コード変換機能で使用するコード変換の仕様について説明しています。

付録 B UNIX の CUP 実行プロセスでユーザが同時に使用できるファイル記述子の最大数

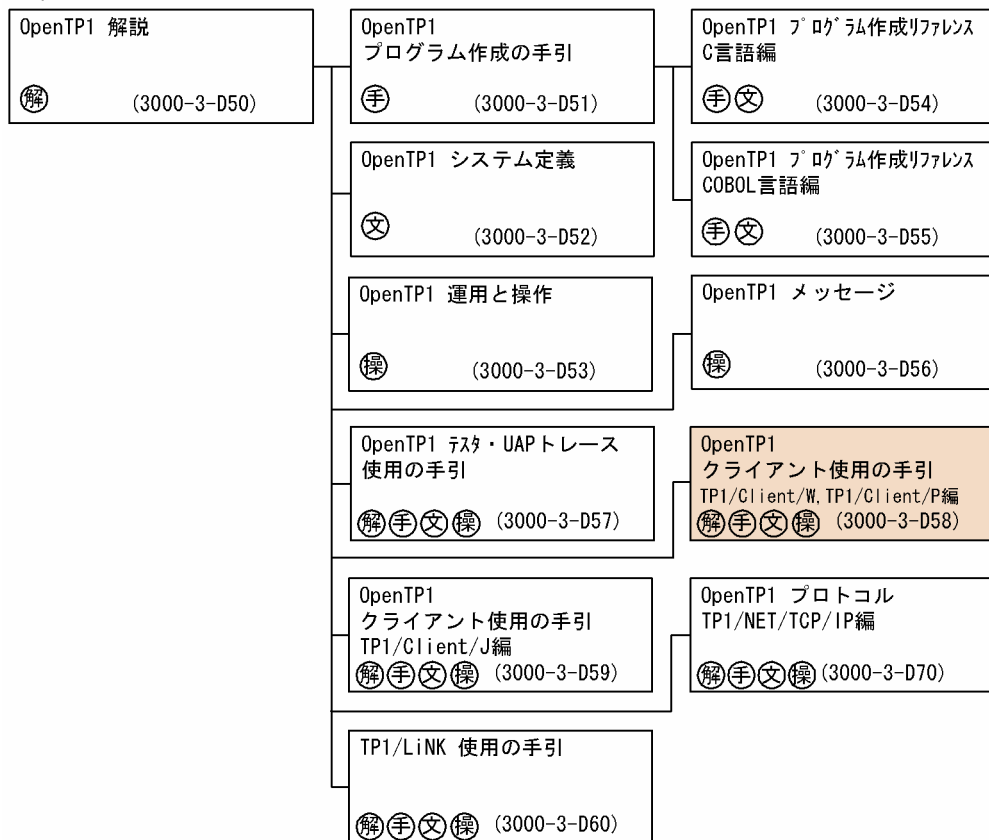
UNIX の CUP 実行プロセスで、ユーザが同時に使用できるファイル記述子の最大数の求め方について説明しています。

付録 C バージョンアップ時の変更点

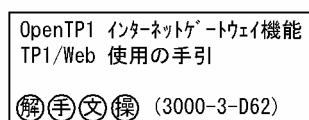
各バージョンでの関数、定義およびメッセージの変更点について説明しています。

■ 関連マニュアル

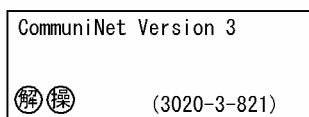
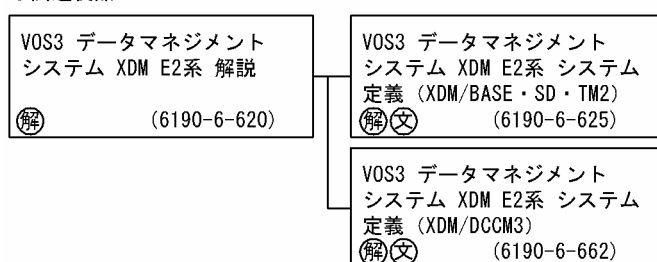
●OpenTP1 Version 7



●そのほかのOpenTP1関連



●関連製品

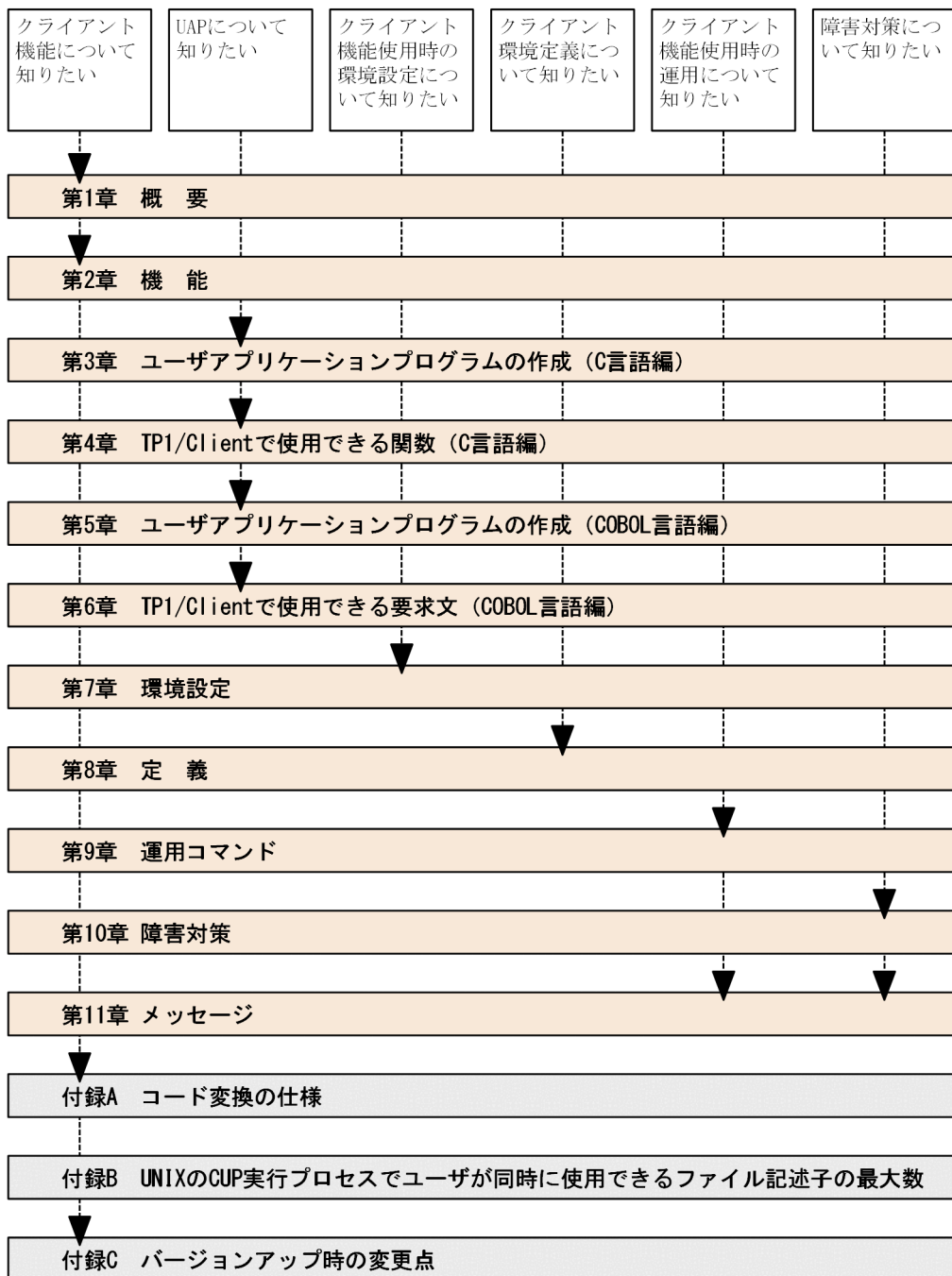


<記号>


- 解 : 解説書
- 手 : 手引書
- 文 : 文法書
- 操 : 操作書

■ 読書手順

このマニュアルは、利用目的に合わせて、章を選択して読むことができます。次の案内に従ってお読みいただくことをお勧めします。



(凡例)

 : 必ず読む項目

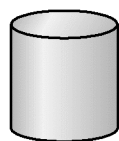
■ 図中で使用する記号

このマニュアルで使用する記号を、次のように定義します。

●ワークステーション,
パーソナル
コンピュータ, 端末



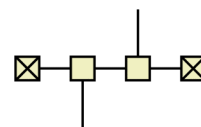
●ファイル



●プログラム



●ネットワーク
(LAN)



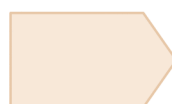
●画面の表示



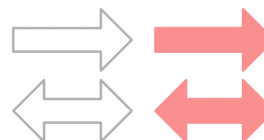
●プログラムの流れ



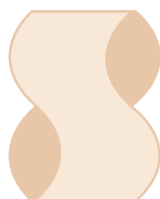
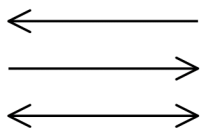
●工程, 作業項目の
流れ



●データ,
メッセージの流れ



●リモートプロシジャ ●コネクション
コール



■ このマニュアルでの表記

(1)製品名

このマニュアルでは、製品の名称を省略して表記しています。製品の名称と、このマニュアルでの表記を次に示します。

製品名称	略称
AIX 6.1	AIX
AIX 7.1	
AIX 7.2	
HP-UX 11i (PA-RISC)	HP-UX
HP-UX 11i V2 (IPF)	
HP-UX 11i V2 (PA-RISC)	
HP-UX 11i V3 (IPF)	
Itanium Processor Family	IPF

製品名称	略称		
Red Hat Enterprise Linux Server 6 (32-bit x86)	Linux		
Red Hat Enterprise Linux Server 6 (64-bit x86_64)			
Red Hat Enterprise Linux Server 7 (64-bit x86_64)			
Red Hat Enterprise Linux Server 8 (64-bit x86_64)			
Solaris 10	Solaris		
Oracle Solaris 11			
uCosminexus TP1/Client/P	TP1/Client/P	TP1/Client/P	TP1/Client
uCosminexus TP1/Client/P(64)	TP1/Client/P(64)		
uCosminexus TP1/Client/W	TP1/Client/W	TP1/Client/W	
uCosminexus TP1/Client/W(64)	TP1/Client/W(64)		
uCosminexus TP1/Extension 1	TP1/Extension 1		
uCosminexus TP1/NET/TCP/IP	TP1/NET/TCP/IP		
uCosminexus TP1/Online Tester	TP1/Online Tester		
uCosminexus TP1/LiNK	TP1/LiNK	TP1/Server	
uCosminexus TP1/Server Base	TP1/Server Base		
Windows 7 Enterprise	Windows 7	Windows 7	Windows
Windows 7 Professional			
Windows 7 Ultimate			
Windows 7 Enterprise(x64)	Windows 7 x64 Edition		
Windows 7 Professional(x64)			
Windows 7 Ultimate(x64)			
Windows 8 Enterprise	Windows 8	Windows 8	
Windows 8 Pro			
Windows 8 Enterprise(x64)	Windows 8 x64 Edition		
Windows 8 Pro(x64)			
Windows 8.1 Enterprise	Windows 8.1	Windows 8.1	
Windows 8.1 Pro			
Windows 8.1 Enterprise(x64)	Windows 8.1 x64 Edition		
Windows 8.1 Pro (x64)			

製品名称	略称		
Windows 10 Pro	Windows 10	Windows 10	Windows
Windows 10 Enterprise			
Windows 10 Pro (x64)	Windows 10 x64 Edition		
Windows 10 Enterprise(x64)			
Windows 11 Pro	Windows 11		
Windows 11 Enterprise			
Windows Server 2008 R2, Datacenter Edition	Windows Server 2008 R2	Windows Server 2008	
Windows Server 2008 R2, Enterprise Edition			
Windows Server 2008 R2, Standard Edition			
Windows Server 2012 Datacenter	Windows Server 2012		
Windows Server 2012 Standard			
Windows Server 2012 R2 Datacenter	Windows Server 2012 R2		
Windows Server 2012 R2 Standard			
Windows Server 2016 Datacenter	Windows Server 2016		
Windows Server 2016 Standard			
Windows Server 2019 Datacenter	Windows Server 2019		
Windows Server 2019 Standard			
Windows Server 2022 Datacenter	Windows Server 2022		
Windows Server 2022 Datacenter: Azure Edition			
Windows Server 2022 Standard			

- AIX, HP-UX, Linux, および Solaris を合わせて UNIX と表記しています。

(2)適用 OS による違いについて

Windows 版の製品をご使用になる場合、マニュアルの記述を次のように読み換えてください。

項目	マニュアルの表記	読み換え
環境変数の表記	\$aaaaaa 例 \$DCDIR	%aaaaaa% 例 %DCDIR%
パス名の区切り文字	:	;
ディレクトリの区切り文字	/	¥

項目	マニュアルの表記	読み換え
完全パス名	ルートディレクトリから指定します。 例 /tmp	先頭にドライブ文字を付加して、ルートディレクトリから指定します。 例 C:*tmp
実行形式ファイル名	ファイル名だけを指定します。 例 mcfmngd	ファイル名に拡張子を付加して指定します。 例 mcfmngd.exe
make コマンド	make	nmake

■ 略語一覧

このマニュアルで使用する英略語の一覧を次に示します。

英略語	英字での表記
API	<u>A</u> pplication <u>P</u> rogramming <u>I</u> nterface
CUP	<u>C</u> lient <u>U</u> ser <u>P</u> rogram
DAM	<u>D</u> irect <u>A</u> ccess <u>M</u> ethod
DLL	<u>D</u> ynamic <u>L</u> inking <u>L</u> ibrary
EBCDIC	<u>E</u> xtended <u>B</u> inary <u>C</u> oded <u>D</u> ecimal <u>I</u> nterchange <u>C</u> ode
ID	<u>I</u> dentifier
JIS	<u>J</u> apanese <u>I</u> ndustrial <u>S</u> tandard
KEIS	<u>K</u> anji <u>P</u> rocessing <u>E</u> xtended <u>I</u> nformation <u>S</u> ystem <u>C</u> ode
MHP	<u>M</u> essage <u>H</u> andling <u>P</u> rogram
MTU	<u>M</u> aximum <u>T</u> ransmission <u>U</u> nit
OLTP	<u>O</u> nline <u>T</u> ransaction <u>P</u> rocessing
PC	<u>P</u> ersonal <u>C</u> omputer
PRF	<u>P</u> e <u>R</u> formance
RPC	<u>R</u> emote <u>P</u> rocedure <u>C</u> all
SPP	<u>S</u> ervice <u>P</u> roviding <u>P</u> rogram
TCP/IP	<u>T</u> ransmission <u>C</u> ontrol <u>P</u> rotocol/ <u>I</u> nternet <u>P</u> rotocol
UAP	<u>U</u> ser <u>A</u> pplication <u>P</u> rogram
WAN	<u>W</u> ide <u>A</u> rea <u>N</u> etwork
WS	<u>W</u> orkstation

■ KB (キロバイト) などの単位表記について

1KB (キロバイト), 1MB (メガバイト), 1GB (ギガバイト), 1TB (テラバイト) はそれぞれ 1,024 バイト, 1,024² バイト, 1,024³ バイト, 1,024⁴ バイトです。

■ X/Open 発行の「X/Open CAE Specification Distributed Transaction Processing : The XATMI Specification」の内容から引用した部分

このマニュアルの記述のうち、次に示す部分は、上記ドキュメントの「Chapter 5 C Reference Manual Pages」の記述を、日本語訳したものです。

4章 TP1/Client で使用できる関数 (C 言語編)

4.8 XATMI インタフェース機能

■ 謝 辞

COBOL 言語仕様は、CODASYL (the Conference on Data Systems Languages : データシステムズ言語協議会) によって、開発された。原開発者に対し謝意を表すとともに、CODASYL の要求に従って以下の謝辞を掲げる。なお、この文章は、COBOL の原仕様書「CODASYL COBOL JOURNAL OF DEVELOPMENT 1984」の謝辞の一部を再掲するものである。

いかなる組織であっても、COBOL の原仕様書とその仕様の全体又は一部分を複製すること、マニュアルその他の資料のための土台として原仕様書のアイデアを利用することは自由である。ただし、その場合には、その刊行物のまえがきの一部として、次の謝辞を掲載しなければならない。書評などに短い文章を引用するときは、"COBOL" という名称を示せば謝辞全体を掲載する必要はない。

COBOL は産業界の言語であり、特定の団体や組織の所有物ではない。

CODASYL COBOL 委員会又は仕様変更の提案者は、このプログラミングシステムと言語の正確さや機能について、いかなる保証も与えない。さらに、それに関連する責任も負わない。

次に示す著作権表示付資料の著作者及び著作権者

FLOW-MATIC (Sperry Rand Corporation の商標), Programming for the Univac (R)

I and II, Data Automation Systems, Sperry Rand Corporation 著作権表示

1958 年, 1959 年 ;

IBM Commercial Translator Form No.F 28-8013, IBM 著作権表示 1959 年 ;

FACT, DSI 27A5260-2760, Minneapolis-Honeywell, 著作権表示 1960 年

は、これら全体又は一部分を COBOL の原仕様書中に利用することを許可した。この許可は、COBOL 原仕様書をプログラミングマニュアルや類似の刊行物に複製したり、利用したりする場合にまで拡張される。

目次

前書き	2
変更内容	5
はじめに	12

1 概要 34

1.1	クライアント機能の特長	35
1.2	TP1/Client の動作の仕組み	36

2 機能 39

2.1	ユーザ認証機能	40
2.1.1	ユーザ認証の実現方法	40
2.1.2	認証要求先の TP1/Server の指定	40
2.1.3	ユーザ認証の抑止	41
2.1.4	TP1/Server 以外のサーバとの通信	41
2.2	常設コネクション	42
2.2.1	常設コネクションの確立・解放	42
2.2.2	常設コネクションを使用する場合に関連する定義	43
2.2.3	DCCM3 論理端末に端末識別情報を通知する	44
2.2.4	常設コネクションを使用するときの注意事項	45
2.3	リモートプロシジャコール	46
2.3.1	RPC の実現方法	46
2.3.2	RPC でのデータの受け渡し	46
2.3.3	RPC の形態	46
2.3.4	連鎖 RPC	47
2.3.5	スケジュール機能	48
2.3.6	ノード間負荷バランス機能	49
2.3.7	RPC の時間監視	52
2.3.8	OpenTP1 以外のサーバへの RPC	52
2.3.9	ネームサービスを使用した RPC	55
2.3.10	マルチスケジューラ機能を使用した RPC	56
2.3.11	窓口となる TP1/Server の切り替え機能	59
2.3.12	窓口となる TP1/Server の負荷分散	62
2.3.13	データ圧縮機能	64
2.3.14	リモート API 機能	65
2.3.15	同期応答型 RPC タイムアウト時のサーバ負荷軽減	68

2.4	トランザクション制御	70
2.4.1	トランザクションの開始と同期点取得	70
2.4.2	同期点取得	71
2.4.3	リモートプロシジャコールの形態と同期点の関係	75
2.4.4	現在のトランザクションに関する識別子の取得	78
2.4.5	現在のトランザクションに関する情報の報告	78
2.4.6	障害発生時のトランザクションの同期点を検証する方法	79
2.4.7	トランザクションを制御するときの注意事項	80
2.5	TCP/IP 通信機能	81
2.5.1	メッセージの一方送信	81
2.5.2	メッセージの一方受信	82
2.5.3	メッセージの送受信	85
2.5.4	メッセージの組み立て機能と送達確認機能	87
2.5.5	ユースケースごとの設定方法とポートの割り当て	94
2.5.6	TCP/IP 通信機能を使用するときの注意事項	100
2.6	サーバからの一方通知受信機能	103
2.6.1	サーバからの一方通知受信機能の概要	103
2.6.2	一方通知連続受信機能の概要	104
2.6.3	一方通知連続受信機能を使用するときの注意事項	105
2.7	XATMI インタフェース機能	107
2.7.1	会話型サービス	108
2.7.2	会話型サービスの時間監視	111
2.7.3	イベントの受信	112
2.7.4	通信データの型	113
2.7.5	XATMI インタフェース機能を使用するときの注意事項	113
2.8	文字コード変換機能	114
2.8.1	コードマッピングテーブルを使用しない場合	114
2.8.2	コードマッピングテーブルを使用する場合	115
2.9	マルチスレッド機能	117
2.9.1	マルチスレッドに対応した CUP の処理の概要	117
2.9.2	マルチスレッドに対応しない関数の実行	118
2.9.3	マルチスレッド機能を使用するときの注意事項	119
2.10	オンラインテスタ機能	120
2.11	トラブルシュート機能	121
2.11.1	エラーログ機能	121
2.11.2	UAP トレース機能	121
2.11.3	ソケットトレース機能	122
2.11.4	モジュールトレース機能	122
2.11.5	TP1/Server の性能検証用トレース	123

2.11.6	トラブルシュート機能を使用するときの注意事項	125
2.12	ホスト名拡張機能	126
2.12.1	C 言語の関数の引数に指定できるホスト名長およびホスト名格納領域長	126
2.12.2	ホスト名拡張機能使用時の COBOL-UAP 作成用プログラム	127
2.12.3	クライアント環境定義のオペランドで指定できる文字数	128
2.12.4	ホスト名拡張機能を使用するときの注意事項	128
2.13	送信元ホスト指定機能	129
2.14	受信ポート固定機能	130
2.14.1	受信ポート固定機能を使用しない場合	130
2.14.2	受信ポート固定機能を使用する場合	131
3	ユーザアプリケーションプログラムの作成 (C 言語編)	132
3.1	関数インタフェース	133
3.1.1	関数の一覧	133
3.1.2	関数の記述形式	136
3.2	ユーザアプリケーションプログラムの翻訳と結合	138
3.2.1	UNIX 環境の場合の翻訳と結合	138
3.2.2	Windows 環境の場合の翻訳と結合	139
3.3	ユーザアプリケーションプログラムの作成例	143
3.3.1	CUP と SPP の作成	143
3.3.2	マルチスレッド対応のユーザアプリケーションプログラムの作成	148
4	TP1/Client で使用できる関数 (C 言語編)	150
4.1	関数を使用するときの注意事項	151
4.2	ユーザ認証機能	152
4.2.1	dc_clt_cltin_s - クライアントユーザの認証要求	152
4.2.2	dc_clt_cltout_s - クライアントユーザの認証解除	157
4.3	リモートプロシジャコール	158
4.3.1	dc_rpc_open_s - UAP の開始	158
4.3.2	dc_rpc_close_s - UAP の終了	159
4.3.3	dc_rpc_call_s - 遠隔サービスの要求	160
4.3.4	dc_rpc_call_to_s - 通信先を指定した遠隔サービスの要求	167
4.3.5	dc_rpc_set_watch_time_s - サービス応答待ち時間の更新	173
4.3.6	dc_rpc_get_watch_time_s - サービス応答待ち時間の参照	174
4.3.7	DCRPC_DIRECT_SCHEDULE - DCRPC_BINDING_TBL 構造体の作成	175
4.4	常設コネクション	177
4.4.1	dc_clt_connect_s - 常設コネクションの確立	177
4.4.2	dc_clt_disconnect_s - 常設コネクションの解放	179
4.4.3	dc_clt_set_raphost_s - 常設コネクション確立要求先の指定	181

4.4.4	dc_clt_get_raphost_s	常設コネクション確立要求先の取得	183
4.4.5	dc_clt_set_connect_inf_s	端末識別情報の設定	185
4.5		トランザクション制御	188
4.5.1	dc_trn_begin_s	トランザクションの開始	188
4.5.2	dc_trn_chained_commit_s	連鎖モードのコミット	190
4.5.3	dc_trn_chained_rollback_s	連鎖モードのロールバック	192
4.5.4	dc_trn_unchained_commit_s	非連鎖モードのコミット	194
4.5.5	dc_trn_unchained_rollback_s	非連鎖モードのロールバック	196
4.5.6	dc_clt_get_trnid_s	現在のトランザクションに関する識別子の取得	198
4.5.7	dc_trn_info_s	現在のトランザクションに関する情報の報告	200
4.6		TCP/IP 通信機能	202
4.6.1	dc_clt_send_s	メッセージの送信	202
4.6.2	dc_clt_receive_s	メッセージの受信	204
4.6.3	dc_clt_receive2_s	メッセージの受信 (障害時メッセージ受信)	207
4.6.4	dc_clt_assem_send_s	組み立てメッセージの送信	209
4.6.5	dc_clt_assem_receive_s	組み立てメッセージの受信	213
4.7		サーバからの一方通知受信機能	217
4.7.1	dc_clt_accept_notification_s	一方通知メッセージの受信	217
4.7.2	dc_clt_cancel_notification_s	一方通知待ち状態のキャンセル	221
4.7.3	dc_clt_open_notification_s	一方通知受信の開始	225
4.7.4	dc_clt_close_notification_s	一方通知受信の終了	228
4.7.5	dc_clt_chained_accept_notification_s	一方通知受信	229
4.8		XATMI インタフェース機能	232
4.8.1	tpalloc	型付きバッファの割り当て	232
4.8.2	tpfree	型付きバッファの解放	233
4.8.3	tpconnect	会話型サービスとのコネクションの確立	234
4.8.4	tpdiscon	会話型サービスとのコネクションの切断	237
4.8.5	tpsend	会話型サービスへのメッセージの送信	239
4.8.6	tprecv	会話型サービスからのメッセージの受信	242
4.9		文字コード変換機能 (コードマッピングテーブルを使用しない場合)	247
4.9.1	dc_clt_code_convert	文字コード変換	247
4.10		文字コード変換機能 (コードマッピングテーブルを使用する場合)	251
4.10.1	dc_clt_codeconv_open	文字コード変換の開始	251
4.10.2	dc_clt_codeconv_close	文字コード変換の終了	253
4.10.3	dc_clt_codeconv_exec	文字コード変換の実行	254
5		ユーザアプリケーションプログラムの作成 (COBOL 言語編)	259
5.1		COBOL-UAP 作成用プログラムの機能	260
5.1.1		各 UAP と機能の対応	260

5.1.2	COBOL-UAP 作成用プログラムの記述形式	263
5.2	ユーザアプリケーションプログラムの翻訳と結合	265
5.2.1	UNIX 環境の場合の翻訳と結合	265
5.2.2	Windows 環境の場合の翻訳と結合	266
5.3	COBOL 言語用テンプレート	269
5.3.1	COBOL 言語用テンプレートのファイル	269
5.3.2	COBOL 言語用テンプレートの使用方法	269
5.3.3	COBOL 言語用テンプレートについての使用上の注意	270
5.4	ユーザアプリケーションプログラムの作成例	271
5.4.1	CUP と SPP の作成	271
5.4.2	マルチスレッド対応のユーザアプリケーションプログラムの作成	275
6	TP1/Client で使用できる要求文 (COBOL 言語編)	282
6.1	要求文を使用するときの注意事項	283
6.2	ユーザ認証機能	284
6.2.1	CBLDCCLS('CLTIN ') - クライアントユーザの認証要求	284
6.2.2	CBLDCCLS('EXCLTIN ') - クライアントユーザの認証要求 (ホスト名長の拡張時)	288
6.2.3	CBLDCCLS('CLTOUT ') - クライアントユーザの認証解除	292
6.3	リモートプロシジャコール	294
6.3.1	CBLDCRPS('OPEN ') - UAP の開始	294
6.3.2	CBLDCRPS('CLOSE ') - UAP の終了	296
6.3.3	CBLDCRPS('CALL ') - 遠隔サービスの要求	298
6.3.4	CBLDCRPS('SETWATCH') - サービス応答待ち時間の更新	305
6.3.5	CBLDCRPS('GETWATCH') - サービス応答待ち時間の参照	306
6.4	常設コネクション	309
6.4.1	CBLDCCLS('CONNECT ') - 常設コネクションの確立	309
6.4.2	CBLDCCLS('DISCNCT ') - 常設コネクションの解放	312
6.4.3	CBLDCCLS('STRAPHST') - 常設コネクション確立要求先の指定	314
6.4.4	CBLDCCLS('GTRAPHST') - 常設コネクション確立要求先の取得	316
6.4.5	CBLDCCLS('STCONINF') - 端末識別情報の設定	318
6.5	トランザクション制御	321
6.5.1	CBLDCTRS('BEGIN ') - トランザクションの開始	321
6.5.2	CBLDCTRS('C-COMMIT') - 連鎖モードのコミット	323
6.5.3	CBLDCTRS('C-ROLL ') - 連鎖モードのロールバック	326
6.5.4	CBLDCTRS('U-COMMIT') - 非連鎖モードのコミット	328
6.5.5	CBLDCTRS('U-ROLL ') - 非連鎖モードのロールバック	330
6.5.6	CBLDCTRS('INFO ') - 現在のトランザクションに関する情報の報告	332
6.5.7	CBLDCCLS('GETTRNID') - 現在のトランザクションに関する識別子の取得	334
6.6	TCP/IP 通信機能	336

- 6.6.1 CBLDCCLS('SEND ') – メッセージの送信 336
- 6.6.2 CBLDCCLS('EXSEND ') – メッセージの送信 (ホスト名長の拡張時) 339
- 6.6.3 CBLDCCLS('RECEIVE ') – メッセージの受信 342
- 6.6.4 CBLDCCLS('RECEIVE2') – メッセージの受信 (障害時メッセージ受信) 345
- 6.6.5 CBLDCCLS('ASMSEND ') – 組み立てメッセージの送信 348
- 6.6.6 CBLDCCLS('ASMRECV ') – 組み立てメッセージの受信 352
- 6.7 サーバからの一方通知受信機能 356
- 6.7.1 CBLDCCLS('NOTIFY ') – 一方通知メッセージの受信 356
- 6.7.2 CBLDCCLS('EXNACPT ') – 一方通知メッセージの受信 (ホスト名長の拡張時) 360
- 6.7.3 CBLDCCLS('CANCEL ') – 一方通知待ち状態のキャンセル 364
- 6.7.4 CBLDCCLS('EXNCANCL') – 一方通知待ち状態のキャンセル (ホスト名長の拡張時) 367
- 6.7.5 CBLDCCLS('O-NOTIFY') – 一方通知受信の開始 371
- 6.7.6 CBLDCCLS('C-NOTIFY') – 一方通知受信の終了 374
- 6.7.7 CBLDCCLS('A-NOTIFY') – 一方通知受信 375
- 6.7.8 CBLDCCLS('EXNCACPT') – 一方通知受信 (ホスト名長の拡張時) 378
- 6.8 文字コード変換機能 (コードマッピングテーブルを使用しない場合) 381
- 6.8.1 CBLDCUTL('CODECNV ') – 文字コード変換 381
- 6.9 文字コード変換機能 (コードマッピングテーブルを使用する場合) 384
- 6.9.1 CBLDCUTL('CNVOPN ') – 文字コード変換の開始 384
- 6.9.2 CBLDCUTL('CNVCLS ') – 文字コード変換の終了 386
- 6.9.3 CBLDCUTL('CNVEXEC ') – 文字コード変換の実行 388

7 環境設定 392

- 7.1 Windows の C ランタイムライブラリのインストール 393

8 定義 394

- 8.1 定義の概要 395
 - 8.1.1 クライアント環境定義の一覧 395
 - 8.1.2 定義の規則 399
- 8.2 クライアント環境定義の詳細 401
 - 8.2.1 TP1/Client/W の形式 401
 - 8.2.2 TP1/Client/P の形式 404
 - 8.2.3 TP1/Client/W の設定方法 405
 - 8.2.4 TP1/Client/P の設定方法 406
 - 8.2.5 TP1/Client/W と TP1/Client/P で共通のオペランド 407
 - 8.2.6 TP1/Client/P 固有のオペランド 432
 - 8.2.7 TP1/Client/W の注意事項 432
 - 8.2.8 TP1/Client/P の注意事項 433

9 運用コマンド 434

- 9.1 運用コマンドの記述形式 435
- 9.2 運用コマンドの詳細 436
- 9.2.1 cltdump (トレースの編集出力) 436

10 障害対策 466

- 10.1 通信障害 467
- 10.2 クライアント側の障害 468
- 10.3 XDM/DCCM3 へ遠隔操作要求時の障害 469
- 10.3.1 XDM/DCCM3 へ遠隔操作要求時の DCRPCER_NET_DOWN 469
- 10.3.2 XDM/DCCM3 での二重入力エラー検出 469

11 メッセージ 470

- 11.1 メッセージの出力形式 471
- 11.2 メッセージの記述形式 472
- 11.2.1 記述形式 472
- 11.2.2 メッセージ ID の記号 472
- 11.3 メッセージ一覧 474

付録 491

- 付録 A コード変換の仕様 492
- 付録 A.1 TP1/Client/P でサポートするコード範囲 492
- 付録 A.2 シフト JIS コードと KEIS コードの変換 493
- 付録 A.3 コード変換例 505
- 付録 A.4 コード変換をするときの注意事項 507
- 付録 B UNIX の CUP 実行プロセスでユーザが同時に使用できるファイル記述子の最大数 508
- 付録 C バージョンアップ時の変更点 509
- 付録 C.1 07-02 での変更点 509
- 付録 C.2 07-01 での変更点 510
- 付録 C.3 07-00 での変更点 510

索引 513

目次

図 1-1	TP1/Server と TP1/Client の関係	35
図 1-2	TP1/Client の動作の仕組み (1/2)	37
図 1-3	TP1/Client の動作の仕組み (2/2)	38
図 2-1	常設コネクションの確立・解放処理	43
図 2-2	RPC のデータの受け渡し	46
図 2-3	同期応答型 RPC の処理の流れ	47
図 2-4	非応答型 RPC の処理の流れ	47
図 2-5	OpenTP1 以外のサーバへの RPC の処理の流れ	54
図 2-6	データ圧縮機能の概要	65
図 2-7	リモート API 機能	66
図 2-8	同期応答型 RPC タイムアウト時のサーバ負荷軽減機能の処理概要	69
図 2-9	トランザクションと RPC の関係	71
図 2-10	トランザクションの連鎖, 非連鎖モード	72
図 2-11	トランザクションのロールバック (TP1/Server の処理でエラーが発生した場合)	73
図 2-12	トランザクションのロールバック (ロールバック要求の関数を呼び出した場合)	74
図 2-13	同期応答型 RPC と同期点の関係	76
図 2-14	非応答型 RPC と同期点の関係	76
図 2-15	連鎖 RPC と同期点の関係 (トランザクションの連鎖 RPC)	77
図 2-16	連鎖 RPC と同期点の関係 (非トランザクションの連鎖 RPC で終了しない指定をした場合)	78
図 2-17	障害発生時のトランザクションの同期点を検証する方法	79
図 2-18	メッセージの一方送信	82
図 2-19	メッセージの一方受信	84
図 2-20	メッセージの一方受信 (障害発生時)	85
図 2-21	メッセージの送受信	87
図 2-22	メッセージの組み立て機能使用時の送受信メッセージの形式	88
図 2-23	送達確認機能使用時のメッセージの形式	89
図 2-24	応答専用データの形式	89
図 2-25	メッセージの組み立て機能使用時の送信の流れ	90
図 2-26	メッセージの組み立て機能使用時の受信の流れ	91
図 2-27	メッセージの送達確認機能使用時の送信の流れ	91
図 2-28	メッセージの送達確認機能使用時の受信の流れ	92
図 2-29	送受信メッセージの衝突時の処理の流れ	93
図 2-30	MHP がサーバ型でメッセージの一方送信を行う場合	95
図 2-31	MHP がクライアント型でメッセージの一方受信を行う場合	96

図 2-32	MHP がサーバ型でメッセージの送受信を一つの接続で行う場合	97
図 2-33	MHP がクライアント型でメッセージの送受信を一つの接続で行う場合	98
図 2-34	サーバからの一方通知受信機能の処理の流れ	103
図 2-35	一方通知連続受信機能の処理の流れ	105
図 2-36	会話型サービスで通信する場合の処理の流れ	108
図 2-37	会話型サービスの通信形態	111
図 2-38	文字コード変換機能の概要	115
図 2-39	マルチスレッド環境で動作する CUP の処理の概要	117
図 2-40	マルチスレッド環境に対応しない関数の実行 1	118
図 2-41	マルチスレッド環境に対応しない関数の実行 2	118
図 2-42	送信元ホスト指定機能を使用しない場合と使用する場合	129
図 2-43	受信ポート固定機能を使用しない場合 (スケジューラダイレクト機能を使用した RPC)	130
図 2-44	受信ポート固定機能を使用する場合 (スケジューラダイレクト機能を使用した RPC)	131
図 3-1	CUP 作成時の手順	140
図 3-2	CUP と SPP の構成例	143
図 5-1	COBOL CUP 新規作成時の手順例	266
図 A-1	文字コードの対応 (DCCLT_CNV_SPCHAN を指定した場合)	506
図 A-2	文字コードの対応 (DCCLT_CNV_TAB を指定した場合)	506
図 A-3	文字コードの対応 (DCCLT_CNV_CNTL を指定した場合)	507

表目次

表 2-1	クライアント環境定義のオペランドの指定とスケジューラデーモンの関連	58
表 2-2	関数の設定, および定義の指定と常設コネクション確立要求先との関係	67
表 2-3	使用する機能とクライアント環境定義 DCCLTDELIVERYCHECK の指定値の関係	88
表 2-4	メッセージ長の妥当性チェックで発生するエラーと対処	93
表 2-5	応答専用データの妥当性チェックで発生するエラーと対処	94
表 2-6	受信メッセージの妥当性チェックで発生するエラーと対処	94
表 2-7	TCP/IP 通信機能のユースケース	94
表 2-8	ユースケースごとの設定方法と使用するポート番号	99
表 2-9	トランザクションを生成するタイミング	110
表 2-10	ファイルサイズを指定するクライアント環境定義と出力されるファイル名	125
表 2-11	C 言語の関数の引数に指定できるホスト名長	126
表 2-12	C 言語の関数の引数に指定できるホスト名格納領域長	127
表 2-13	ホスト名拡張機能利用時に使用する CALL 文で呼び出す COBOL-UAP 作成用プログラム	127
表 3-1	関数の一覧	133
表 3-2	翻訳時の必須オプション	141
表 5-1	TP1/Client の機能と COBOL-UAP 作成用プログラムとの対応	260
表 8-1	クライアント環境定義の一覧	395
表 9-1	dc_rpc_open_s 関数の呼び出し情報 (関数コード: 1)	440
表 9-2	dc_rpc_close_s 関数の呼び出し情報 (関数コード: 2)	440
表 9-3	dc_rpc_call_s 関数の呼び出し情報 (関数コード: 3)	440
表 9-4	dc_clt_cltin_s 関数の呼び出し情報 (関数コード: 4)	441
表 9-5	dc_clt_cltout_s 関数の呼び出し情報 (関数コード: 5)	441
表 9-6	dc_clt_send_s 関数の呼び出し情報 (関数コード: 6)	441
表 9-7	dc_clt_receive_s 関数の呼び出し情報 (関数コード: 7)	442
表 9-8	dc_trn_begin_s 関数の呼び出し情報 (関数コード: 8)	442
表 9-9	dc_trn_chained_commit_s 関数の呼び出し情報 (関数コード: 9)	442
表 9-10	dc_clt_set_raphost_s 関数の呼び出し情報 (関数コード: 1a)	443
表 9-11	dc_clt_get_raphost_s 関数の呼び出し情報 (関数コード: 1b)	443
表 9-12	dc_clt_assem_send_s 関数の呼び出し情報 (関数コード: 1c)	443
表 9-13	dc_clt_assem_receive_s 関数の呼び出し情報 (関数コード: 1d)	444
表 9-14	dc_trn_chained_rollback_s 関数の呼び出し情報 (関数コード: a)	444
表 9-15	dc_trn_unchained_commit_s 関数の呼び出し情報 (関数コード: b)	444
表 9-16	dc_trn_unchained_rollback_s 関数の呼び出し情報 (関数コード: c)	445
表 9-17	dc_trn_info_s 関数の呼び出し情報 (関数コード: d)	445

表 9-18	dc_clt_get_trnid_s 関数の呼び出し情報 (関数コード : e)	445
表 9-19	dc_rpc_get_watch_time_s 関数の呼び出し情報 (関数コード : f)	446
表 9-20	dc_rpc_set_watch_time_s 関数の呼び出し情報 (関数コード : 10)	446
表 9-21	dc_clt_connect_s 関数の呼び出し情報 (関数コード : 13)	446
表 9-22	dc_clt_disconnect_s 関数の呼び出し情報 (関数コード : 14)	446
表 9-23	dc_clt_receive2_s 関数の呼び出し情報 (関数コード : 17)	447
表 9-24	dc_clt_set_connect_inf_s 関数の呼び出し情報 (関数コード : 18)	447
表 9-25	dc_rpc_call_to_s 関数の呼び出し情報 (関数コード : 19)	447
表 9-26	dc_clt_accept_notification_s 関数の呼び出し情報 (関数コード : 100)	448
表 9-27	dc_clt_cancel_notification_s 関数の呼び出し情報 (関数コード : 101)	449
表 9-28	dc_clt_open_notification_s 関数の呼び出し情報 (関数コード : 102)	449
表 9-29	dc_clt_close_notification_s 関数の呼び出し情報 (関数コード : 103)	450
表 9-30	dc_clt_chained_accept_notification_s 関数の呼び出し情報 (関数コード : 104)	450
表 9-31	tpalloc 関数の呼び出し情報 (関数コード : 200)	450
表 9-32	tpfree 関数の呼び出し情報 (関数コード : 201)	451
表 9-33	tpconnect 関数の呼び出し情報 (関数コード : 202)	451
表 9-34	tpdiscon 関数の呼び出し情報 (関数コード : 203)	451
表 9-35	tpsend 関数の呼び出し情報 (関数コード : 204)	451
表 9-36	tprecv 関数の呼び出し情報 (関数コード : 205)	452
表 9-37	dc_rpc_open_s 関数の呼び出し情報 (関数コード : 1)	452
表 9-38	dc_rpc_close_s 関数の呼び出し情報 (関数コード : 2)	453
表 9-39	dc_rpc_call_s 関数の呼び出し情報 (関数コード : 3)	453
表 9-40	dc_clt_cltin_s 関数の呼び出し情報 (関数コード : 4)	453
表 9-41	dc_clt_cltout_s 関数の呼び出し情報 (関数コード : 5)	454
表 9-42	dc_clt_send_s 関数の呼び出し情報 (関数コード : 6)	454
表 9-43	dc_clt_receive_s 関数の呼び出し情報 (関数コード : 7)	455
表 9-44	dc_trn_begin_s 関数の呼び出し情報 (関数コード : 8)	455
表 9-45	dc_trn_chained_commit_s 関数の呼び出し情報 (関数コード : 9)	455
表 9-46	dc_clt_set_raphost_s 関数の呼び出し情報 (関数コード : 1a)	456
表 9-47	dc_clt_get_raphost_s 関数の呼び出し情報 (関数コード : 1b)	456
表 9-48	dc_clt_assem_send_s 関数の呼び出し情報 (関数コード : 1c)	456
表 9-49	dc_clt_assem_receive_s 関数の呼び出し情報 (関数コード : 1d)	457
表 9-50	dc_trn_chained_rollback_s 関数の呼び出し情報 (関数コード : a)	457
表 9-51	dc_trn_unchained_commit_s 関数の呼び出し情報 (関数コード : b)	457
表 9-52	dc_trn_unchained_rollback_s 関数の呼び出し情報 (関数コード : c)	458
表 9-53	dc_trn_info_s 関数の呼び出し情報 (関数コード : d)	458
表 9-54	dc_clt_get_trnid_s 関数の呼び出し情報 (関数コード : e)	458
表 9-55	dc_rpc_get_watch_time_s 関数の呼び出し情報 (関数コード : f)	459
表 9-56	dc_rpc_set_watch_time_s 関数の呼び出し情報 (関数コード : 10)	459

表 9-57	dc_clt_connect_s 関数の呼び出し情報 (関数コード : 13)	459
表 9-58	dc_clt_disconnect_s 関数の呼び出し情報 (関数コード : 14)	459
表 9-59	dc_clt_receive2_s 関数の呼び出し情報 (関数コード : 17)	460
表 9-60	dc_clt_set_connect_inf_s 関数の呼び出し情報 (関数コード : 18)	460
表 9-61	dc_rpc_call_to_s 関数の呼び出し情報 (関数コード : 19)	460
表 9-62	dc_clt_accept_notification_s 関数の呼び出し情報 (関数コード : 100)	461
表 9-63	dc_clt_cancel_notification_s 関数の呼び出し情報 (関数コード : 101)	462
表 9-64	dc_clt_open_notification_s 関数の呼び出し情報 (関数コード : 102)	462
表 9-65	dc_clt_close_notification_s 関数の呼び出し情報 (関数コード : 103)	463
表 9-66	dc_clt_chained_accept_notification_s 関数の呼び出し情報 (関数コード : 104)	463
表 9-67	tpalloc 関数の呼び出し情報 (関数コード : 200)	463
表 9-68	tpfree 関数の呼び出し情報 (関数コード : 201)	464
表 9-69	tpconnect 関数の呼び出し情報 (関数コード : 202)	464
表 9-70	tpdiscon 関数の呼び出し情報 (関数コード : 203)	464
表 9-71	tpsend 関数の呼び出し情報 (関数コード : 204)	465
表 9-72	tprecv 関数の呼び出し情報 (関数コード : 205)	465
表 11-1	メッセージの種類	472
表 11-2	メッセージの出力先種別	473
表 A-1	コード体系	492
表 A-2	シフト JIS コード / KEIS コード変換仕様	494
表 A-3	シフト JIS コードから '83 版 KEIS コードへの変換	494
表 A-4	'83 版 KEIS コードからシフト JIS コードへの変換	495
表 A-5	シフト JIS コード / '78 版 KEIS コード / '83 版 KEIS コードの対応 (1)	497
表 A-6	シフト JIS コード / '78 版 KEIS コード / '83 版 KEIS コードの対応 (2)	498
表 A-7	JIS コードから EBCDIK コードへの変換表 (1)	499
表 A-8	JIS コードから EBCDIK コードへの変換表 (2)	500
表 A-9	EBCDIK コードから JIS コードへの変換表 (1)	501
表 A-10	EBCDIK コードから JIS コードへの変換表 (2)	502
表 A-11	JIS コードから EBCDIC コードへの変換表 (1)	503
表 A-12	JIS コードから EBCDIC コードへの変換表 (2)	503
表 A-13	EBCDIC コードから JIS コードへの変換表 (1)	504
表 A-14	EBCDIC コードから JIS コードへの変換表 (2)	505
表 C-1	TP1/Client/W 07-02, TP1/Client/P 07-02 での関数, 定義およびコマンドの追加と削除	509
表 C-2	TP1/Client/W 07-02, TP1/Client/P 07-02 での動作の変更点	509
表 C-3	TP1/Client/W 07-02, TP1/Client/P 07-02 でのデフォルト値の変更点	510
表 C-4	TP1/Client/W 07-01, TP1/Client/P 07-01 での関数, 定義およびコマンドの追加と削除	510
表 C-5	TP1/Client/W 07-00, TP1/Client/P 07-00 での関数, 定義およびコマンドの追加と削除	511
表 C-6	TP1/Client/W 07-00, TP1/Client/P 07-00 での動作の変更点	511

1

概要

OpenTP1 のクライアント機能の特長を説明します。

1.1 クライアント機能の特長

OpenTP1 のクライアント機能 (TP1/Client) を使用すると、ワークステーション (WS), またはパーソナルコンピュータ (PC) から OpenTP1 のサーバ UAP へ、リモートプロシジャコールでサービスを要求できます。WS, または PC 側のサービスを要求するプログラムを CUP といいます。CUP がサービスを要求できるサーバ UAP は、OpenTP1 のサービス提供プログラム (SPP) です。

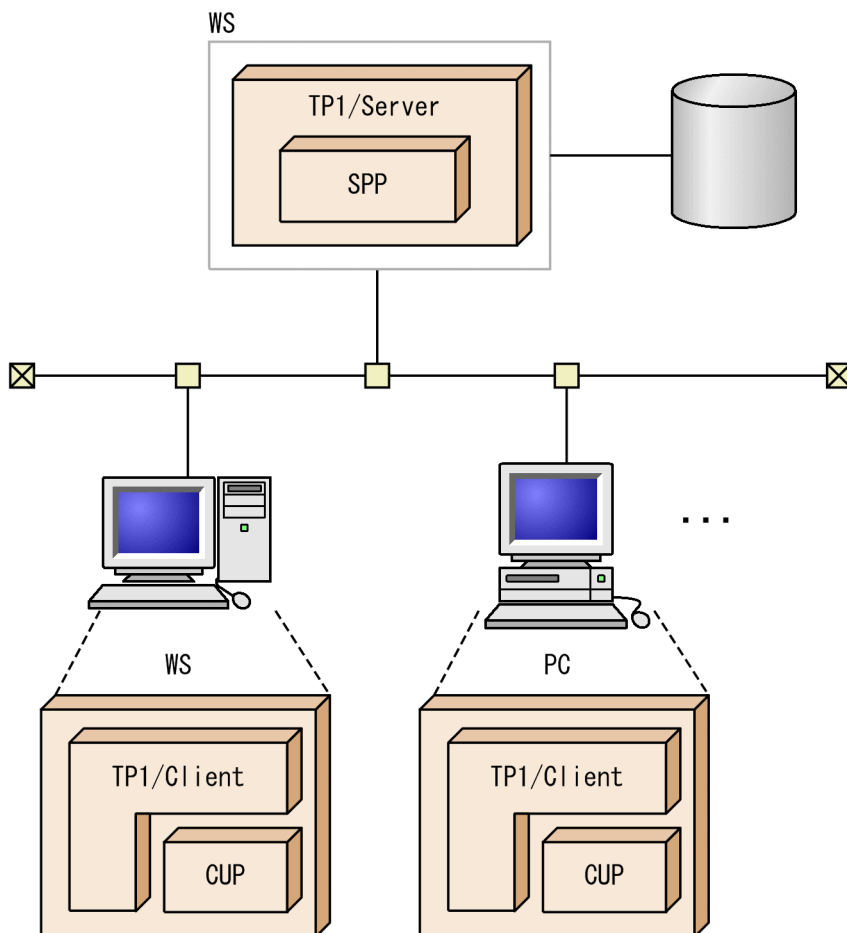
TP1/Client を使用すると、CUP が起動した SPP からトランザクションを起動することもできます。そのため、LAN 上の WS や PC をクライアントとする分散 OLTP 環境を構築できます。

なお、このマニュアルでは、WS, PC に固有の TP1/Client について説明するときには、それぞれ、TP1/Client/W, TP1/Client/P と記載します。

また、TP1/Server Base と TP1/LiNK の総称として、TP1/Server と記載します。

TP1/Server と TP1/Client の関係を次の図に示します。

図 1-1 TP1/Server と TP1/Client の関係



1.2 TP1/Client の動作の仕組み

TP1/Client は、次の機能を実行する場合に、ある特定の TP1/Server を窓口として使用します。

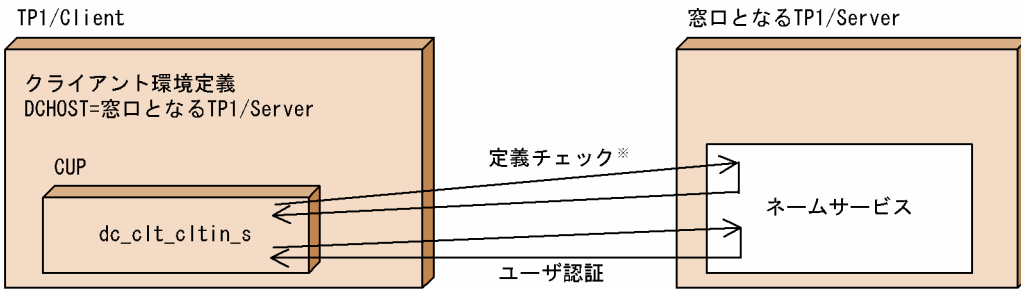
- ユーザ認証機能
- リモートプロシジャコール (RPC)
- ネームサービスを使用しない RPC
- トランザクションを管理するプロセスを割り当てる RPC

窓口となる TP1/Server は、クライアント環境定義DCHOST で定義するか、ユーザ認証要求時に関数の引数で指定します。

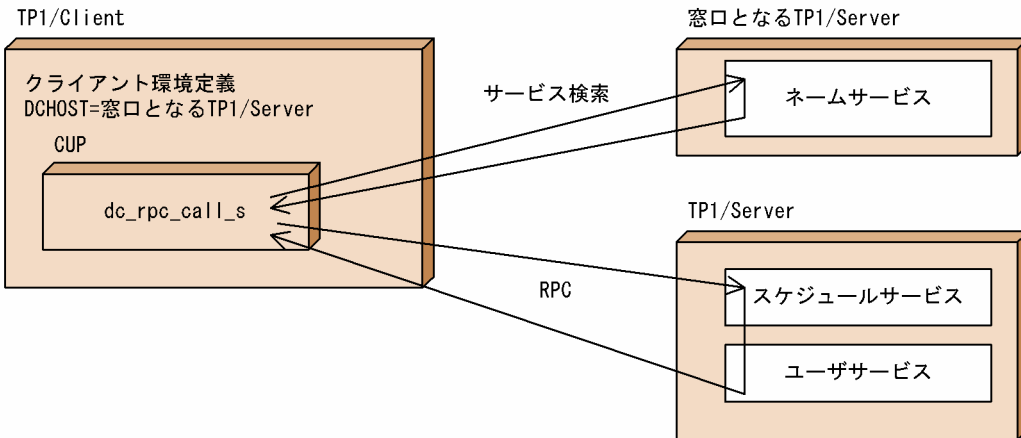
TP1/Client の動作の仕組みを、次の図に示します。

図 1-2 TP1/Client の動作の仕組み (1/2)

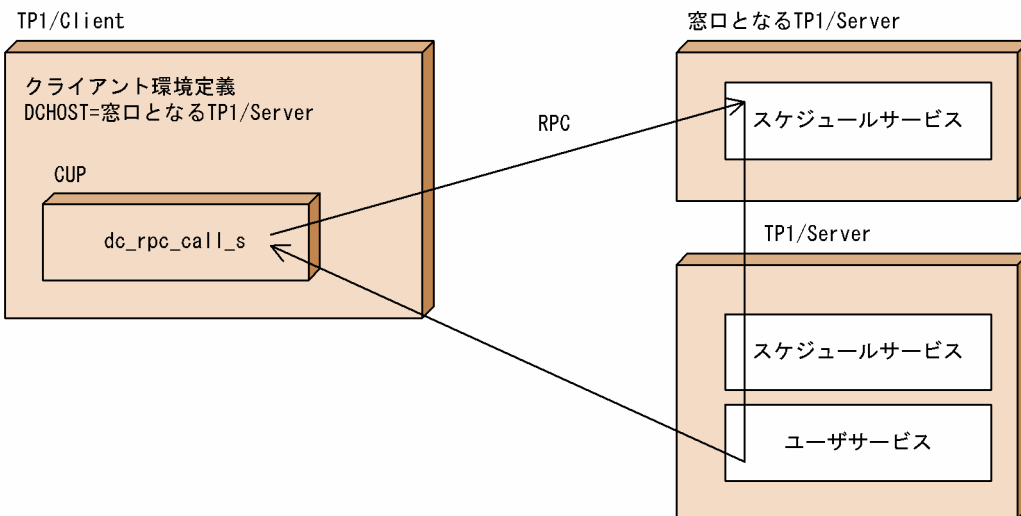
● ユーザ認証



● RPC



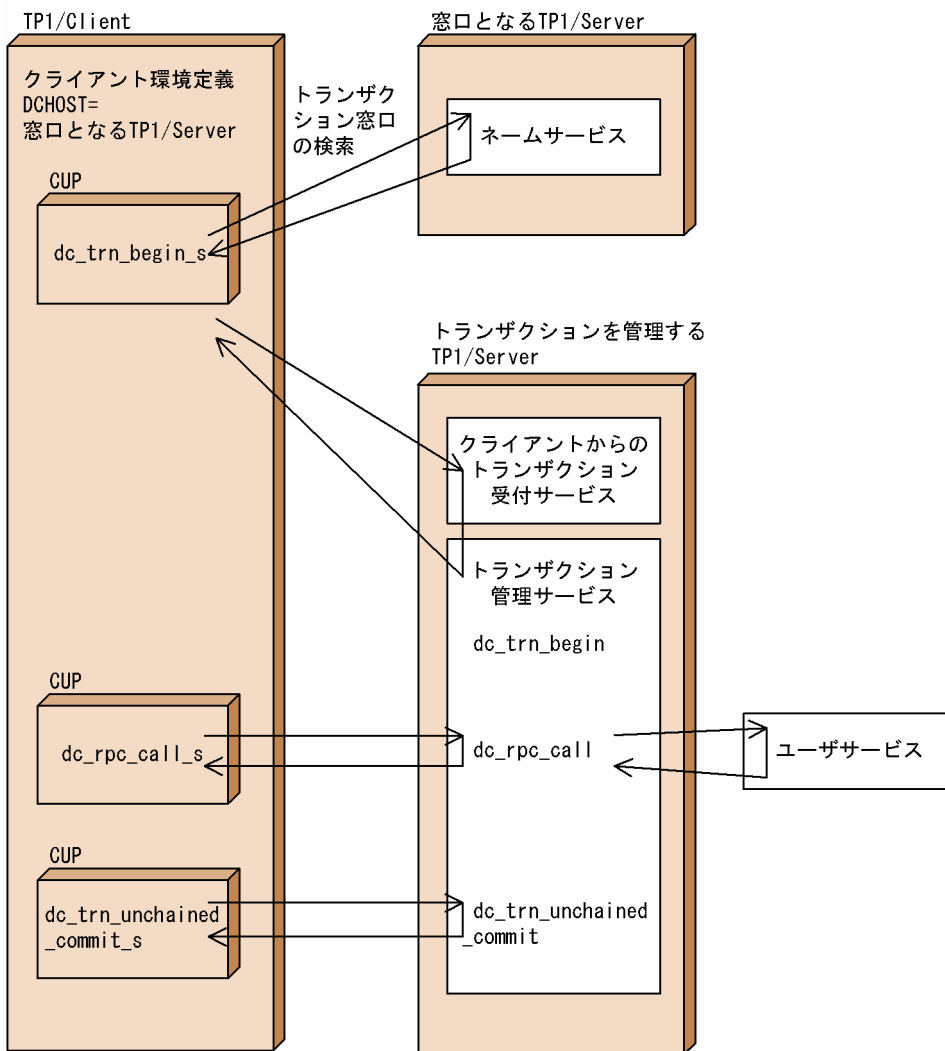
● ネームサービスを使用しないRPC



注※ システム共通定義のclient_uid_checkオペランドの指定をチェックします。

図 1-3 TP1/Client の動作の仕組み (2/2)

- トランザクションを管理するプロセスを割り当てるRPC



2

機能

OpenTP1 のクライアント機能について説明します。

この章では、各機能の DLL を呼び出すときの C 言語の関数名 (dc_xxx_xxx_s) で説明します。通常オブジェクトライブラリの関数 (dc_xxx_xxx) および COBOL 言語を使う場合は、各機能に対応する通常オブジェクトライブラリの関数 (dc_xxx_xxx) および COBOL 言語の機能に置き換えて読んでください。

2.1 ユーザ認証機能

TP1/Client では、TP1/Server が起動しているホストに登録されているユーザだけが TP1/Server 側のサービスを受けられます。これを**ユーザ認証機能**といいます。ユーザ認証機能を使用すると、TP1/Server 側のサービスを受けられるクライアントユーザを制限できます。

2.1.1 ユーザ認証の実現方法

TP1/Client から TP1/Server へサービスを要求する時には、CUP から `dc_clt_cltin_s` 関数を実行する必要があります。このときにユーザ名、パスワードを指定して、TP1/Server 側のサービスを受けられるユーザであることの認証を要求します。

TP1/Server 側では、UNIX や Windows のユーザ管理情報を基にクライアントユーザの認証を行います。ユーザ認証機能を使用するためには TP1/Server 側で次の準備をする必要があります。

- UNIX システムであれば `/etc/passwd` にログイン名とパスワードを登録し、Windows システムであればユーザマネージャでユーザ登録をしておきます。
- TP1/Server のシステム共通定義 `client_uid_check` には Y を指定します。N の場合は、ユーザ管理情報にないユーザ名でも認証を受けることができます。

なお、TP1/Server を実行する OS によっては、ユーザ認証機能を使用できません。TP1/Server 側のマニュアルなどをご確認ください。

ユーザ認証機能を使用できない場合は、ユーザ認証を抑止することをお勧めします。ユーザ認証の抑止については、「[2.1.3 ユーザ認証の抑止](#)」を参照してください。

2.1.2 認証要求先の TP1/Server の指定

認証要求を行う先の、窓口となる TP1/Server は次の順に評価し、決定します。

1. `dc_clt_cltin_s` 関数の引数 `target_host` に指定されたノード
2. クライアント環境定義の `DCHOST` で指定されたノード (`dc_clt_cltin_s` 関数の引数 `target_host` に `NULL` を指定した場合)
3. 任意の TP1/Server に問い合わせ (ブロードキャスト)、最初に応答を返してきたノード (`dc_clt_cltin_s` 関数の引数 `target_host` に `NULL` を指定し、かつクライアント環境定義 `DCHOST` の指定がなかった場合)

なお、認証要求先として、複数のホスト (ノード) を指定することができます。

複数ホストが指定された場合、先頭に指定されたホストが認証要求先となります。接続に失敗した場合、次に指定されたホストが認証要求先となり、接続に成功するまでこれを繰り返します。

また、複数ホストの中から、認証要求先をランダムに選択したい場合は、クライアント環境定義 DCHOSTSELECT に Y を指定します。接続に失敗した場合、残りのホストの中から再びランダムに選択し、接続に成功するまでこれを繰り返します。

指定されたすべてのホストとの接続に失敗した場合、dc_clt_cltin_s 関数は DCCLTER_OLTF_NOT_UP でエラーリターンします。

詳細については「[2.3.12 窓口となる TP1/Server の負荷分散](#)」を参照してください。

2.1.3 ユーザ認証の抑止

リモート API 機能を使用する場合など、ユーザ認証を抑止する（通信を発生させないようにする）必要がある場合は、クライアント環境定義で「DCCLTAUTHENT=N」と定義するか、または dc_clt_cltin_s 関数の引数 flags に DCCLT_NO_AUTHENT を指定してください。

ユーザ認証を抑止する場合でも、dc_clt_cltin_s 関数は必ず発行します。

2.1.4 TP1/Server 以外のサーバとの通信

クライアント環境定義で「DCCLTNOSEVER=Y」と定義すると、TP1/Server が存在しない環境で、DCCM3 論理端末など TP1/Server 以外のサーバと通信できます。

TP1/Server 以外のサーバと通信する場合でも、dc_clt_cltin_s 関数は必ず実行してください。

dc_clt_cltin_s 関数の引数 logname には、NULL 以外の任意の値を指定してください。NULL を指定した場合、dc_clt_cltin_s 関数は DCCLTER_INVALID_ARGS でエラーリターンします。

2.2 常設コネクション

TP1/Client では、CUP とサーバ間のコネクションを常設したままでメッセージを送受信できます。このようなコネクションを常設コネクションといいます。常設コネクション確立要求先のサーバには、TP1/Server と VOS3 XDM/DCCM3 の論理端末を指定できます。また、常設コネクションを使って、リモート API 機能を実現しています。リモート API 機能については、「[2.3.14 リモート API 機能](#)」を参照してください。

常設コネクションを確立することで、コネクション確立・解放のための制御用パケットを減らすことができ、通信の効率が向上します。

なお、これ以降このマニュアルでは、VOS3 XDM/DCCM3 のことを「DCCM3」と表記します。

2.2.1 常設コネクションの確立・解放

CUP は `dc_clt_connect_s` 関数でサーバのクライアント拡張サービスに常設コネクションの確立を要求します。確立要求を受け付けた受け付けプロセスは、常設コネクション確立要求を CUP 実行プロセスに渡します。CUP が、CUP 実行プロセスから常設コネクションの確立応答のメッセージを受信した時点で、CUP と CUP 実行プロセスの間に常設コネクションが確立されます。

CUP と CUP 実行プロセスは 1 : 1 に対応します。一つの OpenTP1 ノードで同時に処理を受け付けられる CUP の最大数は、CUP 実行プロセスの最大プロセス数（クライアントサービス定義 `cup_parallel_count` の指定）と同じです。

常設コネクションを確立するノードは、ユーザ認証の有無によって次のようになります。ユーザ認証機能については、「[2.1 ユーザ認証機能](#)」を参照してください。

ユーザ認証を行う場合

ユーザ認証に成功した OpenTP1 ノードです。

ユーザ認証を抑止する場合

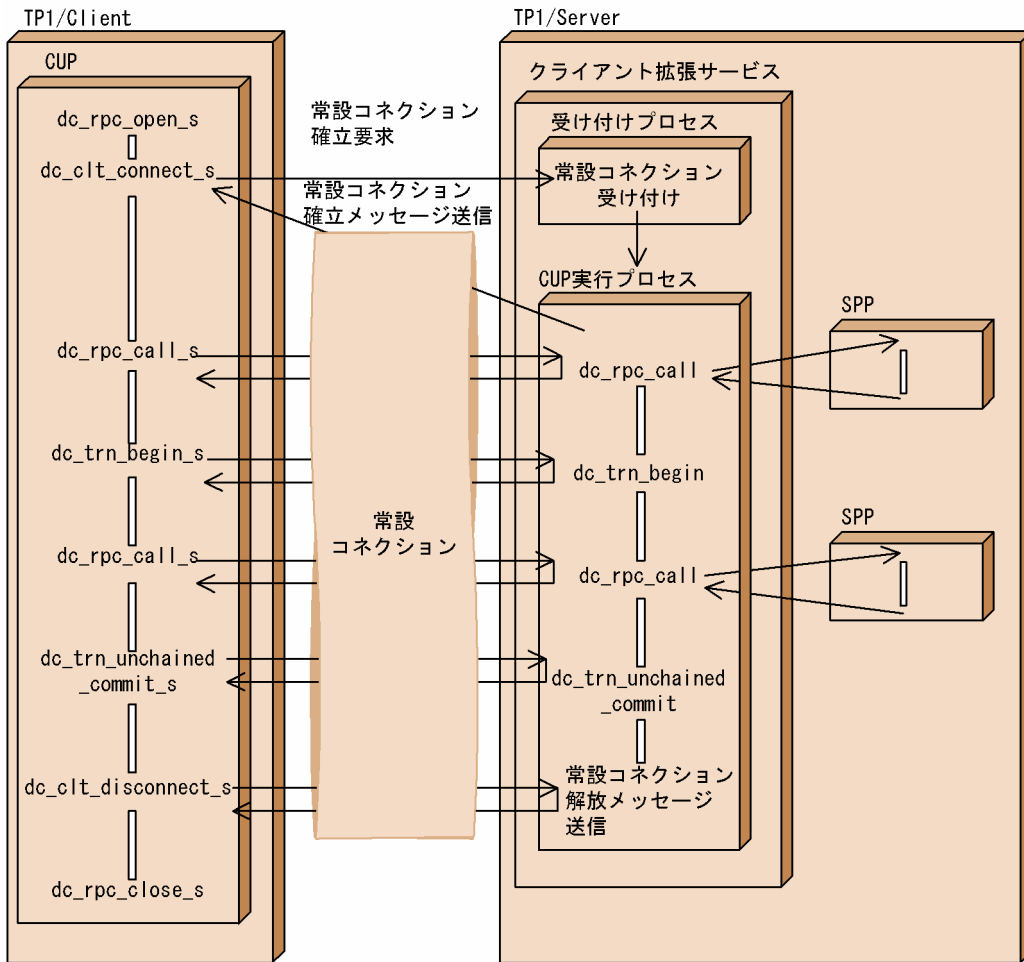
`dc_clt_cltin_s` 関数の引数 `target_host` に指定した OpenTP1 ノード、またはクライアント環境定義 DCHOST オペランドに指定した OpenTP1 ノードです。両方指定した場合、`dc_clt_cltin_s` 関数の引数 `target_host` に指定したノードが優先されます。

なお、ユーザ認証を抑止し、`dc_clt_cltin_s` 関数の引数 `target_host` に NULL を指定、かつクライアント環境定義 DCHOST オペランドの指定を省略した場合、常設コネクションの確立に失敗します。

`dc_clt_disconnect_s` 関数で常設コネクションを解放するまで、メッセージの送受信には常設コネクションを使用します。ただし、何らかの障害が発生した場合、常設コネクションは解放されます。

常設コネクションの確立・解放処理を次の図に示します。

図 2-1 常設コネクションの確立・解放処理



2.2.2 常設コネクションを使用する場合に関連する定義

常設コネクションを使用する場合、必要に応じて次の定義を設定してください。

- クライアント環境定義
 - DCCLTINQUIRETIME
 - DCCLTPORT
 - DCCLTDCCMHOST
 - DCCLTDCCMPORT
- クライアントサービス定義
 - clt_inquire_time
 - clt_port
 - clt_cup_conf
 - cup_parallel_count
 - cup_balance_count

2.2.3 DCCM3 論理端末に端末識別情報を通知する

常設コネクションを使用して DCCM3 論理端末と通信する場合、クライアント側 (TP1/Client) でユーザが指定した端末識別情報を DCCM3 論理端末に通知し、DCCM3 の端末固定割り当て機能を利用できます。

(1) DCCM3 論理端末でのメッセージ送受信方法

DCCM3 論理端末では、通信相手となる TP1/Client の CUP を IP アドレスと DCCM3 論理端末のポート番号で区別する論理端末として定義し、論理端末ごとにメッセージの送受信を行っています。

したがって、マルチスレッド型の CUP を使用する場合、どのスレッドからのサービス要求でも、CUP の IP アドレスと DCCM3 論理端末のポート番号の組み合わせが同じとなるため、該当する論理端末を複数定義した場合、CUP と DCCM3 論理端末との組み合わせを特定できません。そのため、サービス要求を受け付ける DCCM3 論理端末が異なると、DCCM3 側のサーバ処理の順番が保証されなくなるため、業務によっては問題となることがあります。

(2) 端末識別情報の通知

常設コネクションを使用して DCCM3 論理端末と通信する場合、TP1/Client でユーザが指定した端末識別情報を DCCM3 論理端末に通知することで、DCCM3 の端末固定割り当て機能を利用できます。端末識別情報には DCCM3 論理端末の論理端末名称を使用します。

端末識別情報を DCCM3 論理端末に通知することで、CUP が常に同じ論理端末に割り当てられるため、CUP と論理端末の組み合わせを特定できます。

端末識別情報を DCCM3 論理端末に通知するには、クライアント環境定義の DCCLTCONNECTINF に端末識別情報を指定するか、`dc_clt_set_connect_inf_s` 関数を実行して端末識別情報を設定します。

常設コネクションを使用して DCCM3 論理端末と通信する場合、次の二つの方法があります。このうち、DCCM3 論理端末に端末識別情報を通知できるのは、2.の方法だけです。

1. クライアント環境定義 DCCLTDCCMHOST に DCCM3 論理端末のホスト名、DCCLTDCCMPORT に DCCM3 論理端末のポート番号を指定し、`dc_clt_connect_s` 関数の引数 flags には DCCLT_DCCM3 を指定します。
2. クライアント環境定義 DCCLTRAPHOST に DCCM3 論理端末のホスト名およびポート番号を指定し、`dc_clt_connect_s` 関数の引数 flags には DCNOFLAGS を指定します。

(3) DCCM3 論理端末に端末識別情報を通知するときの注意事項

- 端末識別情報の通知によって、DCCM3 の端末固定割り当て機能を利用できるのは、DCCM3 のバージョン 09-03 以降です。端末固定割り当て機能については、マニュアル「VOS3 データマネジメントシステム XDM E2 系 解説」を参照してください。

- TP1/Client で指定した端末識別情報と一致する DCCM3 論理端末の論理端末名称が DCCM3 側で定義されていない場合、dc_clt_connect_s 関数は DCCLTER_NET_DOWN でエラーリターンします。

2.2.4 常設コネクションを使用するときの注意事項

- トランザクション内から常設コネクションを確立することはできません。常設コネクションを確立したあと、トランザクションを生成してください。ただし、常設コネクション確立要求先サーバに DCCM3 の論理端末を指定した場合、CUP からトランザクションの生成はできません。
- 常設コネクション確立機能使用時、クライアントで通信障害やタイムアウトを検知した場合、常設コネクションは解放されます。
- 常設コネクション確立中に、サーバ側で監視している各種タイマ値のどれかが満了となった場合、サーバがダウンする場合があります。サーバがダウンした場合、タイミングによっては、CUP から実行した関数が最大応答待ち時間分待ち状態となり、タイムアウトとなります。これは、クライアント側から送信するパケットをサーバ側が認識できないため、この現象が発生します。この現象を回避するために、各種タイマ値には適切な値を設定してください。

2.3 リモートプロシジャコール

CUP から SPP へのリモートプロシジャコール (RPC) について説明します。

RPC の詳細については、マニュアル「OpenTP1 プログラム作成の手引」を参照してください。

2.3.1 RPC の実現方法

CUP からサービスを要求する関数を実行して、SPP のサービスを要求します。サービスを要求するときには、SPP のサービスグループ名とサービス名を引数に設定した `dc_rpc_call_s` 関数を実行します。

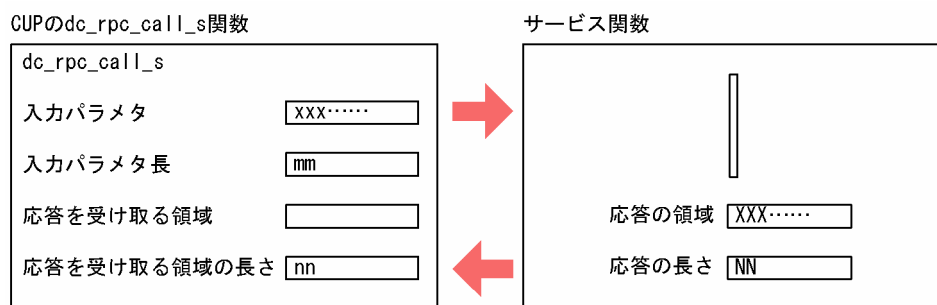
SPP は、TP1/Server のユーザサービス構成定義、または `dcsvstart` コマンドで起動しておきます。

2.3.2 RPC でのデータの受け渡し

RPC でのデータの受け渡しは、`dc_rpc_call_s` 関数に、SPP のサービスグループ名、サービス名、入力パラメタ、入力パラメタ長、サービスの応答格納領域、および応答長を指定して実行します。

RPC のデータの受け渡しを次の図に示します。

図 2-2 RPC のデータの受け渡し



注 サービス関数から返す応答の長さ (NNの値) は、`dc_rpc_call_s`関数で設定した値 (nnの値) と同じか、または小さい値となるようにしてください。

2.3.3 RPC の形態

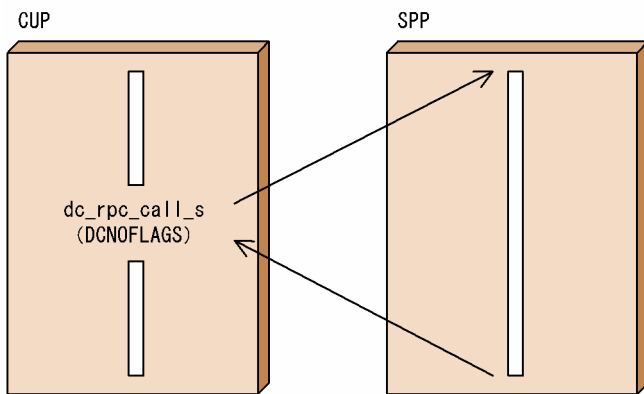
TP1/Client で実行できる RPC の形態は、同期応答型 RPC と非応答型 RPC です。

(1) 同期応答型 RPC

CUP から SPP へ問い合わせメッセージを送信し、応答メッセージを受信する形態です。SPP から処理結果が返ってくるまで、次の処理を待ちます。

同期応答型 RPC の処理の流れを次の図に示します。

図 2-3 同期応答型 RPC の処理の流れ

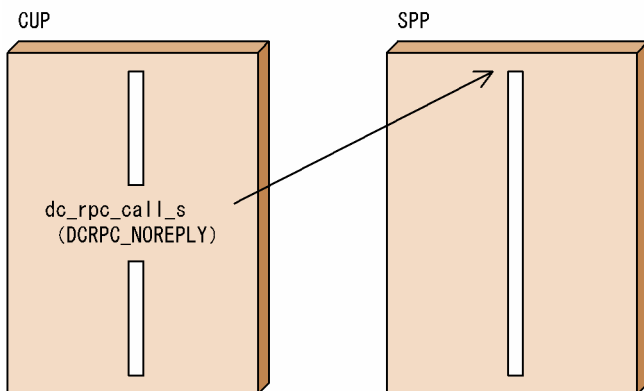


(2) 非応答型 RPC

CUP から SPP へ問い合わせメッセージを送信し、応答メッセージを受信しない形態です。SPP から処理結果を受け取らないで、次の処理を実行します。

非応答型 RPC の処理の流れを次の図に示します。

図 2-4 非応答型 RPC の処理の流れ



2.3.4 連鎖 RPC

SPP の実行プロセスは、マルチサーバ（同じ SPP を複数のプロセスで同時に起動する機能）の場合、サービスが要求されるたびに起動されます。CUP から同じサービスグループを 2 回以上呼び出したとき、そのサービスグループの SPP が以前と同じプロセスで実行されるとは限りません。ただし、同期応答型 RPC で、かつ同じサービスグループに属するサービスを 2 回以上要求する場合に限り、そのサービスを以前と同じプロセスで実行させることができます。これを**連鎖 RPC**といいます。

連鎖 RPC は、CUP からトランザクションを起動している場合、または常設コネクションを確立している場合に使用できます。

連鎖 RPC を使用すると、一つのトランザクション処理に必要なユーザプロセスの数が少なくなり、トランザクション処理に掛かる負荷を軽減できます。トランザクションとして連鎖 RPC を使用する場合は、一つのグローバルトランザクションで動作します。

連鎖 RPC は、CUP のプロセス単位に保証されます。ただし、同じグローバルトランザクション内でも、クライアント UAP が異なれば、複数回呼び出されたサービスが同じプロセスで起動されることは保証されません。

(1) 連鎖 RPC の開始

連鎖 RPC となるサービス要求をする場合は、サービスを要求する `dc_rpc_call_s` 関数の flags に `DCRPC_CHAINED` を指定してください。この値を指定してサービスを要求すると、SPP は連鎖 RPC であることを認識して、プロセスを確保します。2 回目以降のサービス要求の flags にも `DCRPC_CHAINED` を指定します。

(2) 連鎖 RPC の終了

連鎖 RPC は、次のどちらかの方法で終了します。

- 連鎖 RPC を実行しているサービスグループに対して、flags に `DCNOFLAGS` を指定した `dc_rpc_call_s` 関数を実行する。
- 連鎖 RPC を実行しているグローバルトランザクションを同期点処理（コミット、またはロールバック）で完了させる。

(3) 連鎖 RPC の時間監視

連鎖 RPC でサービスを要求される UAP では、CUP に応答を返してから、次のサービス要求がくるまでの間、またはトランザクションの同期点処理がくるまでの時間を監視しています。この監視時間を過ぎても次のサービス要求、または同期点処理要求が来ない場合は、CUP で障害が発生したものとみなして、SPP を異常終了させます。監視時間はユーザサービス定義の `watch_next_chain_time` で指定します。

2.3.5 スケジュール機能

TP1/Server のスケジュール機能は、CUP からの SPP に対するサービス要求でも有効です。TP1/Server は、SPP のサービスグループごとにスケジュールキューを作成し、サービス要求をスケジュールします。ただし、SPP をソケット受信型サーバと指定（TP1/Server のユーザサービス定義で `receive_from=socket` と指定）しておく、スケジュールキューを経由しないで、SPP が直接 CUP からのサービス要求を受信できます。

またマルチノードサーバの場合（同じサービスグループを処理するサーバが複数のノードで起動されている場合）、ノード間負荷バランス機能を使用して、クライアントからのサービス要求を各ノードへ振り分けて負荷を分散できます。ただし、振り分けをするためのオーバーヘッドが掛かり、サーバの情報を格納して

おくためのメモリが必要となります。このため、サーバが唯一であることが明らかな場合には、振り分けをしない方が性能がよい場合もあります。

ノード間負荷バランス機能を使うかどうかは、クライアント環境定義 DCCLTLOADBALANCE で指定します。

サーバの情報を格納しておくためのメモリサイズは、クライアント環境定義 DCCACHE で指定します。このメモリを大きく指定しておくこと、より多くのサーバ情報を格納できるため、多くのサーバにサービス要求を分散させることができます。負荷を分散させる機能を使わない場合でも、サービス情報を検索する頻度が少なくなるため、性能上効果があります。

負荷を分散させる機能を使用する場合、メモリ中に格納されたサーバ情報の有効時間を指定できます。有効時間を短くしておくこと、短い単位でサーバの負荷情報を検索するので、最新の負荷状態に従って振り分けすることができます。その反面、負荷情報の検索にオーバヘッドが掛かるため、逆に性能劣化となる場合もあります。有効時間は、クライアント環境定義 DCCLTCACHETIM で指定します。

2.3.6 ノード間負荷バランス機能

OpenTP1 では、RPC による要求が特定のノードに集中しないようにノード間で負荷を分散する機能があります。これをノード間負荷バランス機能といいます。

ノード間負荷バランス機能を使用するためには、負荷分散の前提として次の条件を満たしている必要があります。

- 複数のノードに同じサービスを提供するユーザサーバが起動されていること。
- 各 OpenTP1 ノードはシステム共通定義の all_node オペランドに自分以外のノードを定義することによって、お互いの OpenTP1 ノードで起動されているユーザサーバの情報（ネーム情報）をやり取りしていること。

ここでは、OpenTP1 のノード間負荷バランス機能を使用する場合のクライアント側、サーバ側の関連する定義と処理、および RPC の処理の内容を説明します。

(1) サーバ側の判断で負荷分散を行う場合

TP1/Server のスケジュールサービスが、ノードのスケジュール状態に応じて、より効率的に処理できるノードへ負荷を分散させます。

(a) TP1/Client 側の定義

クライアント環境定義に次のオペランドを指定または省略する必要があります。

- DCCLTLOADBALANCE=N (デフォルト)

(b) TP1/Server 側の定義

TP1/Server 側の定義では、次の設定をする必要があります。

- スケジュールサービス定義に次のオペランドを指定または省略する。
scd_this_node_first=N (デフォルト)
scd_announce_server_status=Y (デフォルト)

(2) サーバからの負荷情報からクライアント側で判断する場合

サーバから得たサーバの負荷レベルを基に、サービス要求を行うクライアントが OpenTP1 ノードを決めて RPC を実行します。

(a) TP1/Client 側の定義

クライアント環境定義に次の設定をする必要があります。

- 次のオペランドを指定または省略する。
DCSCDDIRECT=N (デフォルト)
- 次のオペランドを指定する。
DCCLTLOADBALANCE=Y

(b) TP1/Server 側の定義

TP1/Server 側の定義では、次の設定をする必要があります。

- スケジュールサービス定義に次のオペランドを指定または省略する。
scd_announce_server_status=Y (デフォルト)

(3) RPC 種別による動作の違い

(a) リモート API 機能を組み合わせた場合

サーバ側の判断で負荷分散を行います。

(b) スケジューラダイレクト機能を組み合わせた場合

スケジューラダイレクト機能を使用した RPC では、窓口となる TP1/Server が複数ある場合、DCHOST オペランドに指定された順番にスケジュールを依頼します。スケジュールを依頼する TP1/Server は障害が発生するまで切り替えません。

最初の RPC でのサービス要求時に、サービス要求先スケジューラをランダムに選択するには、クライアント環境定義に DCHOSTSELECT=Y を指定します。詳細については、「[2.3.12 窓口となる TP1/Server の負荷分散](#)」を参照してください。

(c) ネームサービスを使用した RPC を組み合わせた場合

ネームサービスを使用した RPC では、窓口となる TP1/Server のネームサービスにサービス情報を問い合わせ、ネームサービスからの応答メッセージを基にサービス要求先スケジューラを決定します。

RPC ごとにサービス要求先スケジューラを分散させる場合は、クライアント環境定義に DCCLTLOADBALANCE=Y を指定します。これによって、TP1/Client は、ネームサーバから複数のサービス要求先スケジューラの情報を取得し、負荷レベルが最も低いサービス要求先スケジューラ情報をキャッシュに格納します。

キャッシュに格納されるサービス要求先スケジューラ情報は、1 つだけではなく、複数の場合もあります。

キャッシュに格納されたサービス要求先スケジューラ情報が複数ある場合、最初のサービス要求先スケジューラは、ランダムに選択されます。RPC でのサービス要求が 2 度目以降の場合は、サービス要求先スケジューラ情報取得のためにキャッシュを参照し、RPC ごとにラウンドロビン方式でサービス要求先スケジューラを切り替え、分散させます。

クライアントからの RPC 実行時に、このキャッシュ領域中に該当するサービス情報が存在する場合には、窓口となる TP1/Server のネームサービスに対してサービス情報の問い合わせを行いません。クライアントでは LRU (Least Recently Used) 方式でキャッシュを管理しているため、キャッシュ領域が不足した場合には参照されていないサービス情報から順に削除します。また、クライアント環境定義の DCCLTCACHETIM オペランドに指定した有効時間が過ぎたサービス情報は、RPC 実行時にキャッシュ領域から削除され、ネームサービスに対してサービス情報を問い合わせます。

サーバが 129 以上のマルチノードサーバ構成で、窓口となる TP1/Server でネームサービス定義に [nam_service_extend=1] を指定している場合、クライアント環境定義に DCCLTNAMEXTEND=1 を指定してください。この定義の指定で、クライアントがネームサービスから一度に取得できるサービス情報の最大数を 128 個から 512 個に拡張できます。

注意事項

- クライアント環境定義の DCCACHE オペランドの指定値を大きくすると、多くのサービス情報を格納でき、窓口となる TP1/Server のネームサービスとの通信回数を削減できます。ただし、多くのキャッシュ領域中からサービス情報を検索するのでオーバーヘッドが掛かります。
- クライアント環境定義の DCCACHE オペランドの指定値を小さくすると、キャッシュ領域に該当する SPP の各ノードのサービス情報が入りきらないことがあります。その場合は、クライアントから再度 RPC を実行しても、キャッシュ領域に入りきらなかったノードのスケジュールサービスに対しては RPC 要求を実行しません。
- クライアント環境定義の DCCLTCACHETIM オペランドの指定値を小さくすると、古いサービス情報は直ちに削除され、窓口となる TP1/Server のネームサービスに新しいサービス情報を問い合わせます。この場合、常に最新のサービス情報をキャッシュ領域に保持できるため、サーバの負荷に応じて RPC 要求を振り分けられます。ただし、ネームサービスとの通信回数が増え、また、キャッシュ領域の書き換え処理にもオーバーヘッドが掛かります。

- クライアント環境定義の DCCLTCACHETIM オペランドの指定値を大きくすると、窓口となる TP1/Server のネームサービスとの通信回数を削減できます。ただし、SPP の状態変化への対応が遅れるため、SPP が停止したノードのスケジューラに対して RPC 要求を実行してしまうことがあります。この場合は、クライアントが別のノードに対して RPC 要求を実行する前に、キャッシュ領域中の該当するサービス情報を削除し、窓口となる TP1/Server のネームサービスに対して、該当するサービス情報の削除要求をします。

(d) 通信先を指定した RPC を組み合わせた場合の動作

通信先を指定した遠隔サービスの要求の場合、クライアント環境定義の DCCLTONLYTHISNODE オペランドの指定値が N、または指定値を省略したときは、サービス要求を受け付けたスケジューラサービスのノード優先で負荷分散を行います。クライアント環境定義の DCCLTONLYTHISNODE オペランドの指定値が Y のときは、負荷分散を行いません。

(e) サーバ側のクライアント拡張サービスを組み合わせた場合

サーバ側の判断で負荷分散を行います。

2.3.7 RPC の時間監視

同期応答型 RPC を行う場合、応答メッセージを受信するまでの時間を監視できます。

監視時間は、クライアント環境定義 DCWATCHTIM に指定します。

また、CUP から `dc_rpc_set_watch_time_s` 関数を実行して、監視時間を設定することもできます。要求するサービスに応じて、監視時間を変更したい場合に、RPC を行う前に設定します。

`dc_rpc_get_watch_time_s` 関数を実行すると、監視時間の設定値を参照できます。

監視時間を過ぎても応答メッセージが返らない場合には、RPC がエラーリターンします。

2.3.8 OpenTP1 以外のサーバへの RPC

DCCM3 などの、OpenTP1 以外のサーバへ RPC を行う機能です。なおサーバ側に、OpenTP1 の RPC 要求を解釈する機能が組み込まれていることが前提になります。

(1) 相手サーバの指定方法

RPC を行う際、相手サーバはサービスグループ名とサービス名で指定しますが、この指定方式は OpenTP1 のサーバに対する RPC と同じです。

ただし、OpenTP1 のネームサービスの管理外にあるサーバを呼び出すため、クライアント側にネームサービスに代わるアドレス解決手段を用意します。

(2) 相手サーバのアドレス定義

クライアント側に、サービスグループ名に対応する、RPC 受け付け窓口（ホスト名、ポート番号）のリストを、テキストファイルで作成しておきます。クライアント環境定義 DCCLTSERVICEGROUPLIST にこのテキストファイル名を宣言します。

TP1/Client は RPC 実行時、指定されたサービスグループ名が、このリストに定義されているか検索し、合致するものがあつた場合には対応する RPC 受け付け窓口に RPC を行います。

(3) RPC 機能の概要

RPC の形態としては、同期応答型 RPC と非応答型 RPC です。

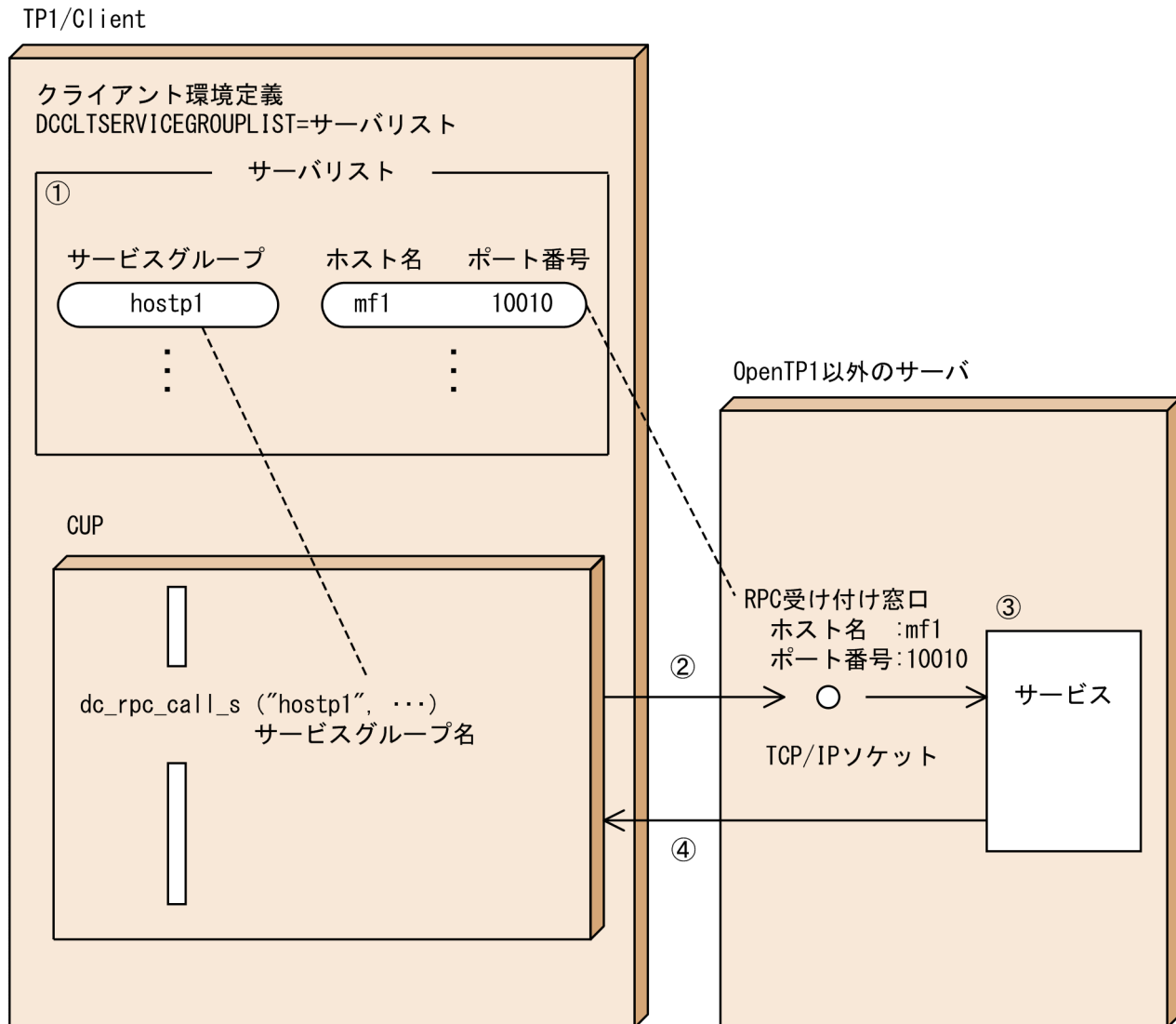
トランザクション制御下で RPC を行っても、OpenTP1 のトランザクションの対象とはなりません。

サービスグループ名とサーバが対になっている関係上、負荷分散機能も対象外となります。ただし、常設コネクションを使用して DCCM3 論理端末へ RPC を行う場合、負荷分散機能を使用できます。詳細については、「[2.3.8\(4\) DCCM3 論理端末へ RPC を行う場合の負荷分散](#)」を参照してください。

DCCM3 に対して RPC を行う場合、サービス名がトランザクション名と評価されます。

OpenTP1 以外のサーバへの RPC の処理の流れを次の図に示します。

図 2-5 OpenTP1 以外のサーバへの RPC の処理の流れ



(説明)

- ① クライアント環境定義でサービスグループ名と、RPC受け付け窓口との対応関係を定義する。
- ② RPCを行う際、サービスグループ名から、RPC受け付け窓口のアドレスを求め、RPCのメッセージを送信する。
- ③ RPCのメッセージを解釈し、要求されたサービスを実行する。
- ④ 同期応答型RPCの場合、サーバからの応答メッセージを受信する。

(4) DCCM3 論理端末へ RPC を行う場合の負荷分散

TP1/Client と DCCM3 論理端末が常設コネクションを使用して RPC を行う場合、複数の DCCM3 に振り分けて負荷を分散できます。クライアント環境定義に指定された複数の DCCM3 論理端末のホスト名およびポート番号の中から、接続先をランダムに選択し、接続を試みます。試みた先の DCCM3 論理端末との接続に失敗した場合は、これを除き、クライアント環境定義に指定した残りの DCCM3 論理端末から再びランダムに選択し、接続を試みます。これを繰り返す、定義に指定された DCCM3 論理端末との接続にすべて失敗した場合、初めてエラーを検知します。

TP1/Client と DCCM3 論理端末との通信方法を次に示します。この中で、負荷分散できるのは、常設コネクションを使用している 1.と 2.の方法です。

1. クライアント環境定義 DCCLTDCCMHOST に DCCM3 論理端末のホスト名、DCCLTDCCMPORT に DCCM3 論理端末のポート番号を指定し、dc_clt_connect_s 関数の引数 flags に DCCLT_DCCM3 を指定します。

この場合、常設コネクションを使用します。

2. クライアント環境定義 DCCLTRAPHOST に DCCM3 論理端末のホスト名およびポート番号を指定し、dc_clt_connect_s 関数の引数 flags に DCNOFLAGS を指定します。

この場合、常設コネクションを使用します。

3. クライアント環境定義 DCCLTSERVICEGROUPLIST に、サービスグループごとに DCCM3 論理端末のホスト名およびポート番号を指定したファイルを指定します。

この場合、常設コネクションを使用しません。

2.3.9 ネームサービスを使用した RPC

(1) ネームサービスを使用した RPC

TP1/Client から RPC をする際には、TP1/Server のネームサービスの機能を使用しています。ネームサービスの機能を使用することで、OpenTP1 システムのサービス情報の管理を行っています。

TP1/Client はサービス情報のキャッシュを持っているため、TP1/Server のネームサービスへの問い合わせを毎回行いません。しかし、キャッシュにサービス情報が登録されていないとき、またはキャッシュの有効期限が切れたときは、TP1/Server のネームサービスに対して通信が発生します。

(2) ネームサービスを使用しない RPC

TP1/Client とサーバ間の通信回数を少しでも削減したい場合は、TP1/Server のネームサービスの機能を抑止できます。

この場合、RPC 実行時にネームサービスに問い合わせないため、TP1/Client とサーバ間の通信負荷を削減できます。TP1/Client とサーバが WAN 経由して接続されている場合などに有効です。

ネームサービスを使用しない RPC を行うかどうかは、クライアント環境定義 DCSCDDIRECT で指定します。

この機能を使用する場合、TP1/Server のスケジュールサービスに対して直接サービスの要求を行います。

なお、この機能の使用時は、ソケット受信型 SPP に対して RPC の発行はできません。

(3) 窓口となる TP1/Server

ネームサービスを使用した RPC、およびネームサービスを使用しない RPC の窓口となる TP1/Server は、ユーザ認証の有無によって次のようになります。ユーザ認証機能については、「2.1 ユーザ認証機能」を参照してください。

(a) ユーザ認証を行う場合

ユーザ認証に成功した TP1/Server です。

(b) ユーザ認証を抑止する場合

dc_clt_cltin_s 関数の引数 target_host に指定した OpenTP1 ノード、またはクライアント環境定義 DCHOST に指定した TP1/Server です。両方指定した場合、dc_clt_cltin_s 関数の引数 target_host に指定した TP1/Server が優先されます。

なお、ユーザ認証を抑止し、dc_clt_cltin_s 関数の引数 target_host に NULL を指定、かつクライアント環境定義 DCHOST の指定を省略した場合、ネームサービスを使用した RPC、およびネームサービスを使用しない RPC を使用できません。

2.3.10 マルチスケジューラ機能を使用した RPC

CUP からスケジューラキューを使う SPP（キュー受信型サーバ）にサービスを要求した場合、要求先 SPP があるノードのスケジューラデーモンが、いったんサービス要求メッセージを受信し、該当する SPP のスケジューラキューに格納します。スケジューラデーモンとは、スケジューラサービスを提供するシステムデーモンのことです。

長大なサービス要求メッセージは、一定の長さに分割してスケジューラデーモンに送信します。スケジューラデーモンは、サービス要求メッセージを組み立ててキュー受信型サーバのスケジューラキューに格納します。スケジューラデーモンは、OpenTP1 システムごとに 1 プロセスです。そのため、分割されたサービス要求メッセージの受信処理が完了するまで、スケジューラデーモンはほかのサービス要求メッセージを受信できません。通信速度が遅い回線を使用して、長大なサービス要求メッセージを送信した場合、ほかのサービス要求のスケジューリングが遅延することがあります。また、システムの大規模化、マシンやネットワークの高性能化などに伴って、効率良くスケジューリングできないことがあります。この場合、従来のスケジューラデーモンとは別に、サービス要求受信専用デーモンを複数プロセス起動し、サービス要求メッセージ受信処理を並行動作させることによって、スケジューリング遅延を回避できます。この機能をマルチスケジューラ機能といいます。以降、従来のスケジューラデーモンをマスタスケジューラデーモン、サービス要求専門デーモンをマルチスケジューラデーモンと呼びます。

マルチスケジューラ機能の検討が必要なシステム構成については、マニュアル「OpenTP1 プログラム作成の手引」を参照してください。

(1) マルチスケジューラデーモンをランダムに選択する

マルチスケジューラ機能を使用することで、複数起動されているマルチスケジューラデーモンの中から、利用できるマルチスケジューラデーモンをランダムに選択してサービス要求を送信できます。

マルチスケジューラデーモンをランダムに選択して、ネームサービスを使用しない RPC、または通常の RPC を実行できます。

(a) ネームサービスを使用しない RPC

クライアント環境定義 DCSCDDIRECT に Y を指定してネームサービスを使用しない RPC を行う場合、クライアント環境定義 DCSCDPORT を指定しているかどうかで次のような違いがあります。

• DCSCDPORT を指定する場合

DCSCDPORT を指定した場合、窓口となる TP1/Server のネームサービスに問い合わせることなく、サービス要求を送信するスケジューラデーモンをランダムに選択できるため、通信回数が削減され、また、ネームサービスの負荷軽減にもつながります。

サービス要求を送信するスケジューラデーモンのポート番号は、次に示す範囲からランダムに選択します。

- 下限値：DCSCDPORT に指定したポート番号
- 上限値：下限値 + DCSCDMULTICOUNT に指定したプロセス数 - 1

DCSCDPORT には、マルチスケジューラデーモンのポート番号を指定します。マスタスケジューラデーモンとマルチスケジューラデーモンのベースとなるポート番号が連続している場合は、マスタスケジューラデーモンのポート番号も指定できます。

DCSCDMULTICOUNT には、TP1/Server で起動しているスケジューラデーモンのプロセス数を指定します。DCSCDMULTICOUNT に指定するプロセス数は、DCSCDPORT に指定するポート番号の内容によって、次のとおり異なります。

DCSCDPORT 指定値	DCSCDMULTICOUNT 指定値	
マルチスケジューラデーモンのベースとなるポート番号	マルチスケジューラデーモンのプロセス数と同値か、それ以下の値を指定する。	
マルチスケジューラデーモンの任意のポート番号	「マルチスケジューラデーモンのポート番号の最大 - DCSCDPORT に指定したポート番号 + 1」の値か、それ以下の値を指定する。	
マスタスケジューラデーモンのポート番号	マスタスケジューラデーモンとマルチスケジューラデーモンのベースとなるポート番号が連続している場合	「マルチスケジューラデーモンのプロセス数 + 1」の値か、それ以下の値を指定する。
	上記以外	1 を指定するか、または指定を省略する (マルチスケジューラデーモンは使用できない)。

なお、DCHOST で指定した窓口となる TP1/Server 間で、スケジュールサービス定義の scdmulti で指定した値を統一する必要があります。

- DCSCDPORT を指定しない場合

クライアントユーザの認証を要求したあと、最初のサービス要求時に窓口となる TP1/Server のネームサービスへ問い合わせ、サービス情報を得ます。この情報を基に、マルチスケジューラデーモンをランダムに選択してサービス要求を送信します。サービス情報は次のどちらかの時点まで有効です。

- クライアントユーザの認証を解除 (dc_clt_cltout_s 関数の実行)
- 窓口となる TP1/Server の切り替え

サービス要求を送信するマルチスケジューラデーモンのポート番号の範囲は、スケジュールサービス定義の scdmulti の最初に定義した、-p オプションに指定したポート番号を基にして決めます。そのため、スケジュールサービス定義の指定順序を考慮する必要があります。

(b) 通常の RPC

マルチスケジューラ機能を使用して、通常の RPC (ネームサービスを使用する) を行う場合について説明します。サービス情報を一時的に格納する領域に該当するサービス情報がない場合にネームサービスにサービス情報を問い合わせます。サービス情報を基にマルチスケジューラデーモンをランダムに選択してサービス要求を送信します。

(2) クライアント環境定義とサービス要求を送信するスケジューラデーモンの関連

マルチスケジューラ機能を使用した場合、サービス要求を送信するスケジューラデーモンはクライアント環境定義の指定によって異なります。

クライアント環境定義のオペランドの指定とスケジューラデーモンの関連を次の表に示します。

表 2-1 クライアント環境定義のオペランドの指定とスケジューラデーモンの関連

クライアント環境定義のオペランドの指定				サービス要求を送信するスケジューラデーモン
DCSCDMULTI	DCSCDDIRECT	DCSCDPORT	DCSCDMULTICOUNT	
Y	Y	○	2 以上の値	ランダムに選択したスケジューラデーモン※1
			1, または指定しない	DCSCDPORT に指定したポート番号で起動されているスケジューラデーモン
		—	×	スケジュールサービス定義 scdmulti の最初に定義した、-p オプションに指定したポート番号を基にして決めた、マルチスケジューラデーモンからランダムに選択したマルチスケジューラデーモン※2

クライアント環境定義のオペランドの指定				サービス要求を送信するスケジューラデーモン
DCSCDMULTI	DCSCDDIRECT	DCSCDPORT	DCSCDMULTICOUNT	
Y	N		×	サービス情報を基にランダムに選択したマルチスケジューラデーモン※2
N	Y	○	×	DCSCDPORT に指定したポート番号で起動されているスケジューラデーモン
		—	×	マスタスケジューラデーモン※2
	N		×	ソケット受信型サーバ、またはマスタスケジューラデーモン※2

(凡例)

- Y：オペランドの指定値が Y である
- N：オペランドの指定値が N である
- ：オペランドに値を指定する
- ×：指定値は無効である
- ：オペランドに値を指定しない

注※1

スケジューラデーモンのポート番号は、次に示す範囲の値から選択します。

下限値：DCSCDPORTに指定したポート番号

上限値：下限値 + DCSCDMULTICOUNTに指定したプロセス数 - 1

注※2

ネームサービスへの問い合わせが発生します。

2.3.11 窓口となる TP1/Server の切り替え機能

窓口となる TP1/Server で障害が発生すると、エラーリターン後、窓口となる TP1/Server の定義を変更する必要があります。しかし、複数の窓口となる TP1/Server を指定しておくこと、TP1/Client は、指定された次の TP1/Server に切り替えを試みます。

ホスト切り替えは、通信が成功するまで繰り返し行います。一巡し、指定されたすべてのホストとの接続に失敗した場合、関数はエラーリターンします。

ホスト切り替えに成功し関数が正常終了したあと、再度ホスト切り替えが発生した場合、前回障害となったホストも切り替えるホストの対象となります。

ホスト切り替えが発生すると、dc_clt_cltout_s 関数を実行するまで、そのホストが通信先ホストとなります。

ただし、CUP から一斉にサービスを要求した場合、一つの窓口となる TP1/Server のネームサービスに問い合わせが集中してしまうので、負荷が集中してしまうことになります。複数の窓口となる TP1/Server をランダムに選択すれば、窓口となる TP1/Server の負荷を分散できます。詳細については「[2.3.12 窓口となる TP1/Server の負荷分散](#)」を参照してください。

(1) 切り替えのタイミング

- ユーザ認証時に TP1/Server のネームサービスとの接続に失敗した場合、時間内に応答が返ってこなかった場合、またはエラー応答を受信した場合。
- RPC 発行時に窓口となる TP1/Server のネームサービスとの接続に失敗した場合、時間内に応答が返ってこなかった場合、またはエラー応答を受信した場合。
- ネームサービスを使用しない RPC 発行時に窓口となる TP1/Server のスケジュールサービスとの通信が失敗した場合。
- トランザクションの開始要求時に窓口となる TP1/Server のネームサービスとの接続に失敗した場合、時間内に応答が返ってこなかった場合、またはエラー応答を受信した場合。
- 常設コネクション確立要求時に窓口となる TP1/Server のネームサービスとの接続に失敗した場合、時間内に応答が返ってこなかった場合、またはエラー応答を受信した場合。
- サービス要求時、スケジュールサービス開始処理中、およびスケジュールサービス終了処理中に、窓口となる TP1/Server からエラー応答を受信した場合。

(2) 切り替え先 OpenTP1 の指定方法

- `dc_clt_cltin_s` 関数の引数 `target_host`、またはクライアント環境定義 `DCHOST` に、複数の窓口となる TP1/Server を指定します。クライアント環境定義 `DCHOST` の指定は、`dc_clt_cltin_s` 関数の引数 `target_host` に `NULL` を指定した場合に有効となります。
- サービス要求時、スケジュールサービス開始処理中、およびスケジュールサービス終了処理中に、窓口となる TP1/Server からエラー応答を受信した場合、上記の指定に加え、クライアント環境定義 `DCHOSTCHANGE` に `Y` を指定します。

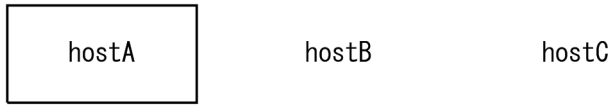
(3) 切り替えの順序

初回選択時 (`dc_clt_cltin_s` 関数実行時) は、先頭に指定されたホストを選択します。障害発生時は、障害となったホストの次に指定されたホストを選択します。

〈例〉 次の指定の場合

```
DCHOST=hostA, hostB, hostC  
DCHOSTSELECT=N
```

1. `dc_clt_cltin_s` 関数発行
先頭に指定された `hostA` を選択

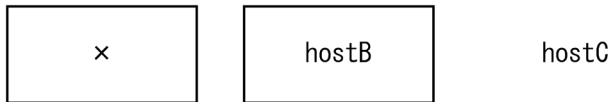


hostA との接続に成功

2. dc_rpc_call_s 関数発行

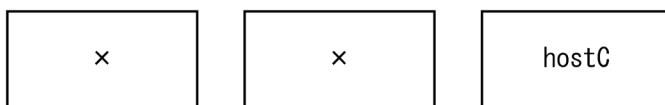
hostA との接続障害発生

障害となった hostA の次に指定された hostB を選択



hostB との接続障害発生

障害となった hostB の次に指定された hostC を選択



hostC との接続に成功

3. dc_rpc_call_s 関数発行

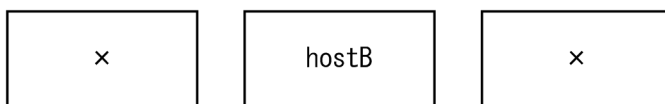
hostC との接続障害発生

障害となった hostC の次に指定された hostA を選択



hostA との接続障害発生

障害となった hostA の次に指定された hostB を選択



hostB との接続障害発生

選択対象ホストがなくなり関数はエラーリターン

4. dc_rpc_call_s 関数発行

hostB との接続に成功

5. dc_clt_cltout_s 関数発行

6. dc_clt_cltin_s 関数発行

先頭に指定された hostA を選択

2.3.12 窓口となる TP1/Server の負荷分散

CUP から一斉にサービスを要求した場合、一つの窓口となる TP1/Server のネームサービスに問い合わせが集中してしまうので、負荷が集中してしまうことになります。しかし、TP1/Client が複数の窓口となる TP1/Server の中からランダムに選択すれば、窓口となる TP1/Server への負荷を分散できます。

(1) 窓口となる TP1/Server をランダムに選択する

窓口となる TP1/Server をランダムに選択するには、クライアント環境定義 DCHOSTSELECT に Y を指定します。

CUP からサービスを要求する場合、dc_clt_cltin_s 関数の引数 target_host、またはクライアント環境定義 DCHOST に指定された窓口となる TP1/Server のネームサービスに、要求するサービスがどのノードにあるかを問い合わせます。

複数の窓口となる TP1/Server を指定している場合、複数指定されたホストの中からランダムに選択します。障害発生時は、障害となったホストを除外した残りのホストの中からランダムに選択します。

〈例〉 次の指定の場合

```
DCHOST=hostA,hostB,hostC
DCHOSTSELECT=Y
```

1. dc_clt_cltin_s 関数発行

hostA, hostB, hostC からランダム選択

hostA hostB hostC

hostB との接続に成功

2. dc_rpc_call_s 関数発行

hostB との接続障害発生

障害となった hostB を除いてランダム選択

hostA × hostC

hostA との接続障害発生

hostC を選択

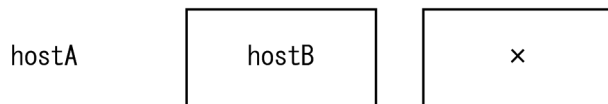
× × hostC

hostC との接続に成功

3. dc_rpc_call_s 関数発行

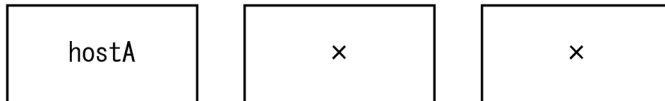
hostC との接続障害発生

障害となった hostC を除いてランダム選択



hostB との接続障害発生

hostA を選択



hostA との接続障害発生

選択対象ホストがなくなり関数はエラーリターン

4. dc_rpc_call_s 関数発行

hostA との接続に成功

5. dc_clt_cltout_s 関数発行

6. dc_clt_cltin_s 関数発行

hostA, hostB, hostC からランダム選択

(2) 窓口となる TP1/Server を優先して負荷分散する

TP1/Server では、サービス要求を各ノードに振り分けてノード間の負荷を分散しています。これによって、サービス要求を受け付けたノードのスケジュールサービスは、負荷を分散させるために、別のノードにある TP1/Server にサービス要求を転送する場合があります。

しかし、TP1/Client 側で窓口となる TP1/Server をランダムに選択していた場合は、すでに選択された窓口となる TP1/Server から、さらに別のノードにある TP1/Server へ振り分けられることとなります。このため、オーバーヘッドが掛かることがあります。

このような現象を回避するために、TP1/Client 側で選択した窓口となる TP1/Server を優先して負荷を分散できます。窓口となる TP1/Server を優先して負荷を分散するには、クライアント環境定義 DCSCDLOADPRIORITY に Y を指定します。

この定義は、ネームサービスを使用しない RPC を行う場合（クライアント環境定義 DCSCDDIRECT に Y を指定）だけ、有効です。

注意事項

窓口となる TP1/Server を優先して負荷分散した場合、あるノードの TP1/Server が障害によって停止すると、その TP1/Server を窓口としていた TP1/Client は、別の窓口となる TP1/Server に切り替えます。その後、停止した TP1/Server が再起動されても、TP1/Client は現時点で窓口となっている TP1/Server を優先します。そのため、再起動した TP1/Server は、負荷が低い状態であっても、この TP1/Client からのサービス要求が入りにくくなります。

なお、窓口としている TP1/Server を変えるには、クライアントユーザの認証解除 (dc_clt_cltout 関数または dc_clt_cltout_s 関数) およびクライアントユーザの認証要求 (dc_clt_cltin_s 関数) の再実行が必要です。これによって、再起動した TP1/Server が窓口となる TP1/Server に割り当てられることがあります。

2.3.13 データ圧縮機能

データ圧縮機能を使用すると、RPC によってネットワーク上に送り出されるユーザデータを圧縮できます。これによってネットワーク上に送り出されるパケット数を削減し、ネットワークの混雑を緩和できます。

データ圧縮機能を使用するかどうかは、クライアント環境定義 DCCLTDATACOMP で指定します。

この機能を使用すると、サービス要求データ (CUP から実行する dc_rpc_call_s 関数または dc_rpc_call_to_s 関数に設定する入力パラメタ (in) の値) に圧縮効果がある場合、TP1/Client は圧縮してネットワーク上に送り出します。そのサービス要求データに対し、SPP から返される応答データも TP1/Server で圧縮されてネットワーク上に送り出されてきます。その応答を受け取った TP1/Client は、圧縮データを復元して CUP に渡します。

この機能は、サービス要求先の TP1/Server Base が 03-03 以降の場合に使用できます。ただし、サービス要求先の TP1/Server Base のバージョンによって、次の点が異なります。

TP1/Server Base が 03-05 以前の場合

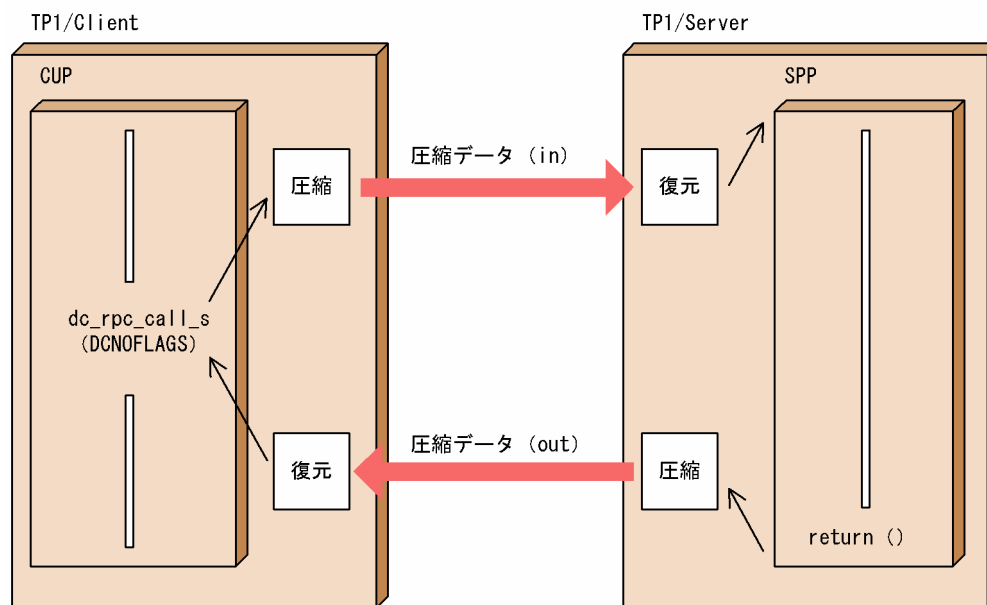
サービス要求データに圧縮効果がないために圧縮されなかったときは、応答データは圧縮効果があっても圧縮されません。

TP1/Server Base が 03-06 以降の場合

入力パラメタの値が圧縮されていなかった場合でも、返される応答の値に圧縮効果があれば、応答の値は TP1/Server Base で圧縮されます。

データ圧縮機能の概要を次の図に示します。

図 2-6 データ圧縮機能の概要



注 TP1/Server Baseのバージョンが03-06以降の場合、inが圧縮されていなくても、outに圧縮効果があればTP1/Server Baseで圧縮されます。

(1) データ圧縮機能の効果

データ圧縮機能の効果は、ユーザデータの内容に依存します。ユーザデータ中に連続した同じ文字が多く現れる場合には効果がありますが、ユーザデータの内容によっては、ほとんど効果がない場合もあります。

同じCUPからdc_rpc_call_s関数、またはdc_rpc_call_to_s関数を複数実行する場合は、CUP単位でデータ圧縮機能の効果を検討してください。

また、データ圧縮/復元のためのオーバーヘッドが掛かるため、データ圧縮による効果とのバランスを考慮し、事前に性能評価をしてからデータ圧縮機能の使用を検討してください。

2.3.14 リモート API 機能

TP1/Clientでは、CUPとサーバ間に常設コネクションを確立して、CUPが発行したAPIをサーバ側に転送して、サーバ側のプロセスで実行できます。このような機能をリモートAPI機能といいます。リモートAPI機能を使うと、ファイアウォールの内側にあるUAPに対しても、サービスを要求できます。ファイアウォールとは、共用ネットワークと使用制限のあるネットワークとの間に位置し、第三者が共用以外のネットワークに不正侵入することを防ぐためのハードウェア、およびソフトウェアをいいます。

CUPでは、リモートAPI機能を使用してファイアウォールを通過する場合、dc_clt_cltin_s関数(flagsにDCCLT_NO_AUTHENTを指定)を発行したあと、常設コネクション確立要求をTP1/Serverのrapリスナーに送信します。

常設コネクションの確立応答を受信した時点で、rap サーバとの常設コネクションが確立されます。コネクション確立後は、rap サーバに対して発行される要求は、常設コネクションを使用して送信します。ただし、常設コネクションが解放されたあとは、再び rap リスナーへ要求を送信します。

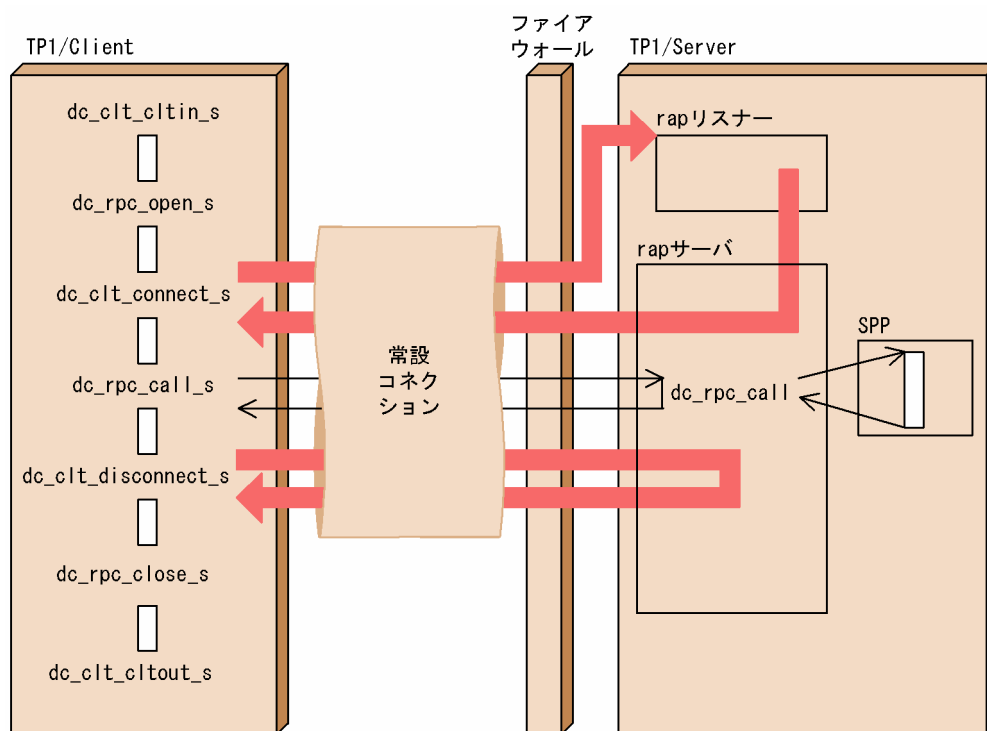
この機能を使用する場合は、クライアント環境定義 DCCLTRAPHOST を指定してください。

また、DCCM3 の論理端末に対してもリモート API 機能を使用してサービスを要求できます。

なお、クライアント環境定義 DCCLTRAPAUTOCONNECT に Y を指定している場合、CUP と rap サーバまたは DCCM3 の論理端末との間に自動的に常設コネクションを確立します。この場合、dc_clt_connect_s 関数、および dc_clt_disconnect_s 関数を実行する必要はありません。

リモート API 機能について、次の図に示します。

図 2-7 リモート API 機能



(1) 前提条件

リモート API 機能は、サービス要求先の TP1/Server Base が 03-05 以降の場合に使用できます。

リモート API 機能を使用して、トランザクションを制御する場合、サービス要求先の TP1/Server Base が 03-06 以降の必要があります。

ただし、DCCM3 の論理端末にサービスを要求するには、サービス要求先の DCCM3 が 09-02 以降の場合で、TP1/Client/W 03-04 以降、および TP1/Client/P 03-04 以降の必要があります。

(2) 適用範囲

この機能を使用して DCCM3 に対して RPC を行う場合、CUP がルートトランザクションとなるようなトランザクションは生成できません。また、同時に複数の相手システムとの常設コネクションを確立することはできません。

なお、この機能では、XATMI インタフェースは使用できません。

(3) 常設コネクション確立先の選定方法

常設コネクションの確立先は、dc_clt_connect_s 関数で設定する flags, およびクライアント環境定義 DCCLTDCCMHOST と DCCLTRAPHOST の指定によって異なります。リモート API 機能を使用してファイアウォールを通過する場合は、クライアント環境定義 DCCLTRAPHOST を指定してください。

関数の設定、および定義の指定と常設コネクション確立要求先との関係を、次の表に示します。

表 2-2 関数の設定、および定義の指定と常設コネクション確立要求先との関係

引数 flags	クライアント環境定義		常設コネクション確立要求先
	DCCLTDCCMHOST	DCCLTRAPHOST	
DCNOFLAGS	○	○	rap サーバ, または DCCM3 の論理端末 ^{*1}
		—	CUP 実行プロセス
	—	○	rap サーバ, または DCCM3 の論理端末 ^{*1}
		—	CUP 実行プロセス
DCCLT_DCCM3	○	○	DCCM3 の論理端末 ^{*2}
		—	DCCM3 の論理端末 ^{*2}
	—	○	エラーリターン
		—	エラーリターン

(凡例)

○ : 指定あり

— : 指定なし

注※1

DCCLTRAPHOST に指定された DCCM3 の論理端末に対して常設コネクションの確立を要求します。

注※2

DCCLTDCCMHOST に指定された DCCM3 の論理端末に対して常設コネクションの確立を要求します。

(4) リモート API 機能を使用して、トランザクションを制御するときの注意事項

- クライアント環境定義 DCCLTRAPHOST に指定する常設コネクション確立要求先は、TP1/Server Base 03-06 以降の必要があります。これより古いバージョンを使用した場合、dc_trn_begin_s 関数は DCCLTER_PROTO でエラーリターンします。また、常設コネクション確立先を DCCM3 論理端末とした場合も同様で、dc_trn_begin_s 関数は DCCLTER_PROTO でエラーリターンします。
- TP1/Server のオンラインテスト機能は使用できません。クライアント環境定義 DCUTOKEY にテストユーザ ID を指定した場合、dc_trn_begin_s 関数は DCCLTER_PROTO でエラーリターンします。

2.3.15 同期応答型 RPC タイムアウト時のサーバ負荷軽減

TP1/Client の CUP から dc_rpc_call_s 関数を実行すると、TP1/Server ではサービス要求を受け付けません。

この要求は SPP の実行待ち時間、実行時間、通信障害などによって、遅れるおそれがあるため、TP1/Client 側では応答待ち時間に上限を設けることで異常を監視しています。

一方 TP1/Server 側では、TP1/Client の最大応答待ち時間を認識していないため、TP1/Client 側ではタイムアウトを検出していても、TP1/Server 側はサービスを処理し続けている場合があります。

同期応答型 RPC タイムアウト時のサーバ負荷軽減機能を使用すると、TP1/Server 側では上記のような不要な処理を軽減できます。

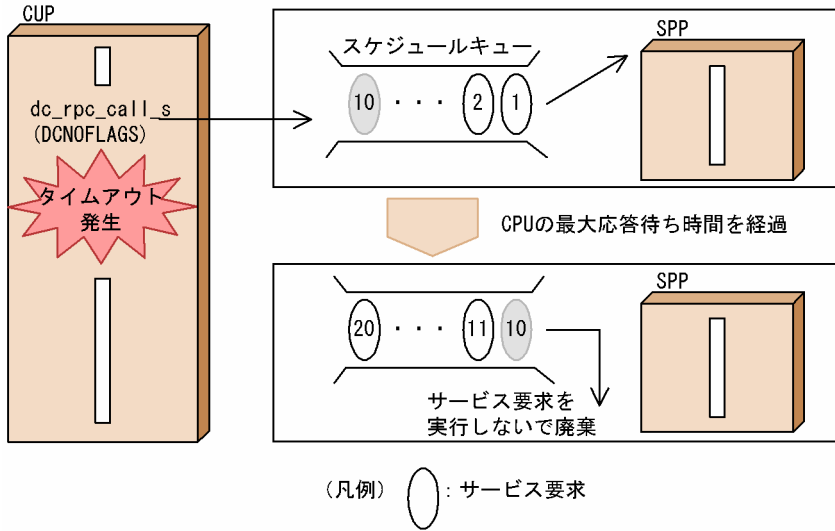
同期応答型 RPC タイムアウト時のサーバ負荷軽減機能を使用するかどうかは、クライアント環境定義 DCWATCHTIMRPCINHERIT で指定します。

この機能は、サービス要求先の TP1/Server Base が 03-05 以降の場合に使用できます。

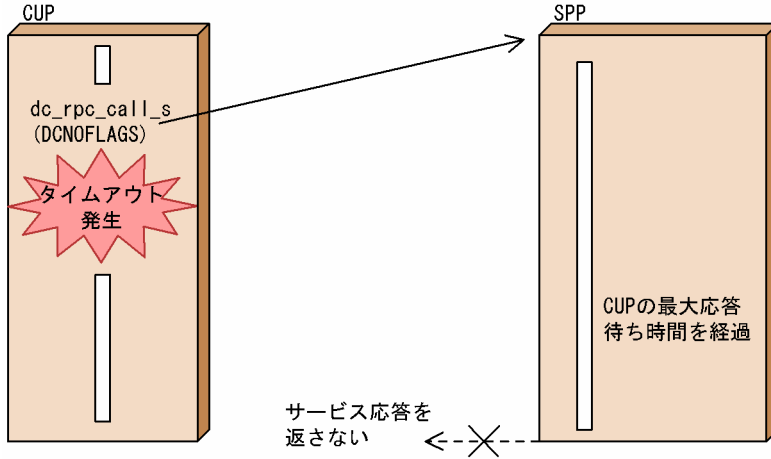
同期応答型 RPC タイムアウト時のサーバ負荷軽減機能の処理概要を次の図に示します。

図 2-8 同期応答型 RPC タイムアウト時のサーバ負荷軽減機能の処理概要

- サーバ側で負荷軽減のためサービス要求を廃棄する例



- サーバ側で負荷軽減のためサービス応答を返さない例



2.4 トランザクション制御

CUP からトランザクションを制御する関数を実行できます。この場合、トランザクションとして実行する SPP は、ユーザサービス定義で `atomic_update=Y` と指定しておく必要があります。

トランザクション制御の詳細については、マニュアル「OpenTP1 プログラム作成の手引」を参照してください。

なお、この機能は、TP1/Server Base のバージョンが 03-00 以降の場合に使用できます。

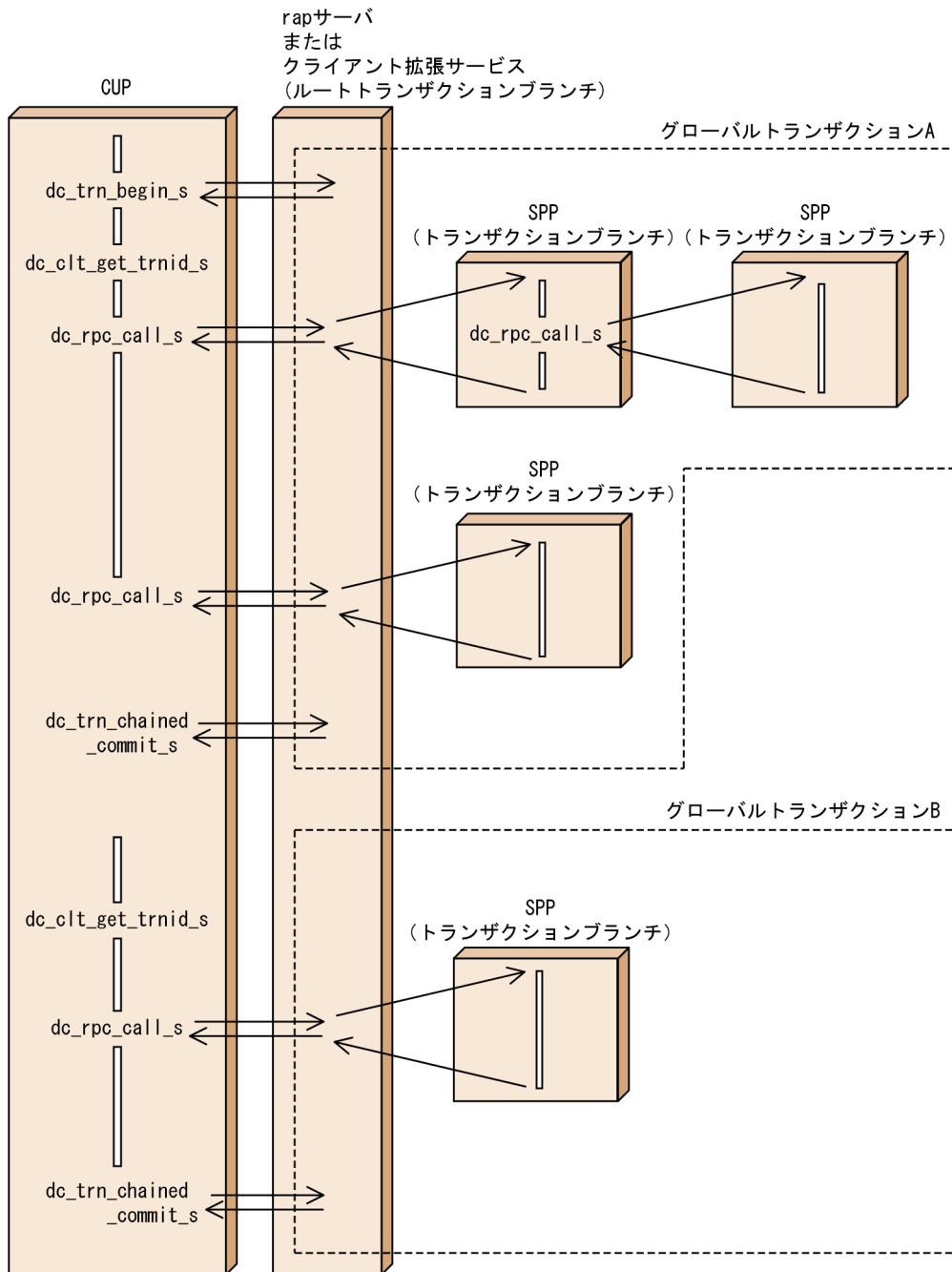
2.4.1 トランザクションの開始と同期点取得

CUP から `dc_trn_begin_s` 関数を実行して、トランザクションを開始します。

`dc_trn_begin_s` 関数を実行してから、同期点取得（コミット）までが、グローバルトランザクションの範囲となります。`dc_trn_begin_s` 関数を実行したあと、新たな `dc_trn_begin_s` 関数は実行できません。CUP から SPP へ RPC を実行すると、rap サーバまたはクライアント拡張サービスがルートトランザクションブランチとなり、RPC 先の SPP はトランザクションブランチとして実行されます。CUP はルートトランザクションブランチとならないため、CUP 上でのリソースマネージャへのアクセスをトランザクションに含めることはできません。

トランザクションと RPC の関係を次の図に示します。

図 2-9 トランザクションとRPCの関係



2.4.2 同期点取得

(1) コミット

トランザクションが正常終了したときの同期点取得（コミット）は、CUPからコミット要求の関数を呼び出して行います。グローバルトランザクションは、すべてのトランザクションブランチが正常に終了したことで正常終了となります。

(a) 連鎖, 非連鎖モードのコミット

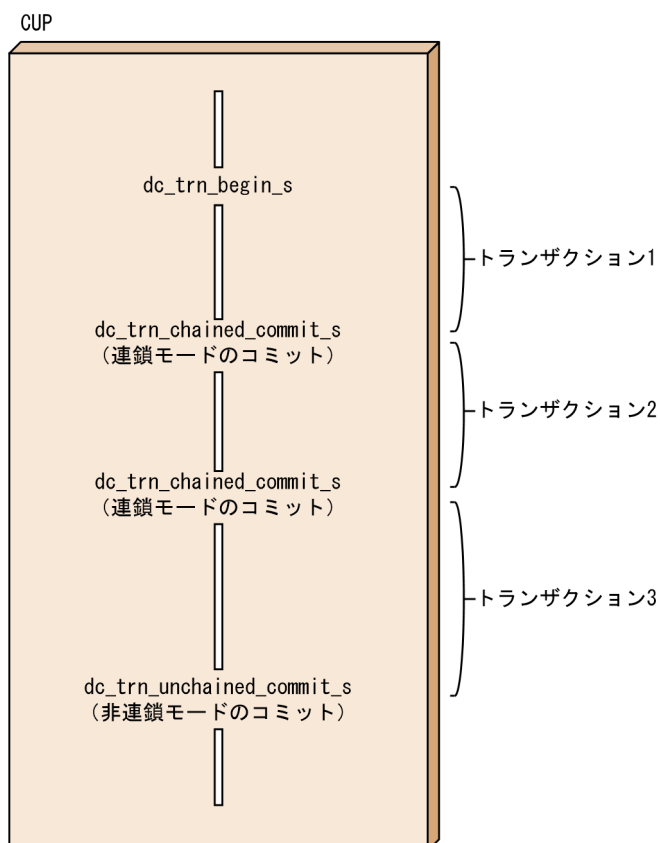
トランザクション処理の同期点取得には、一つのトランザクションの終了後、同期点を取得して次のトランザクションを続けて起動する連鎖モードのコミットと、トランザクションの終了で同期点を取得したあと、新たなトランザクションを起動しない非連鎖モードのコミットがあります。

連鎖モードのコミットは、`dc_trn_chained_commit_s` 関数を実行して要求します。

非連鎖モードのコミットは、`dc_trn_unchained_commit_s` 関数を実行して要求します。

トランザクションの連鎖, 非連鎖モードを次の図に示します。

図 2-10 トランザクションの連鎖, 非連鎖モード



(b) コミット要求の関数を呼び出さない場合の処理

次に示すどちらかの場合、トランザクションはロールバックされます。

- コミット要求の関数を呼び出さないで、かつ `dc_rpc_close_s` 関数または `dc_clt_cltout_s` 関数も発行しないでプログラムを終了したとき
- コミット要求の関数を呼び出す前に CUP が異常終了したとき

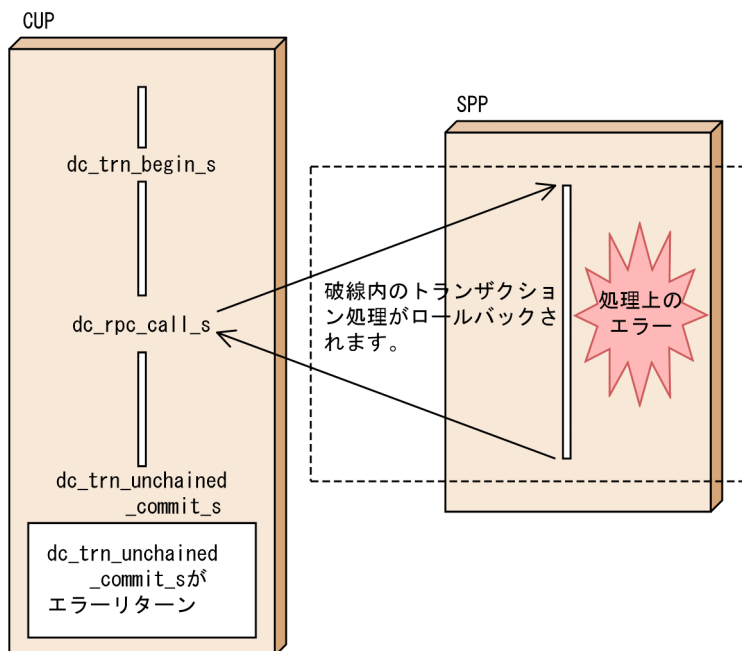
(2) ロールバック

(a) TP1/Server の処理でのエラーの場合

トランザクションでエラーが発生すると、コミット要求の関数がエラーリターンされます。そのトランザクションは部分回復対象としてロールバックされます。グローバルトランザクション内のどれか一つのトランザクションブランチでエラーが発生した場合でも、グローバルトランザクション全体がロールバックの対象となります。このとき TP1/Server は、トランザクションブランチをロールバック対象とみなして、部分回復処理をします。

TP1/Server の処理でエラーが発生した場合のトランザクションのロールバックを、次の図に示します。

図 2-11 トランザクションのロールバック (TP1/Server の処理でエラーが発生した場合)

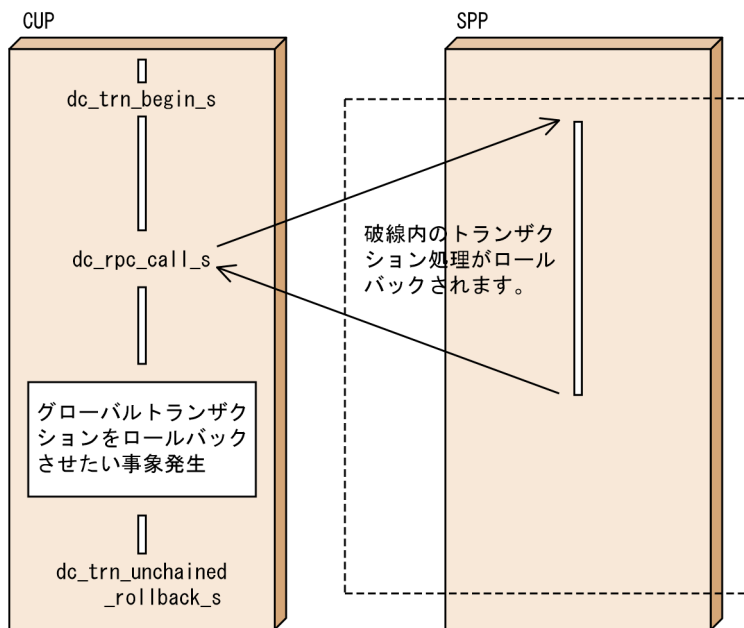


(b) ロールバック要求の関数の発行

トランザクションを CUP の判断でロールバックしたいときは、CUP からロールバック要求の関数を実行します。

ロールバック要求の関数を実行した場合のトランザクションのロールバックを、次の図に示します。

図 2-12 トランザクションのロールバック（ロールバック要求の関数を呼び出した場合）



ロールバックには、連鎖モードのロールバックと非連鎖モードのロールバックがあります。

連鎖モードのロールバックは、`dc_trn_chained_rollback_s` 関数を実行して要求します。連鎖モードのロールバック関数を実行してロールバックした場合、このトランザクションは、ロールバック処理後も、グローバルトランザクションの範囲内にあります。

非連鎖モードのロールバックは、`dc_trn_unchained_rollback_s` 関数を実行して要求します。非連鎖モードのロールバック関数を実行してロールバックした場合、このトランザクションは、ロールバック処理後、グローバルトランザクションの範囲外となります。

(3) ヒューリスティック発生時の処置

トランザクション処理でヒューリスティックが発生すると、CUPの同期点取得時にエラーリターンされません。この場合のリターン値について次に示します。

- ヒューリスティック決定の結果が、グローバルトランザクションの同期点の結果と一致しなかった場合 … DCTRNER_HEURISTIC (-3403)
- 障害のため、ヒューリスティックに完了したトランザクションブランチの同期点の結果が判明しない場合 … DCTRNER_HAZARD (-3404)

これらのリターン値が戻る原因、およびグローバルトランザクションの同期点の結果については、メッセージログファイルを参照してください。

ヒューリスティック発生時の処置の詳細については、マニュアル「OpenTP1 プログラム作成の手引」を参照してください。

(4) トランザクションの処理時間について

トランザクションに関する、次に示す時間をクライアント環境定義で指定できます。詳細については「[8.2 クライアント環境定義の詳細](#)」を参照してください。

- トランザクションブランチ限界経過時間
- トランザクションブランチの時間監視に、次に示す時間を含むかどうか
監視対象のトランザクションブランチが、RPC 機能を使ってほかのトランザクションブランチを呼び出し、その処理が終わるのを待つ時間
- トランザクション問い合わせ間隔最大時間
- トランザクションブランチ CPU 監視時間
- トランザクション同期点処理時の最大通信待ち時間
- トランザクションブランチ最大実行可能時間

(5) トランザクションブランチの統計情報取得タイプについて

トランザクションブランチごとに取得するトランザクション統計情報の取得タイプを、クライアント環境定義で指定できます。詳細については「[8.2 クライアント環境定義の詳細](#)」を参照してください。

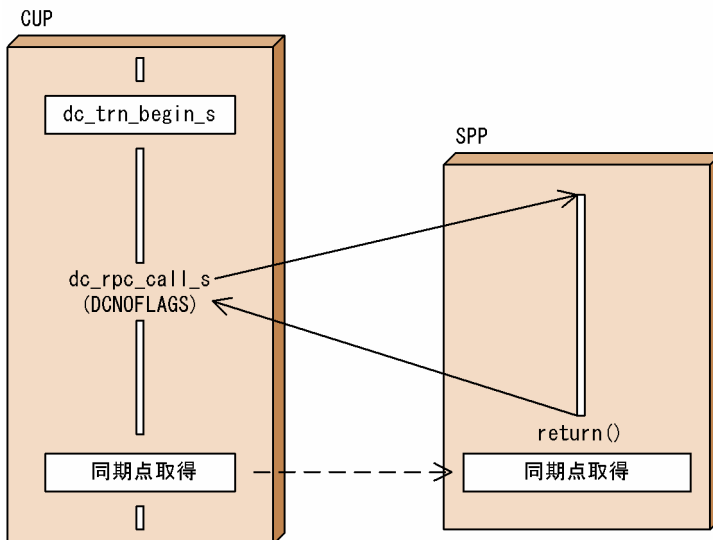
2.4.3 リモートプロシジャコールの形態と同期点の関係

(1) 同期応答型 RPC と同期点の関係

同期応答型 RPC のトランザクション処理の場合、CUP に処理結果が戻って、同期点取得処理を終えた時点で、トランザクションの終了となります。

同期応答型 RPC と同期点の関係を次の図に示します。

図 2-13 同期応答型 RPC と同期点の関係

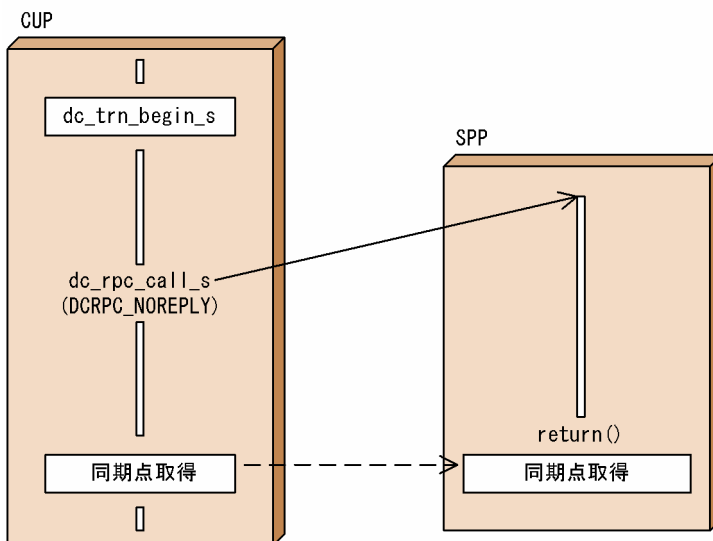


(2) 非応答型 RPC と同期点の関係

非応答型 RPC のトランザクション処理の場合は、CUP の同期点取得で SPP の処理終了を待ち、そのあとで同期点取得処理をします。

非応答型 RPC と同期点の関係を次の図に示します。

図 2-14 非応答型 RPC と同期点の関係



(3) 連鎖 RPC と同期点の関係

連鎖 RPC を使った場合は、一つの SPP のプロセスで実行します。したがって、トランザクションブランチも連鎖 RPC を使った回数に関係なく、一つになります。

連鎖 RPC のトランザクション処理の場合、同期点処理を終了した時点でトランザクションが終了して、処理していた SPP のプロセスは解放されます。

トランザクション実行中に非トランザクションの連鎖 RPC を使った場合、通常は同期点処理を終了した時点で、処理していた SPP のプロセスを解放します。同期点処理の終了で処理していた SPP のプロセスを解放しないで、同期応答型 RPC で解放する場合は、ユーザサービス定義の `rpc_extend_function` オペランドに `00000002` を指定してください。

連鎖 RPC と同期点の関係を、次の図に示します。

図 2-15 連鎖 RPC と同期点の関係 (トランザクションの連鎖 RPC)

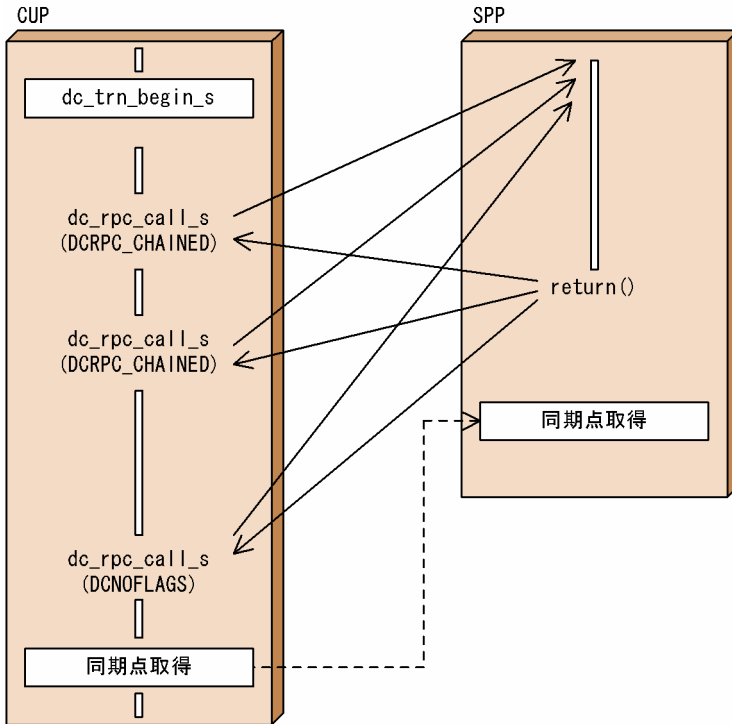
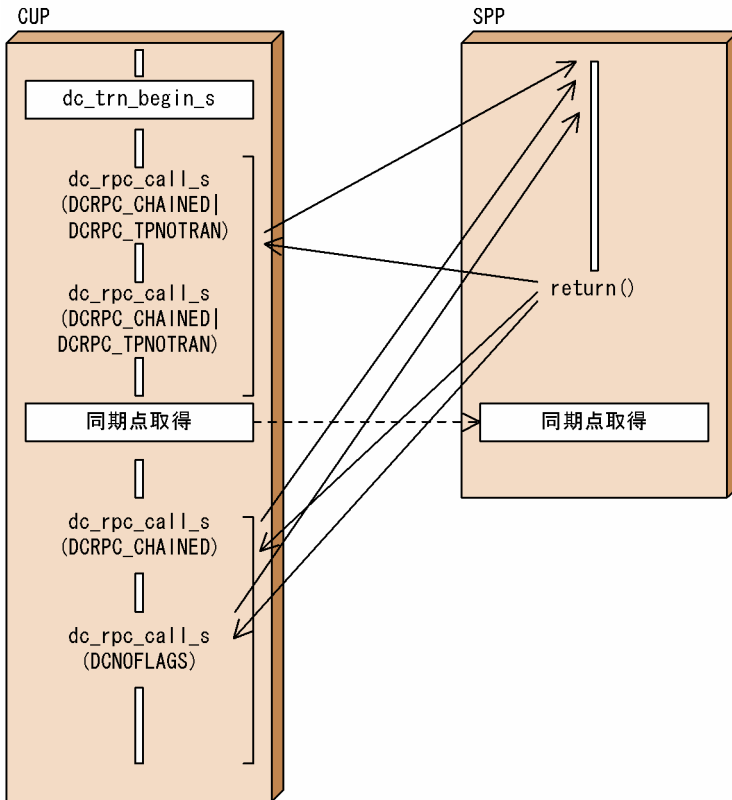


図 2-16 連鎖 RPC と同期点の関係 (非トランザクションの連鎖 RPC で終了しない指定をした場合)



2.4.4 現在のトランザクションに関する識別子の取得

CUP から `dc_clt_get_trnid_s` 関数を実行すると、現在のトランザクショングローバル識別子、およびトランザクションブランチ識別子を取得できます。

障害が発生した場合、CUP から開始したトランザクションがコミットされたかどうかを調べるために、トランザクショングローバル識別子が必要になります。障害に備えて、次に示す関数を実行したあとには、必ず `dc_clt_get_trnid_s` 関数を実行してください。

- `dc_trn_begin_s` 関数
- `dc_trn_chained_commit_s` 関数
- `dc_trn_chained_rollback_s` 関数

2.4.5 現在のトランザクションに関する情報の報告

CUP から `dc_trn_info_s` 関数を実行すると、リターン値で、トランザクションとして稼働中かどうかを確認できます。

2.4.6 障害発生時のトランザクションの同期点を検証する方法

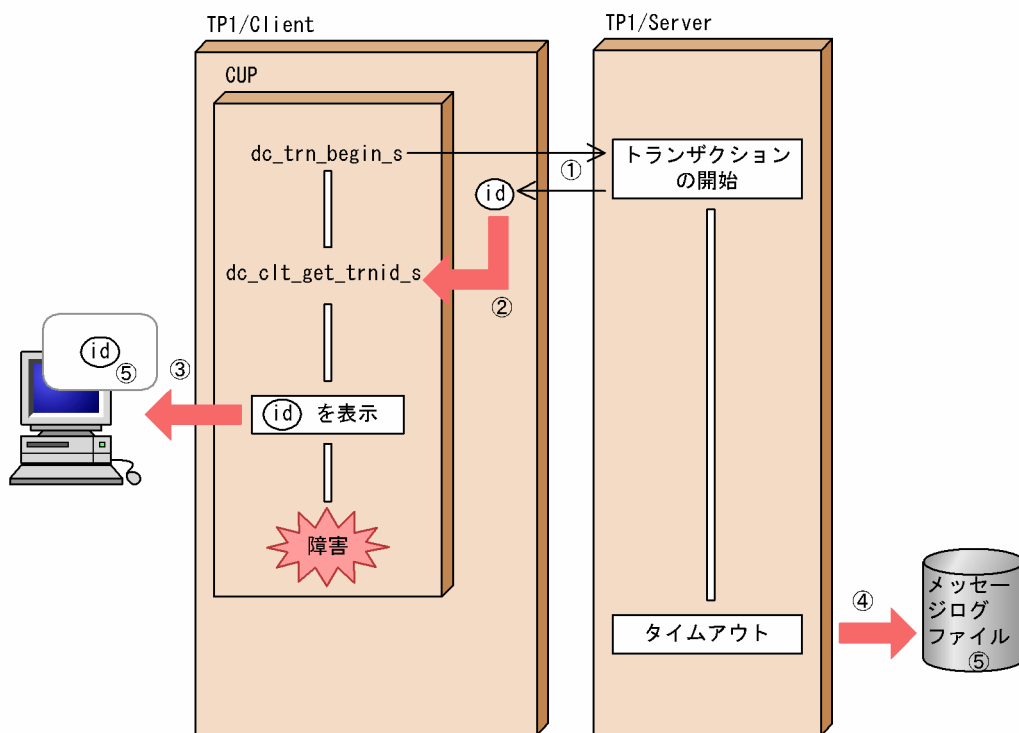
CUP から開始したトランザクションで障害が発生した場合、そのトランザクションブランチがコミットされたかどうかを検証できます。この場合、トランザクション開始後に、必ず `dc_clt_get_trnid_s` 関数を実行して現在のトランザクショングローバル識別子、およびトランザクションブランチ識別子を取得しておく必要があります。

CUP で取得しておいたトランザクショングローバル識別子と、サーバ側のメッセージログファイルに出力されるトランザクションの結果を突き合わせることによって、CUP から開始したトランザクションがコミットされたかどうか検証できます。

メッセージログファイルの内容は、`logcat` コマンドで表示できます。`logcat` コマンドについては、マニュアル「OpenTP1 運用と操作」を参照してください。

障害発生時のトランザクションの同期点を検証する方法を次の図に示します。

図 2-17 障害発生時のトランザクションの同期点を検証する方法



(凡例) (id) : トランザクショングローバル識別子

(説明)

- ① CUPから`dc_trn_begin_s`関数を実行してトランザクションが開始されると、TP1/Serverはそのトランザクションのトランザクショングローバル識別子をTP1/Clientに通知します。
- ② CUPは`dc_clt_get_trnid_s`関数を実行して、トランザクショングローバル識別子を取得します。
- ③ 取得したトランザクショングローバル識別子を表示します。
- ④ CUPで障害が発生し、TP1/Server側でタイムアウトが発生すると、トランザクションの処理結果がメッセージログファイルに出力されます。
- ⑤ ③で表示したトランザクショングローバル識別子とメッセージログファイルの内容を検証します。

2.4.7 トランザクションを制御するときの注意事項

- トランザクションの範囲内でタイムアウトが発生した場合、次に実行する関数が DCCLTER_OLTF_NOT_UP, または DCRPCER_OLTF_NOT_UP でエラーリターンする場合があります。これは、サーバ側の処理中にクライアントの最大応答待ち時間が満了し、クライアント側が先にタイムアウトした場合に、サーバとの接続が切断されるため、この現象が発生します。
この現象を回避するためには、クライアント環境定義 DCWATCHTIM に指定する値を、トランザクショナル RPC 実行プロセスの watch_time に指定する値よりも大きくする必要があります。
トランザクショナル RPC 実行プロセスの watch_time は、次に示す方法で指定できます。
 - TP1/Server Base 03-02 以前システム共通定義、またはユーザサービスデフォルト定義の watch_time オペランドで指定します。
 - TP1/Server Base 03-03 以降クライアントサービス定義の watch_time オペランドで指定します。また、クライアント環境定義 DCWATCHTIMINHERIT および DCCLTDELAY を指定することでも、この現象を回避できます。
- CUP からトランザクションを起動した場合は、必ず非連鎖モードのコミット、またはロールバック要求をしてから CUP を終了させてください。トランザクションを完結させずに CUP を終了すると、同期点でのトランザクションの結果が検証できなくなります。このとき、トランザクショナル RPC 実行プロセスは、トランザクションブランチ限界経過時間またはトランザクション問い合わせ間隔最大時間に達するまで、実行中のままとなります。経過時間に達すると、トランザクションはロールバックされます。
CUP が非連鎖モードのコミット、ロールバック要求をしないで CUP の終了、または dc_clt_cltout_s 関数を実行した場合には、自動的に非連鎖モードのコミットが行われます。ただし、CUP には同期点の結果は通知されません。
- 実時間監視タイムアウトになったあと、CUP から dc_rpc_call_s 関数を実行すると、DCRPCER_SYSERR になる場合があります。
- サーバ側で監視している各種タイマ値のどれかが満了となった場合、サーバがダウンする場合があります。サーバがダウンした場合、タイミングによっては、CUP から実行した関数が最大応答待ち時間分待ち状態となり、タイムアウトとなります。これは、クライアント側から送信するパケットをサーバ側が認識できないため、この現象が発生します。この現象を回避するために、各種タイマ値には適切な値を設定してください。

2.5 TCP/IP 通信機能

TCP/IP 通信機能を使用したメッセージの送受信について説明します。

注意事項

TCP/IP 通信機能を使用することによって MHP と通信できるようになるため、ここでは、通信する相手システムを「MHP」と呼んでいます。相手システムは MHP に限らず、ユーザが自由に選択できます。

TCP/IP 通信機能を使用したメッセージの送受信には次に示す 3 種類があります。

- CUP から MHP へのメッセージの一方送信
- MHP から CUP へのメッセージの一方受信
- MHP と CUP との間でのメッセージの送受信

また、メッセージの送受信時には、自動的にメッセージ長を付加したり、メッセージの送達確認をすることもできます。

2.5.1 メッセージの一方送信

CUP から MHP へ一方的にメッセージを送信できます。これを**メッセージの一方送信**といいます。

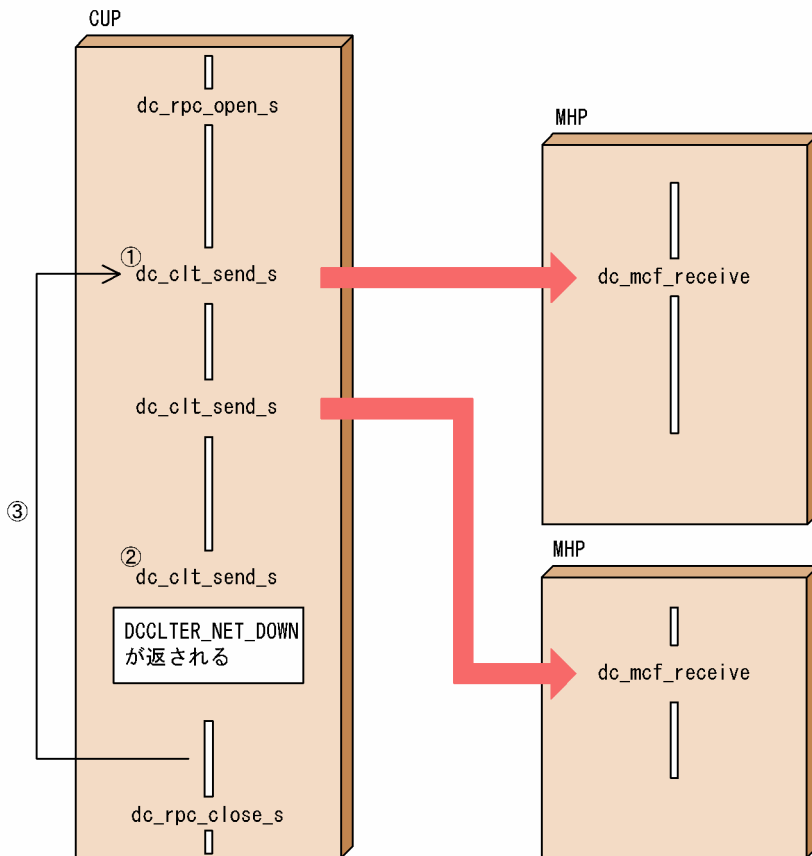
メッセージの一方送信を行う場合は、CUP で flags に DCCLT_ONERWAY_SND を指定した `dc_rpc_open_s` 関数を実行しておく必要があります。その後、CUP から `dc_clt_send_s` 関数を実行して、MHP へメッセージを送信します。メッセージの組み立て機能、または送達確認機能を使用する場合は、`dc_clt_assem_send_s` 関数を実行します。

メッセージを一方送信するには、接続先のノード名、およびポート番号（MCF 通信構成定義の定義コマンド `mcftalccn` の `portno` で指定したポート番号）を指定する必要があります。

- 接続先のノード名は次のどちらかの方法で指定します。
 - (a) `dc_clt_send_s` 関数、または `dc_clt_assem_send_s` 関数の引数 `hostname` に指定
 - (b) クライアント環境定義 `DCSNDHOST` に指定
- 接続先のポート番号は次のどちらかの方法で指定します。
 - (c) `dc_clt_send_s` 関数、または `dc_clt_assem_send_s` 関数の引数 `portnum` に指定
 - (d) クライアント環境定義 `DCSNDPORT` に指定

メッセージの一方送信を次の図に示します。

図 2-18 メッセージの一方送信



(説明)

- ① MHPが起動されたあと、CUPを起動し、`dc_clt_send_s`関数を実行します。
- ② `dc_clt_send_s`関数を発行したときに、MHPからコネクションが解放されていた場合、CUPにリターン値`DCCLTER_NET_DOWN`を返します。
- ③ 再びメッセージを一方送信するには、`dc_clt_send_s`関数を実行します。

2.5.2 メッセージの一方受信

MHP から送信されたメッセージを CUP で受信できます。これをメッセージの一方受信といいます。

メッセージの一方受信を行う場合は、CUP で flags に `DCCLT_ONEWAY_RCV` を指定した `dc_rpc_open_s` 関数を実行しておく必要があります。その後、CUP で、`dc_clt_receive_s` 関数、または `dc_clt_receive2_s` 関数を実行して、MHP からのメッセージを TCP/IP プロトコルを使用して受信します。メッセージの組み立て機能、または送達確認機能を使用する場合は、`dc_clt_assem_receive_s` 関数を実行します。

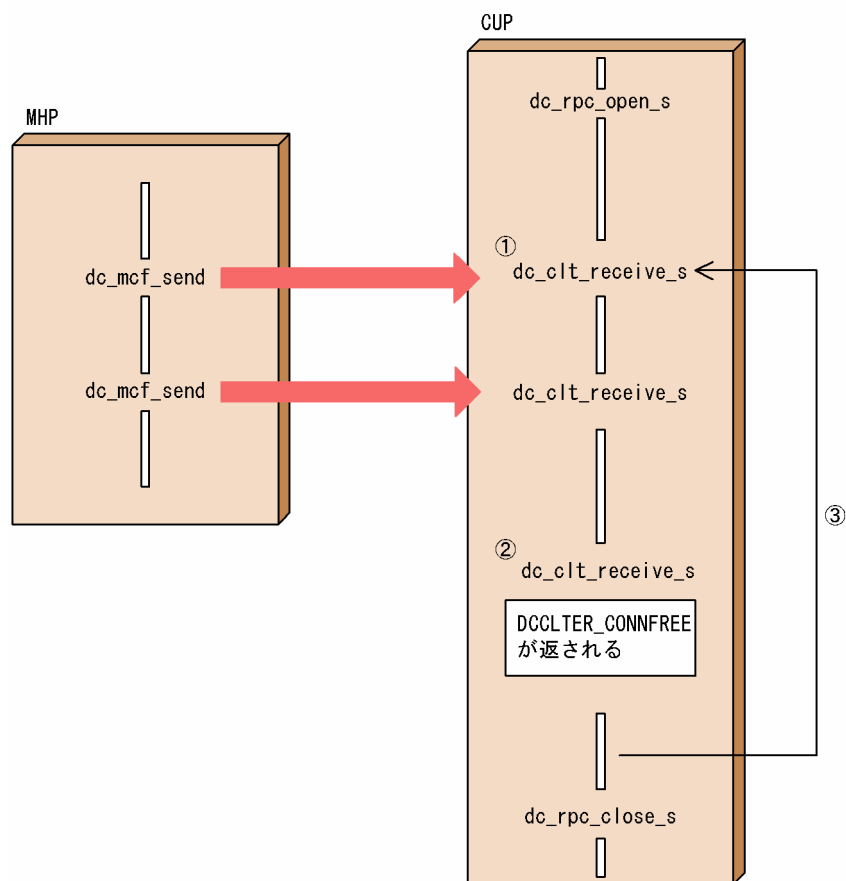
TCP/IP プロトコルでは、一つのメッセージを複数のパケットに分割したり、複数のメッセージを一つのパケットに詰め込んだりします。そのため、受信したメッセージの切れ目は、ユーザが指定するメッセージ長で判断します。ユーザはメッセージ長を含めた固定長のヘッダを最初に受信し、ヘッダに含まれているメッセージ長を指定して実際のメッセージを受信してください。ただし、メッセージの組み立て機能、または送達確認機能を使用するときは、これらの処理を CUP で作り込む必要はありません。

指定したメッセージ長よりも短いメッセージを受信した場合、TP1/Client は、メッセージが分割されているものとみなし、指定した長さ分のメッセージを受信するまで、CUP に制御を戻しません。指定した長さ分のメッセージを受信する前に時間切れ（タイムアウト）や、エラーが発生した場合、`dc_clt_receive_s` 関数を実行すると、受信したメッセージは破棄されます。`dc_clt_receive2_s` 関数、または `dc_clt_assem_receive_s` 関数を実行する場合は、その時点までに受信したメッセージを破棄しないで、CUP に制御を戻すことができます。タイムアウトやエラー発生時に指定した長さ分のメッセージに満たない場合でも、その時点までのメッセージを受信できます。ただし、それ以降のメッセージの組み立てについては、ユーザの責任で行ってください。

メッセージを一方受信するには、あらかじめ、クライアント環境定義 DCRCVPORT に、CUP の受信用ポート番号（MCF 通信構成定義の定義コマンド `mcftalccn` の `oportno` で指定したポート番号）を指定しておく必要があります。マルチスレッドで動作する CUP を含め、同一マシンで CUP を複数実行する場合は、クライアント環境定義 DCRCVPORT を CUP ごと（スレッドごと）に異なるポート番号となるように設定してください。CUP ごとのクライアント環境定義の設定方法については、「[8.2.3 TP1/Client/W の設定方法](#)」、および「[8.2.4 TP1/Client/P の設定方法](#)」を参照してください。受信用ソケットの開設は、`dc_rpc_open_s` 関数実行時に行います。

メッセージの一方受信を、次の図に示します。

図 2-19 メッセージの一方受信



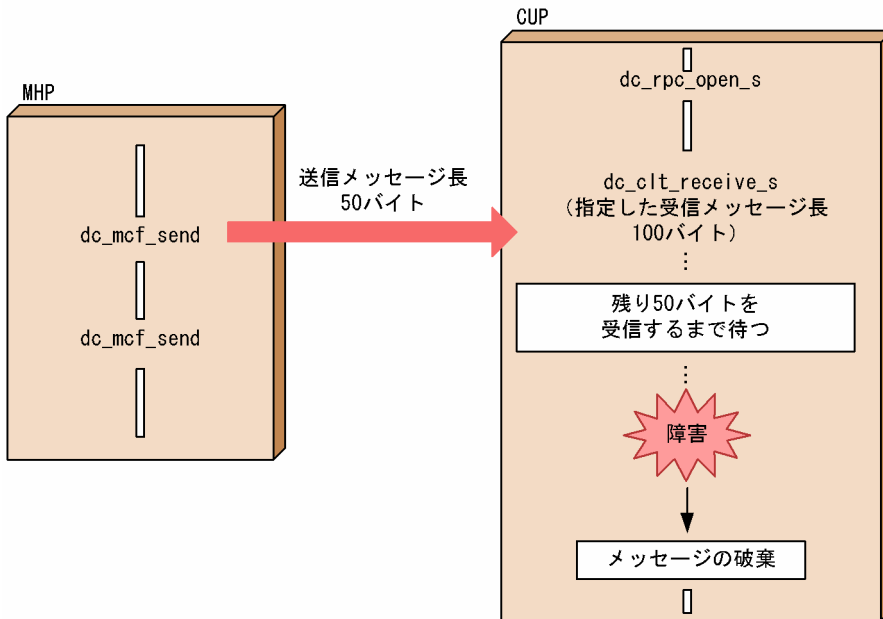
(説明)

- ① MHPが接続の確立要求を再試行している間にCUPを起動し、dc_clt_receive_s関数を実行します。再試行中に接続を確立できなかった場合、mcftactcnコマンドを入力して接続を確立します。
- ② MHPから接続が解放された場合、CUPにリターン値DCCLTER_CONNFREEを返します。
- ③ 再びメッセージを一方受信するには、dc_clt_receive_s関数を実行します。この場合、mcftactcnコマンドを入力して、接続を確立してください。

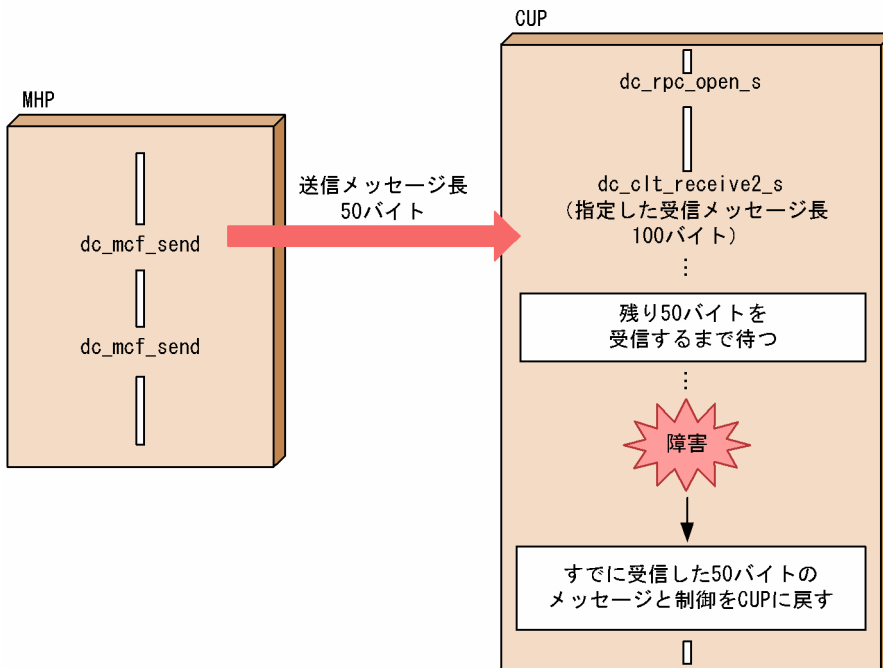
障害発生時のメッセージの一方受信を、次の図に示します。

図 2-20 メッセージの一方受信 (障害発生時)

● dc_clt_receive_s関数の場合



● dc_clt_receive2_s関数の場合



2.5.3 メッセージの送受信

CUP と MHP との間で、メッセージを送受信できます。

メッセージの送受信を行う場合は、CUP で flags に DCCLT_SNDRCV を指定した `dc_rpc_open_s` 関数を実行しておく必要があります。メッセージの送受信は、コネクションが解放されていない場合同じコネクションを使用してメッセージを送受信します。

MHP がサーバ型の場合

CUP から MHP に対してメッセージを送信した際に確立したコネクションを使用してメッセージの送受信を行います。このため、CUP では受信用のソケットを使用しません。

CUP から `dc_clt_send_s` 関数を実行して MHP へメッセージを送信します。その上で `dc_clt_receive_s` 関数、または `dc_clt_receive2_s` 関数を実行して、MHP からのメッセージを受信します。メッセージの組み立て機能、または送達確認機能を使用する場合は、CUP から `dc_clt_assem_send_s` 関数を実行して MHP へメッセージを送信します。その上で `dc_clt_assem_receive_s` 関数を実行して MHP からのメッセージを受信します。

MHP がクライアント型の場合

MHP から送信されたメッセージを CUP で受信した際に確立したコネクションを使用してメッセージの送受信を行います。

CUP で `dc_clt_receive_s` 関数、または `dc_clt_receive2_s` 関数を実行して、MHP からのメッセージを受信します。その上で `dc_clt_send_s` 関数を実行して MHP へメッセージを送信します。メッセージの組み立て機能、または送達確認機能を使用する場合は、CUP で `dc_clt_assem_receive_s` 関数を実行して MHP からのメッセージを受信します。その上で `dc_clt_assem_send_s` 関数を実行して MHP へメッセージを送信します。

メッセージを送受信するには、次に示す指定をしておく必要があります。クライアント環境定義の各オペランドの詳細については、「[8.2.5 TP1/Client/W と TP1/Client/P で共通のオペランド](#)」を参照してください。

MHP がサーバ型の場合

- 接続先のノード名を次のどちらかの方法で指定します。
 - (a) `dc_clt_send_s` 関数、または `dc_clt_assem_send_s` 関数の引数 `hostname` に指定
 - (b) クライアント環境定義 `DCSNDHOST` に指定
- 接続先のポート番号（MCF 通信構成定義の定義コマンド `mcftalccn` の `portno` オペランドで指定したポート番号）を次のどちらかの方法で指定します。
 - (c) `dc_clt_send_s` 関数、または `dc_clt_assem_send_s` 関数の引数 `portnum` に指定
 - (d) クライアント環境定義 `DCSNDPORT` に指定
- クライアント環境定義 `DCSOCKOPENATRCV` に `Y` を指定します。
CUP では受信用ソケットを使用しません。クライアント環境定義 `DCSOCKOPENATRCV` に `Y` を指定し、受信用ソケットを開設しないでください。

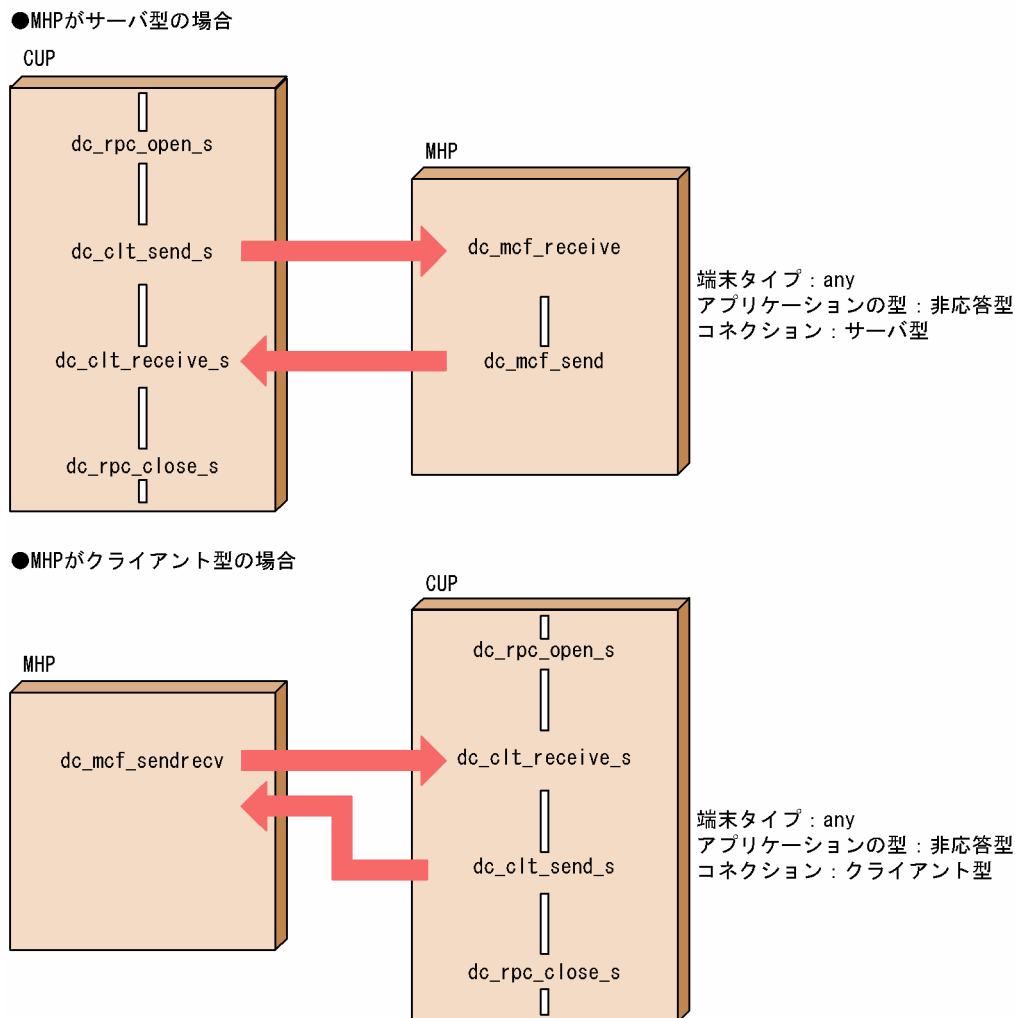
MHP がクライアント型の場合

クライアント環境定義 `DCRCVPORT` に CUP の受信用ポート番号（MCF 通信構成定義の定義コマンド `mcftalccn` の `oportno` で指定したポート番号）を指定してください。

マルチスレッドで動作する CUP を含め、同一マシンで CUP を複数実行する場合は、クライアント環境定義 `DCRCVPORT` を CUP ごと（スレッドごと）に異なるポート番号となるように設定してください。CUP ごとのクライアント環境定義の設定方法については、「[8.2.3 TP1/Client/W の設定方法](#)」、および「[8.2.4 TP1/Client/P の設定方法](#)」を参照してください。

メッセージの送受信を次の図に示します。

図 2-21 メッセージの送受信



2.5.4 メッセージの組み立て機能と送達確認機能

TP1/Client では、メッセージの送受信時に自動的にメッセージの先頭 4 バイトにメッセージ長を付加・削除する機能があります。これを、**メッセージの組み立て機能**といいます。この機能を使用すると、CUP を作成するときにメッセージ長を意識する必要がなくなります。

さらに、先頭 4 バイトのメッセージ長に加えて、1 バイトのセグメント情報、および 6 バイトのメッセージ ID を付加してメッセージを送信し、応答専用データを受信して、メッセージの送達確認をすることもできます。これを、**メッセージの送達確認機能**といいます。この機能を使用すると、CUP を作成するときにメッセージの妥当性およびメッセージが送達したかどうかを意識する必要がなくなります。

メッセージの組み立て機能とメッセージの送達確認機能は、共に次の関数を発行してメッセージを送受信することで実現します。

- dc_clt_assem_send_s 関数

- dc_clt_assem_receive_s 関数

これらの関数を発行したときに、メッセージの組み立て機能だけ使用するか、さらにメッセージの送達確認機能も使用するかは、クライアント環境定義 DCCLTDELIVERYCHECK で次の表に示すように指定します。

表 2-3 使用する機能とクライアント環境定義 DCCLTDELIVERYCHECK の指定値の関係

使用する機能	クライアント環境定義 DCCLTDELIVERYCHECK の指定値
メッセージの組み立て機能	N, または指定を省略
メッセージの送達確認機能	Y

(1) 使用するための条件

どちらの機能を使用する場合も、引数 flags に DCCLT_ONEWAY_SND, DCCLT_ONEWAY_RCV, または DCCLT_SNDRCV を指定した dc_rpc_open_s 関数を発行しておく必要があります。

さらに、相手システムである TP1/NET/TCP/IP も、受信メッセージの組み立て機能、または送達確認機能を使用する設定にします。TP1/NET/TCP/IP の設定方法については、マニュアル「OpenTP1 プロトコル TP1/NET/TCP/IP 編」を参照してください。ただし、送受信メッセージの形式が同じである場合は、相手システムは限定しません。

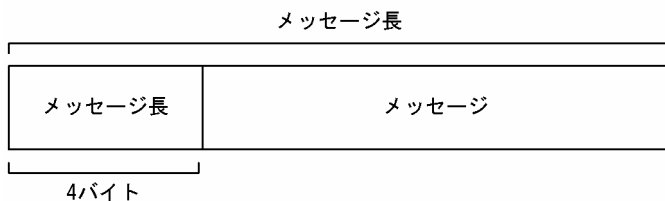
(2) メッセージの形式

メッセージの組み立て機能を使用する場合と、送達確認機能を使用する場合に分けて、メッセージの形式について説明します。

(a) メッセージの組み立て機能を使用する場合

メッセージの組み立て機能を使用する場合の、送受信メッセージの形式を次の図に示します。

図 2-22 メッセージの組み立て機能使用時の送受信メッセージの形式



メッセージ長とは、組み立て、分解をするメッセージの全体長です。

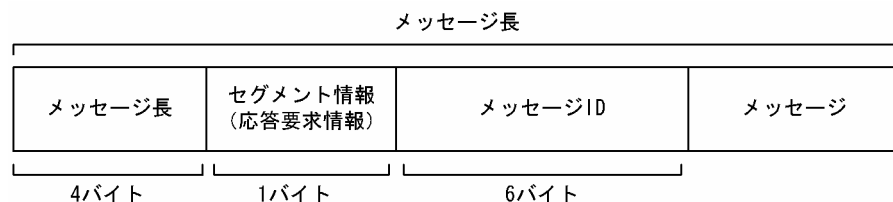
(b) メッセージの送達確認機能を使用する場合

ここでは、メッセージの送達確認機能を使用する場合の送受信メッセージの形式、および応答専用データの形式について説明します。

- 送受信メッセージの形式

TP1/Client と TP1/NET/TCP/IP 間で送受信するメッセージの形式を次の図に示します。

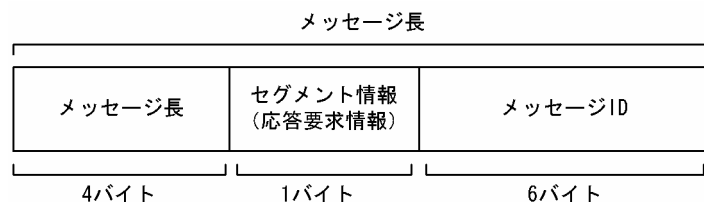
図 2-23 送達確認機能使用時のメッセージの形式



それぞれの項目について説明します。

- メッセージ長**
 組み立て、分解をするメッセージの全体長です。
- セグメント情報**
 セグメントの種別と応答要求のための種別です。単一セグメントおよび自動応答要求を示す、0x18 が設定されます。
- メッセージ ID**
 メッセージと応答専用データの組み合わせを確認するための情報です。メッセージを送信するたびに、ユニークな値が設定されます。
- 応答専用データの形式**
 メッセージの送達確認機能を使用するとき送受信される、応答専用データの形式を次の図に示します。

図 2-24 応答専用データの形式



それぞれの項目について説明します。

- メッセージ長**
 組み立て、分解をするメッセージの全体長です。
- セグメント情報**
 セグメントの種別と応答要求のための種別です。単一セグメントおよび応答専用データを示す、0x10 が設定されます。
- メッセージ ID**
 メッセージと応答専用データの組み合わせを確認するための情報です。受信したメッセージに付加されているメッセージ ID と、同じ値が設定されます。TP1/Client は、送信時にメッセージに付加したメッセージ ID と、TP1/NET/TCP/IP が応答専用データに付加したメッセージ ID を比較し、一致した場合に処理を終了します。

(3) メッセージの組み立て機能を使用する場合の送受信の流れ

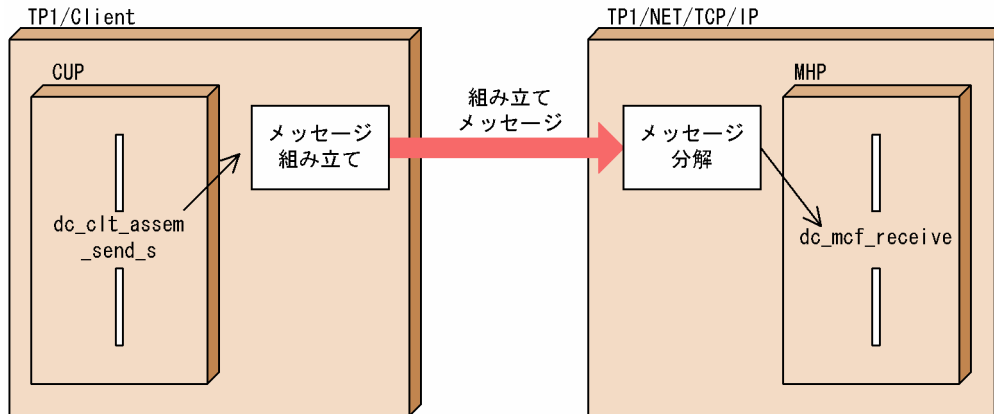
メッセージの組み立て機能を使用する場合のメッセージの組み立て、分解の流れについて説明します。

(a) メッセージの送信

メッセージの組み立て機能を使用してメッセージを送信する場合、`dc_clt_assem_send_s` 関数を発行します。`dc_clt_assem_send_s` 関数は、メッセージの先頭 4 バイトにメッセージ長を付加して送信します。

メッセージの組み立て機能を使用するときの送信の流れを次の図に示します。

図 2-25 メッセージの組み立て機能使用時の送信の流れ



メッセージを送信してすぐに接続を解放したい場合は、`dc_clt_assem_send_s` 関数の引数 `flags` に `DCCLT_SND_CLOSE` を指定します。`DCNOFLAGS` を指定した場合、`dc_rpc_close_s` 関数を発行するまで接続は解放されません。ただし、障害時を除きます。

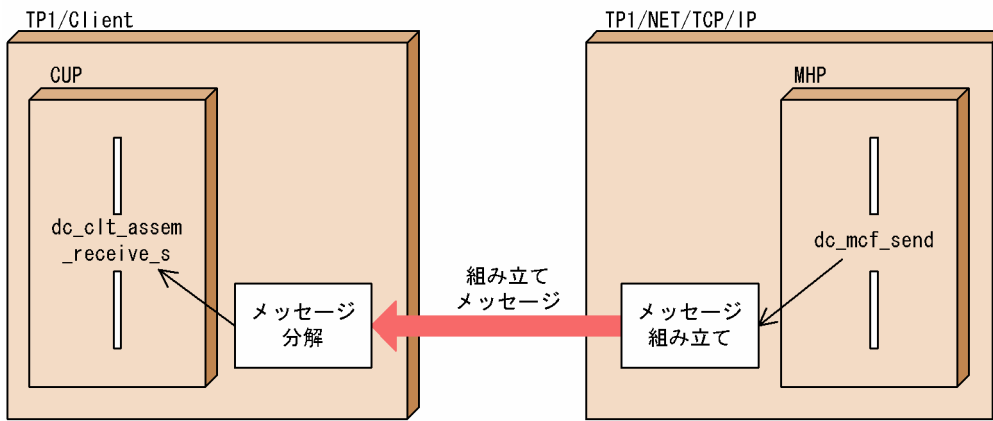
通信先となる TP1/NET/TCP/IP のホスト名はクライアント環境定義 `DCSNDHOST` に、ポート番号はクライアント環境定義の `DCSNDPORT` に指定します。ホスト名とポート番号は、`dc_clt_assem_send_s` 関数の引数に指定することもできます。

(b) メッセージの受信

メッセージ組み立て機能を使用してメッセージを受信する場合、`dc_clt_assem_receive_s` 関数を発行します。送信時にメッセージの先頭に付加された、メッセージ長分のメッセージを受信します。

メッセージの組み立て機能を使用するときの受信の流れを次の図に示します。

図 2-26 メッセージの組み立て機能使用時の受信の流れ



メッセージの受信後すぐに接続を解放したい場合は、`dc_clt_assem_receive_s` 関数の引数 `flags` に `DCCLT_RCV_CLOSE` を指定します。 `DCNOFLAGS` を指定した場合、`dc_rpc_close_s` 関数を発行するまで、接続は解放されません。ただし、障害時を除きます。

CUP のポート番号は、クライアント環境定義 `DCRCVPORT` に指定します。

(4) メッセージの送達確認機能を使用する場合の送受信の流れ

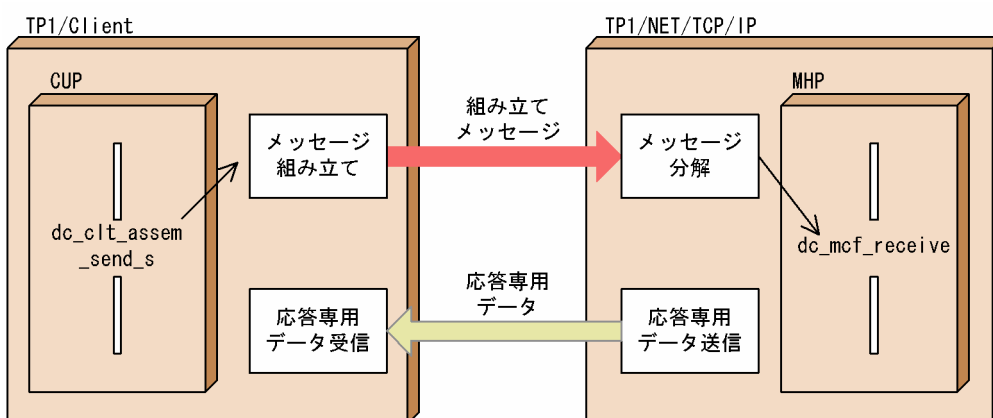
メッセージの送達確認機能を使用する場合のメッセージの組み立て、分解、応答専用データの送受信の流れについて説明します。

(a) メッセージの送信と送達確認

メッセージの送達確認機能を使用してメッセージを送信する場合、`dc_clt_assem_send_s` 関数を発行します。メッセージの送信後に、TP1/Client は TP1/NET/TCP/IP の応答専用データを待ち合わせます。TP1/Client は、TP1/NET/TCP/IP から応答専用データを受信したあと、CUP に制御を戻します。なお、受信した応答専用データは、CUP に通知しません。

メッセージの送達確認機能を使用するときの送信の流れを次の図に示します。

図 2-27 メッセージの送達確認機能使用時の送信の流れ



メッセージの送達確認機能を使用すると、メッセージの送信から応答専用データの受信までを監視できます。監視タイマがタイムアウトした場合、TP1/Client は、TP1/NET/TCP/IP との接続を解放し、DCCLTER_TIMED_OUT を返します。

メッセージの送信および応答専用データは、同一接続を使用して受信します。応答専用データを受信してすぐに接続を解放したい場合は、dc_clt_assem_send_s 関数の引数 flags に DCCLT_SND_CLOSE を指定します。DCNOFLAGS を指定した場合、dc_rpc_close_s 関数を発行するまで、接続は解放されません。ただし、障害時を除きます。

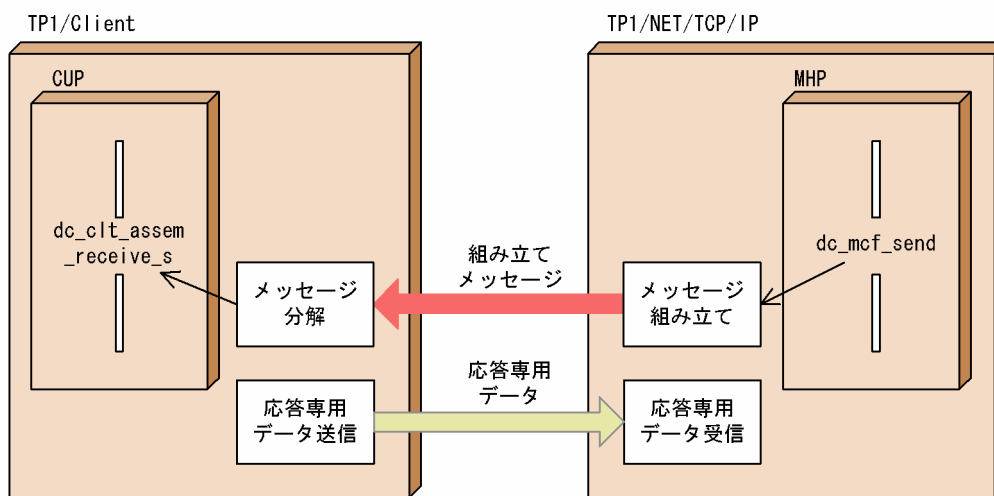
通信先となる TP1/NET/TCP/IP のホスト名はクライアント環境定義 DCSNDHOST に、ポート番号はクライアント環境定義の DCSNDPORT に指定します。ホスト名とポート番号は、dc_clt_assem_send_s 関数の引数に指定することもできます。

(b) メッセージの受信と送達確認

メッセージの送達確認機能を使用してメッセージを受信する場合、dc_clt_assem_receive_s 関数を発行します。TP1/NET/TCP/IP からメッセージを受信したあと、TP1/NET/TCP/IP に応答専用データを送信し、CUP に制御を戻します。

メッセージの送達確認機能を使用するときの受信の流れを次の図に示します。

図 2-28 メッセージの送達確認機能使用時の受信の流れ



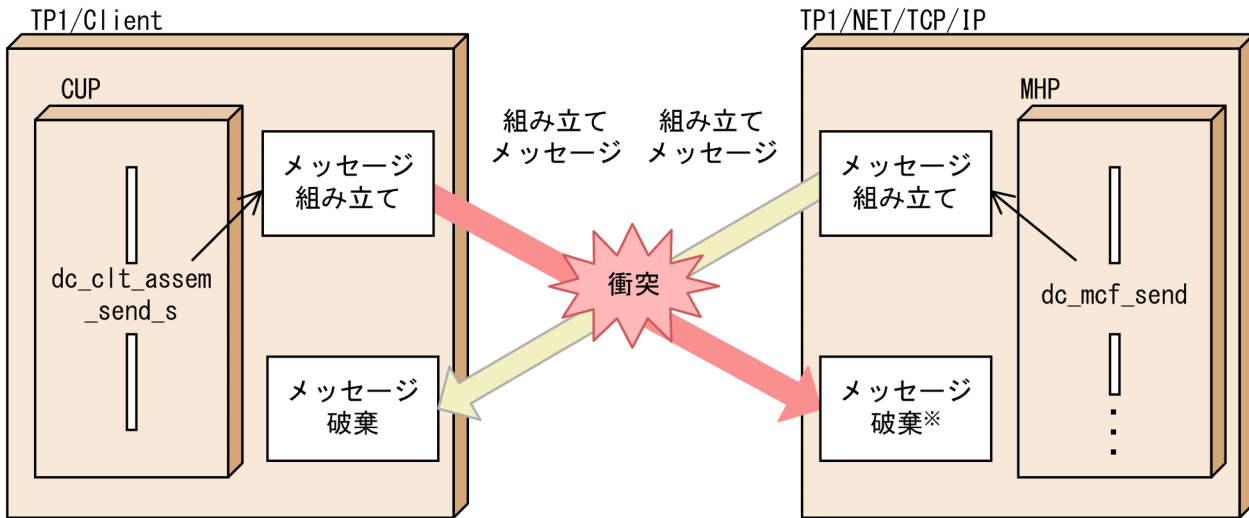
メッセージの受信および応答専用データは、同一接続を使用して送信します。応答専用データを送信してすぐに接続を解放したい場合は、dc_clt_assem_receive_s 関数の引数 flags に DCCLT_RCV_CLOSE を指定します。DCNOFLAGS を指定した場合、dc_rpc_close_s 関数を発行するまで、接続は解放されません。ただし、障害時を除きます。

(c) 送受信メッセージの衝突

TP1/Client から送信したメッセージと、TP1/NET/TCP/IP から送信したメッセージが衝突した場合、TP1/Client は受信したメッセージを破棄し、TP1/NET/TCP/IP との接続を解放したあと、DCCLTER_COLLISION_MESSAGE を返します。

送受信メッセージが衝突した場合の、処理の流れを次の図に示します。

図 2-29 送受信メッセージの衝突時の処理の流れ



注※ 送信メッセージが衝突したあとのTP1/NET/TCP/IPの動作は、設定によって異なります。

(5) 妥当性チェック

メッセージの組み立て機能、および送達確認機能を使用する場合、送受信するメッセージの妥当性が TP1/Client によって自動的にチェックされます。

(a) メッセージ長の妥当性チェック (メッセージの組み立て機能)

メッセージの組み立て機能を使用する場合、TP1/Client はメッセージを受信後に、メッセージ長の妥当性チェックをします。発生するエラー別の対処を次の表に示します。

表 2-4 メッセージ長の妥当性チェックで発生するエラーと対処

項番	エラー	システムの処理	対処
1	不正なメッセージを受信 (メッセージ長が 0~4)	処理を中断	相手システムの設定を見直す
2	受信バッファ不足が発生 (メッセージ長-4 が引数 <code>recvleng</code> よりも大きい)	処理を中断	<code>dc_clt_assem_receive_s</code> 関数の引数 <code>recvleng</code> に指定した値が適切かどうか、または MHP を見直す

(b) 応答専用データの妥当性チェック (メッセージの送達確認機能)

メッセージの送達確認機能を使用する場合、TP1/Client はメッセージの送信後に応答専用データを受信し、妥当性をチェックします。発生するエラー別の対処を次の表に示します。

表 2-5 応答専用データの妥当性チェックで発生するエラーと対処

項番	エラー	システムの処理	対処
1	次に示す不正なメッセージを受信 <ul style="list-style-type: none"> メッセージ長が 0~10 メッセージ長が 11 で、セグメント情報が 0x10 以外 セグメント情報が 0x10, 0x18 以外 	処理を中断	相手システムの設定を見直す
2	メッセージが衝突	処理を中断	必要に応じてリトライ
3	メッセージ ID が不一致	受信リトライ	特になし

(c) 受信メッセージの妥当性チェック (メッセージの送達確認機能)

メッセージの送達確認機能を使用する場合、TP1/Client はメッセージを受信後に、妥当性をチェックします。エラーが発生した場合、TP1/Client は処理を中断します。発生するエラー別の対処を次の表に示します。

表 2-6 受信メッセージの妥当性チェックで発生するエラーと対処

項番	エラー	システムの処理	対処
1	次に示す不正なメッセージを受信 <ul style="list-style-type: none"> メッセージ長が 0~11 セグメント情報が 0x10, 0x18 以外 	処理を中断	相手システムの設定を見直す
2	受信バッファ不足が発生 (メッセージ長-11 が引数 recvleng よりも大きい)	処理を中断	dc_clt_assem_receive_s 関数の引数 recvleng に指定した値が適切かどうか、または MHP を見直す

2.5.5 ユースケースごとの設定方法とポートの割り当て

TCP/IP 通信機能の主なユースケースを表 2-7 に示し、それぞれのユースケースの設定方法とポートの割り当てについて説明します。ユースケースごとの設定方法と使用するポート番号を表 2-8 に示します。

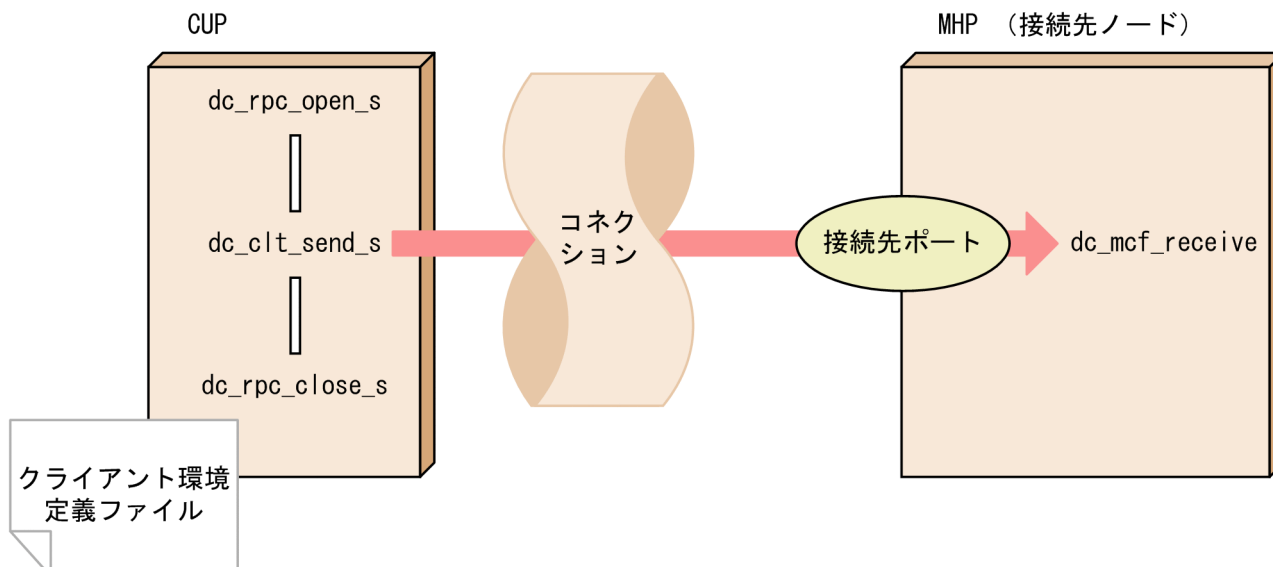
表 2-7 TCP/IP 通信機能のユースケース

項番	ユースケース
1	MHP がサーバ型でメッセージの一方送信を行う場合
2	MHP がクライアント型でメッセージの一方受信を行う場合
3	MHP がサーバ型でメッセージの送受信を一つの接続で行う場合
4	MHP がクライアント型でメッセージの送受信を一つの接続で行う場合

(1) MHP がサーバ型でメッセージの一方送信を行う場合

メッセージの一方送信を次の図に示します。

図 2-30 MHP がサーバ型でメッセージの一方送信を行う場合



メッセージの一方送信は、CUP から接続先のポートに対してコネクションを確立しメッセージの送信を行います。

メッセージの一方送信を行う場合は、次の設定をしてください。その他の設定は条件に応じて設定してください。

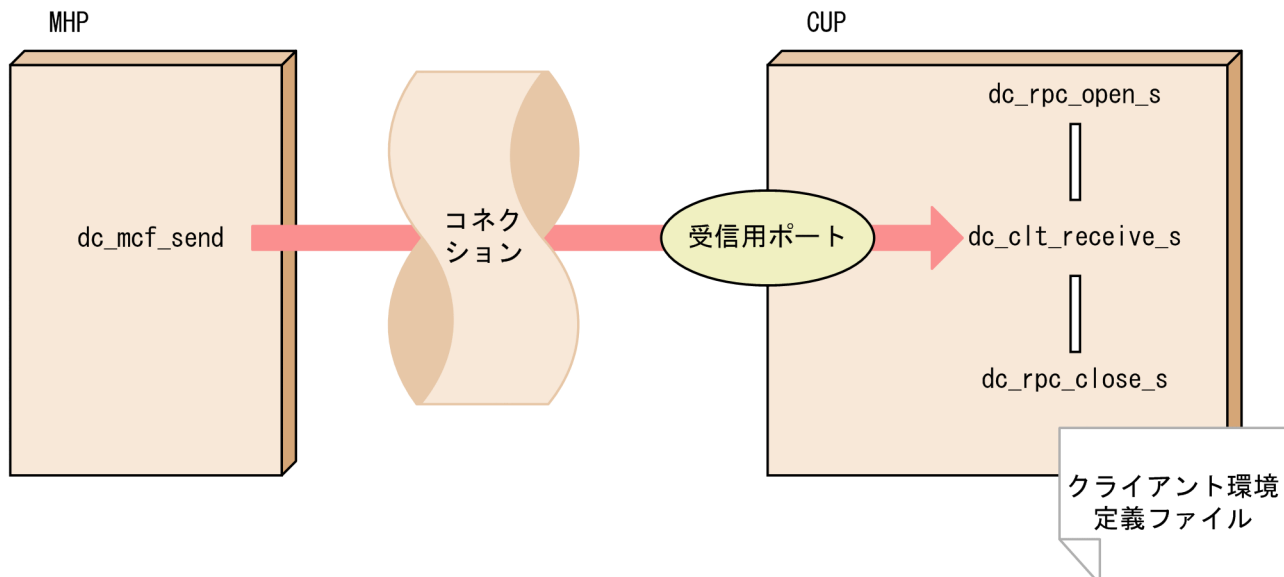
- 接続先のノード名を次のどちらかの方法で指定します。
 - (a) `dc_clt_send_s` 関数、または `dc_clt_assem_send_s` 関数の引数 `hostname` に指定
 - (b) クライアント環境定義 `DCSNDHOST` に指定
- 接続先のポート番号を次のどちらかの方法で指定します。
 - (a) `dc_clt_send_s` 関数、または `dc_clt_assem_send_s` 関数の引数 `portnum` に指定
 - (b) クライアント環境定義 `DCSNDPORT` に指定
- `dc_rpc_open_s` 関数の引数 `flags` に `DCCLT_ONEWAY_SND` を指定します。

接続先のノード名、またはポート番号が CUP ごとに異なる場合は、CUP ごとに設定を変更してください。CUP ごとのクライアント環境定義の設定方法については、「[8.2.3 TP1/Client/W の設定方法](#)」、および「[8.2.4 TP1/Client/P の設定方法](#)」を参照してください。

(2) MHP がクライアント型でメッセージの一方受信を行う場合

メッセージの一方受信を次の図に示します。

図 2-31 MHP がクライアント型でメッセージの一方受信を行う場合



メッセージの一方受信は、MHP から CUP に割り当てられた受信用ポートに対してコネクションを確立し、メッセージの送信を行います。

メッセージの一方受信を行う場合は、次の設定をしてください。その他の設定は条件に応じて設定してください。

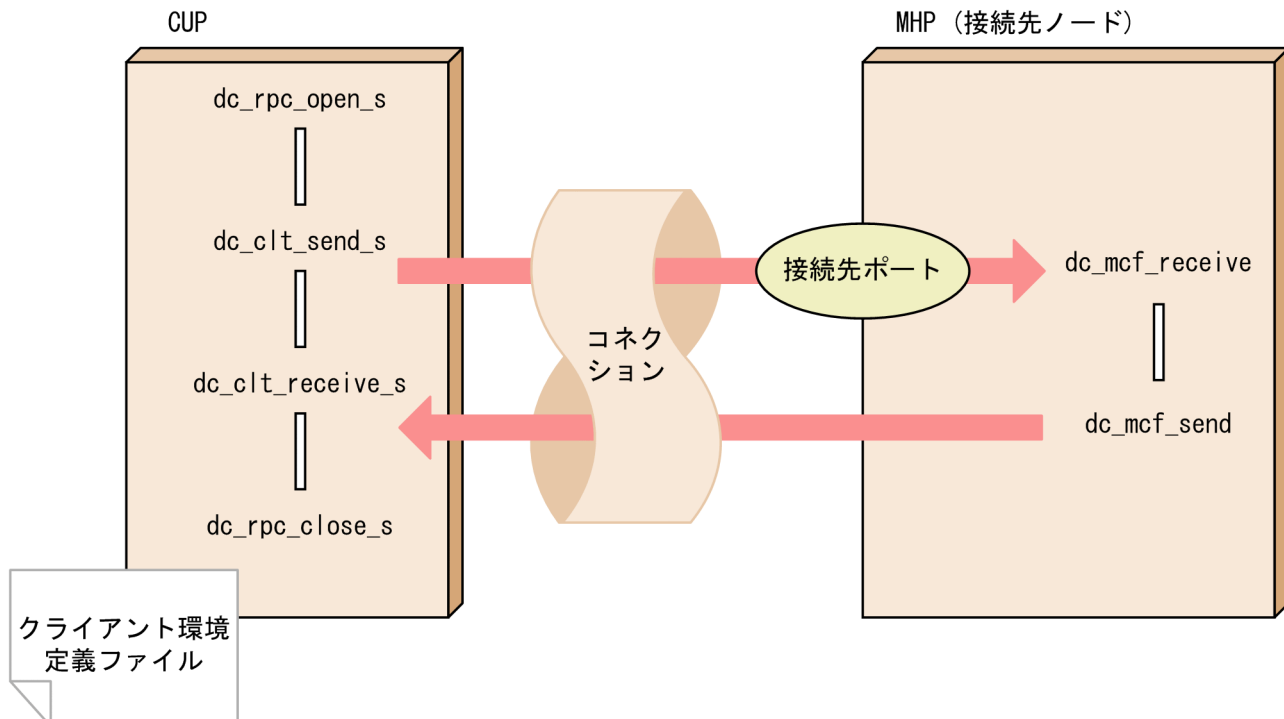
- クライアント環境定義 DCRCVPORT に、CUP の受信用ポート番号（MCF 通信構成定義の定義コマンド `mcftalccn` の `oportno` で指定したポート番号）を指定します。
- `dc_rpc_open_s` 関数の引数 `flags` に `DCCLT_ONEWAY_RCV` を指定します。

マルチスレッドで動作する CUP を含め、同一マシンで CUP を複数実行する場合は、クライアント環境定義 DCRCVPORT を CUP ごと（スレッドごと）に異なるポート番号となるように設定してください。CUP ごとのクライアント環境定義の設定方法については、「[8.2.3 TP1/Client/W の設定方法](#)」、および「[8.2.4 TP1/Client/P の設定方法](#)」を参照してください。

(3) MHP がサーバ型でメッセージの送受信を一つのコネクションで行う場合

MHP がサーバ型でメッセージの送受信を一つのコネクションで行う場合を次の図に示します。

図 2-32 MHP がサーバ型でメッセージの送受信を一つの接続で行う場合



MHP がサーバ型でメッセージの送受信を一つの接続で行う場合，CUP から接続先のポートに対してコネクションを確立します。そのコネクションを使用してメッセージを送受信するため，CUP では受信用のポートを使用しません。そのため，クライアント環境定義 `DCRCVPORT` オペランドを設定する必要はありません。

MHP がサーバ型でメッセージの送受信を一つの接続で行う場合は，次の設定をしてください。その他の設定は条件に応じて設定してください。

- 接続先のノード名を次のどちらかの方法で指定します。
 - (a) `dc_clt_send_s` 関数，または `dc_clt_assem_send_s` 関数の引数 `hostname` に指定
 - (b) クライアント環境定義 `DCSNDHOST` に指定
- 接続先のポート番号を次のどちらかの方法で指定します。
 - (a) `dc_clt_send_s` 関数，または `dc_clt_assem_send_s` 関数の引数 `portnum` に指定
 - (b) クライアント環境定義 `DCSNDPORT` に指定
- クライアント環境定義 `DCSOCKOPENATRCV` オペランドに `Y` を指定します。
- `dc_rpc_open_s` 関数の引数 `flags` に `DCCLT_SNDRCV` を指定します。
- `dc_clt_send_s` 関数，または `dc_clt_assem_send_s` 関数の引数 `flags` に `DCNOFLAGS` を指定します。

接続先のノード名，またはポート番号が CUP ごとに異なる場合は，CUP ごとに設定を変更してください。CUP ごとのクライアント環境定義の設定方法については，「[8.2.3 TP1/Client/W の設定方法](#)」，および「[8.2.4 TP1/Client/P の設定方法](#)」を参照してください。

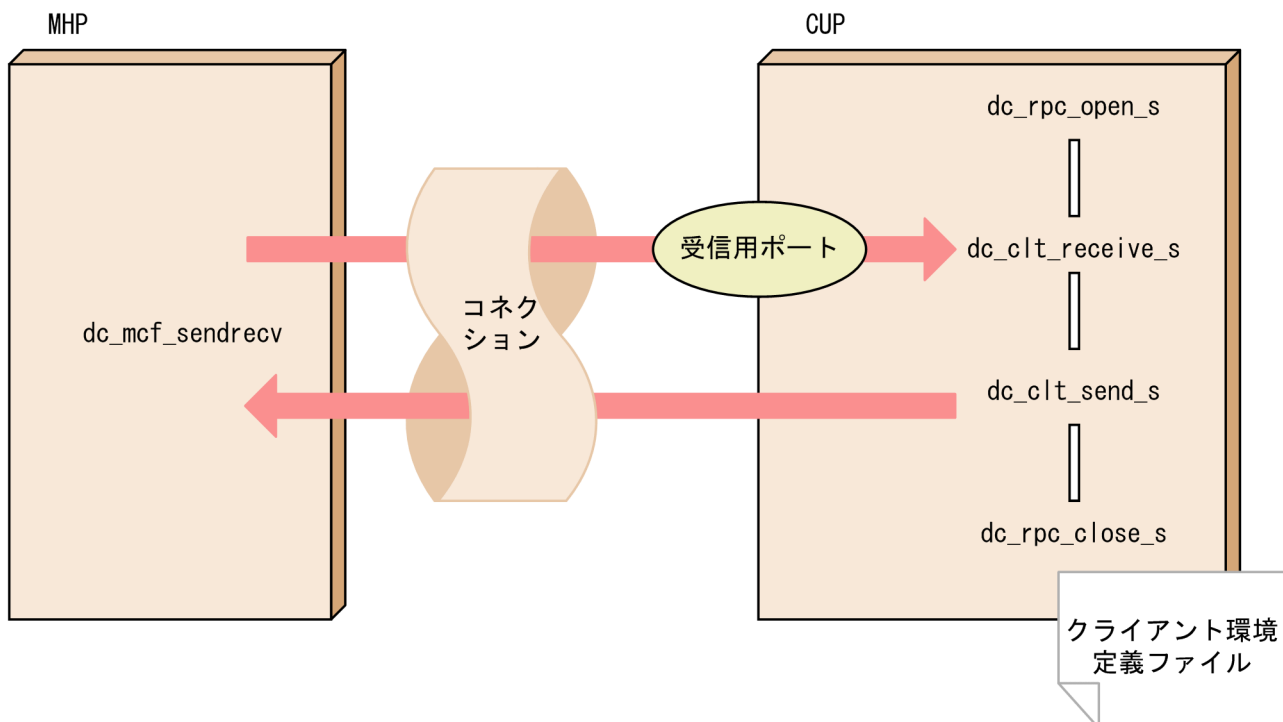
注意事項

何らかの障害が発生し接続が切断された場合、`dc_clt_receive_s` 関数、`dc_clt_receive2_s` 関数、および `dc_clt_assem_receive_s` 関数を実行しないでください。接続の切断を検知した場合は、`dc_clt_send_s` 関数、または `dc_clt_assem_send_s` 関数を再実行し、接続を確立してからメッセージを受信してください。

(4) MHP がクライアント型でメッセージの送受信を一つの接続で行う場合

MHP がクライアント型でメッセージの送受信を一つの接続で行う場合を次の図に示します。

図 2-33 MHP がクライアント型でメッセージの送受信を一つの接続で行う場合



MHP がクライアント型でメッセージの送受信を一つの接続で行う場合、MHP から CUP に割り当てられた受信ポートに対して接続を確立します。その接続を使用してメッセージを送受信します。

MHP がクライアント型でメッセージの送受信を一つの接続で行う場合は、次の設定をしてください。その他の設定は条件に応じて設定してください。

- クライアント環境定義 `DCRCVPORT` に、CUP の受信ポート番号 (MCF 通信構成定義の定義コマンド `mcftalccn` の `oportno` で指定したポート番号) を指定します。
- クライアント環境定義 `DCSOCKOPENATRCV` オペランドに `N` を指定、または指定を省略します。
- `dc_rpc_open_s` 関数の引数 `flags` に `DCCLT_SNDRCV` を指定します。

- dc_clt_receive_s 関数, dc_clt_receive2_s 関数, または dc_clt_assem_receive_s 関数の引数 flags に DCNOFLAGS を指定します。

マルチスレッドで動作する CUP を含め, 同一マシンで CUP を複数実行する場合は, クライアント環境定義 DCRCVPORT を CUP ごと (スレッドごと) に異なるポート番号となるように設定してください。CUP ごとのクライアント環境定義の設定方法については, 「8.2.3 TP1/Client/W の設定方法」, および 「8.2.4 TP1/Client/P の設定方法」 を参照してください。

表 2-8 ユースケースごとの設定方法と使用するポート番号

項番	ユースケース	クライアント環境定義のオペランド				提供 API の使用方法		CUP の使用ポート番号	
		DCSOC KOPENA TRCV	DCSND HOST*1	DCSNDP ORT*1	DCRCVP ORT	dc_rpc_ open_s 関数のフ ラグ	発行順序 *2	送信	受信
1	MHP がサーバ型でメッセージの一方送信を行う場合 詳細については, 「2.5.5(1) MHP がサーバ型でメッセージの一方送信を行う場合」を参照してください。	—	接続先の ノード名	接続先の ポート 番号	—	DCCLT_ ONEWA Y_SND	dc_clt_se nd_s	OS 自動 割り当て	使用し ない
2	MHP がクライアント型でメッセージの一方受信を行う場合 詳細については, 「2.5.5(2) MHP がクライアント型でメッセージの一方受信を行う場合」を参照してください。	—	—	—	CUP の ポート 番号	DCCLT_ ONEWA Y_RCV	dc_clt_re ceive_s	使用し ない	DCRCV PORT の 値
3	MHP がサーバ型で送受信を一本のコネクションで行う場合 詳細については, 「2.5.5(3) MHP がサーバ型でメッセージの送受信を一つのコネクション	Y*3	接続先の ノード名	接続先の ポート 番号	—	DCCLT_ SNDRC V	dc_clt_se nd_s*5 → dc_clt_re ceive_s	OS 自動 割り当て	使用し ない

項番	ユースケース	クライアント環境定義のオペランド				提供 API の使用方法		CUP の使用ポート番号	
		DCSOC KOPENA TRCV	DCSND HOST*1	DCSNDP ORT*1	DCRCVP ORT	dc_rpc_ open_s 関数のフ ラグ	発行順序 *2	送信	受信
3	「で行う場合」を参照してください。	Y*3	接続先の ノード名	接続先の ポート 番号	—	DCCLT_ SNDRC V	dc_clt_se nd_s*5 → dc_clt_re ceive_s	OS 自動 割り当て	使用し ない
4	MHP がクライアント型で送受信を一本の接続で行う場合詳細については、 「2.5.5(4) MHP がクライアント型でメッセージの送受信を一つの接続で行う場合」を参照してください。	N, また は省略*4	—	—	CUP の ポート 番号		dc_clt_re ceive_s* 6 → dc_clt_se nd_s	使用し ない	DCRCV PORT の 値

注※1

接続先ノード名、およびポート番号は、それぞれ dc_clt_send_s 関数、および dc_clt_assem_send_s 関数の引数 hostname (ノード名)、portnum (ポート番号) に指定できます。

注※2

表中の dc_clt_send_s 関数は、dc_clt_assem_send_s 関数を含みます。また、dc_clt_receive_s 関数は、dc_clt_receive2_s 関数、および dc_clt_assem_receive_s 関数を含みます。

注※3

DCSOCKOPENATRCV オペランドに N を指定する設定は、受信用のポートを余分に割り当てるため推奨しません。

注※4

DCSOCKOPENATRCV オペランドに Y を指定する設定は、MHP との接続確立の同期をユーザ処理で実装する必要があるため推奨しません。

注※5

dc_clt_send_s() の引数 flags には DCNOFLAGS を指定してください。

注※6

dc_clt_receive_s() の引数 flags には DCNOFLAGS を指定してください。

2.5.6 TCP/IP 通信機能を使用するときの注意事項

TCP/IP 通信機能を使用するときの注意事項について説明します。

(1) メッセージ送信時の注意事項

(a) 障害発生時のメッセージの消失

次に示す障害が発生すると、TP1/Client では、メッセージが消失したことを検出できません。したがって、ユーザはメッセージ中に通番を付けるなどして、障害に備えてください。ただし、メッセージの送達確認機能を使用する場合は、TP1/Client がメッセージの通番を管理するため、CUP に処理を作り込む必要はありません。

- ソケットのバッファに TP1/Client が送信したメッセージが書き込まれて送信が正常終了した直後に、通信障害が発生したり、コネクションが解放されたりした場合
- TP1/Client が送信したメッセージが MHP の受信バッファに書き込まれる直前に通信障害が発生したり、コネクションが解放されたりした場合

(b) コネクションの確立

TP1/Client がクライアントとなり、MHP にメッセージを送信します。そのため、TP1/Client から MHP に対して、コネクションを確立します。MHP が TP1/NET/TCP/IP を使用している場合、コネクションはサーバ型となります。

(2) メッセージ受信時の注意事項

(a) 障害発生時のメッセージの消失

次に示す障害が発生すると、TP1/Client では、メッセージが消失したことを検出できません。したがって、ユーザはメッセージ中に通番を付けるなどして、障害に備えてください。ただし、メッセージの送達確認機能を使用する場合は、TP1/Client がメッセージの通番を管理するため、CUP に処理を作り込む必要はありません。

- ソケットのバッファに MHP が送信したメッセージが書き込まれて送信が正常終了した直後に、通信障害が発生したり、コネクションが解放されたりした場合
- MHP が送信したメッセージが TP1/Client 側の受信バッファに書き込まれる直前に通信障害が発生したり、コネクションが解放されたりした場合

(b) 受信するメッセージの確認

任意の MHP からのメッセージを受信できます。そのため、コネクションの確立要求を受けると、その要求を無条件に受諾してメッセージを受信します。ユーザは、メッセージ識別子が含まれたヘッダをメッセージ中に含めるなどして、CUP が受け取るメッセージかどうかを確認してください。

(c) メッセージ長

TCP/IP プロトコルを使用してメッセージを受信します。

TCP/IP プロトコルでは、一つのメッセージを複数のパケットに分割したり、複数のメッセージを一つのパケットに詰め込んだりします。そのため、受信したメッセージの切れ目は、ユーザが指定するメッセージ長で判断します。ユーザはメッセージ長を含めた固定長のヘッダを最初に受信し、ヘッダに含まれているメッセージ長を指定して、実際のメッセージを受信してください。ただし、メッセージ組み立て機能、またはメッセージの送達確認機能を使用する場合は、TP1/Client がメッセージ長を管理するため、CUP に処理を作り込む必要はありません。

指定したメッセージ長より短いメッセージを受信した場合、TP1/Client は、メッセージが分割されているものとみなします。そのため、指定した長さ分のメッセージを受信するまで、CUP に制御を戻しません。

(d) コネクションの確立

TP1/Client がサーバとなり、MHP からのメッセージを受信します。そのため、MHP から TP1/Client に対して、コネクションを確立します。MHP が TP1/NET/TCP/IP を使用している場合、コネクションはクライアント型となります。

(3) その他の注意事項

ここでは、TP1/NET/TCP/IP の受信メッセージの組み立て機能、およびメッセージの送達確認機能を使用するときの注意事項について説明します。なお、TP1/Client のメッセージの組み立て機能、およびメッセージの送達確認機能を利用する場合は、CUP を作成するときに、これらの注意事項を意識する必要はありません。

TP1/NET/TCP/IP のプロトコル固有定義の詳細については、マニュアル「OpenTP1 プロトコル TP1/NET/TCP/IP 編」を参照してください。

(a) TP1/NET/TCP/IP の受信メッセージの組み立て機能

TP1/NET/TCP/IP の受信メッセージの組み立て機能を使用すると、MHP が送信したデータの先頭に 4 バイトのメッセージ長が付与されます。また、MHP にデータを送信する場合は、先頭 4 バイトにメッセージ長を設定する必要があります。メッセージ長は、ネットワークバイトオーダーにしてください。なお、このメッセージ長は、MHP が受信するときには削除されます。CUP では、送信時および受信時に、メッセージ長を意識する必要がありますので注意してください。

(b) TP1/NET/TCP/IP の受信メッセージの送達確認機能

TP1/NET/TCP/IP のメッセージの送達確認機能を使用すると、MHP が送信したデータの先頭に 11 バイトのメッセージ情報が付与されます。さらに、受信完了後は、応答専用データを送信します。また、MHP にデータを送信する場合は、先頭 11 バイトにメッセージ情報を設定する必要があります。送信完了後は、応答専用データの到着を待ち合わせます。

メッセージ情報中のメッセージ長は、ネットワークバイトオーダーにしてください。なお、このメッセージ長は、MHP が受信するときには削除されます。CUP では、送信時および受信時に、メッセージ情報を意識する必要がありますので注意してください。

2.6 サーバからの一方通知受信機能

サーバからクライアントへの一方通知メッセージの受信機能について説明します。

2.6.1 サーバからの一方通知受信機能の概要

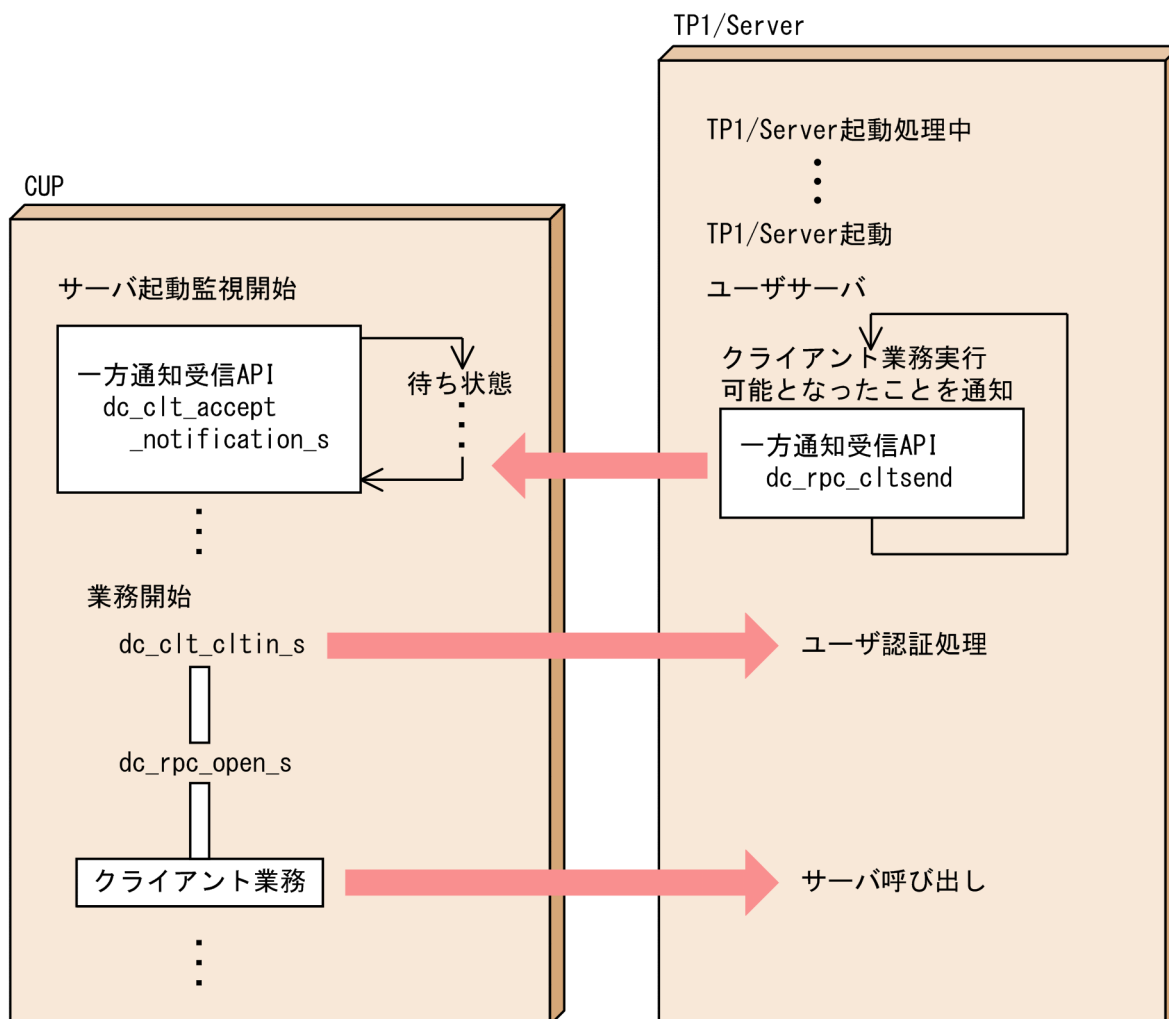
サーバからの一方通知受信機能を使用すると、オンライン開始合図をクライアント側に一斉配布する、従来メインフレームの OLTP での端末一斉起動と同様な運用ができるようになります。

サーバからの一方通知受信機能を使用する場合、`dc_clt_accept_notification_s` 関数を実行します。これによって、クライアント側ではサーバの状態（起動・未起動）に関係なくサーバ側からの送信メッセージを関数に指定した時間中待ち続けます。

サーバ側で起動時にメッセージを送信することによって、クライアント側はサーバ起動を検出し、これを契機にユーザ業務（CUP）を開始できます。

サーバからの一方通知受信機能の処理の流れを次の図に示します。

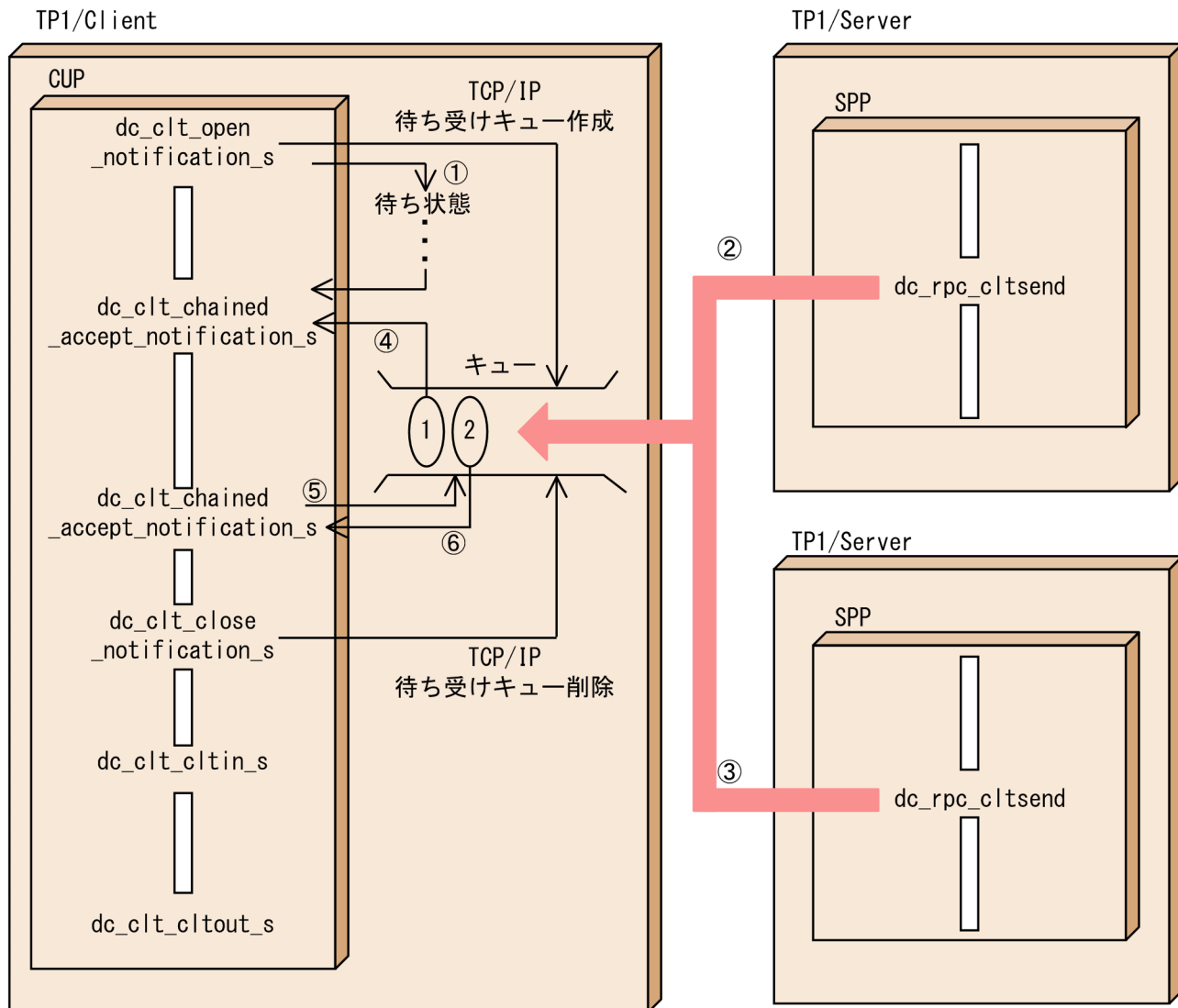
図 2-34 サーバからの一方通知受信機能の処理の流れ



2.6.2 一方通知連続受信機能の概要

一方通知連続受信機能を使用すると、`dc_clt_open_notification_s` 関数を実行してから `dc_clt_close_notification_s` 関数を実行するまでの間は、サーバ側からの一方通知メッセージを連続して受信できます。この機能を利用すると、クライアント側がサーバ側から送信される一方通知メッセージを受信できる状態になっていない場合にサーバ側から一方通知メッセージを送信しても、エラーリターンしません。その場合は、クライアント側が一方通知メッセージを受信する関数を実行した時点で、待ち受けキューからメッセージを取り出すことができます。一方通知連続受信機能の処理の流れを、次の図に示します。

図 2-35 一方通知連続受信機能の処理の流れ



(説明)

- ① 一方通知メッセージが送られてくるのを待ちます。
- ②および③ TP1/Serverが起動し、クライアント業務実行できる状態となったことを、一方通知メッセージを送信してCUPに通知します。サーバからの一方通知メッセージは、TCP/IPの待ち受けキューに格納されます。
- ④ TCP/IPの待ち受けキューに一方通知メッセージが届いたため、一方通知メッセージを取り出して、CUPに制御を戻します。
- ⑤および⑥ `dc_clt_chained_accept_notification_s`関数の発行時に、TCP/IPの待ち受けキューにサーバからの一方通知メッセージが届いていたため、一方通知メッセージを取り出してCUPに制御を戻します。

2.6.3 一方通知連続受信機能を使用するときの注意事項

一方通知連続受信機能を使用するときの注意事項を次に示します。

- クライアント環境定義 DCSELINT に 0 を指定した場合、OS に制御は戻りません。

- TCP/IP の待ち受けキューに保留できるメッセージの数には上限があります。この数は、OS に定義されている上限値に依存します。上限値を超えるメッセージが到着した場合、サーバ側で実行した `dc_rpc_cltsend` 関数は、`DCRPCER_SERVICE_NOT_UP` でエラーリターンします。
- TP1/Client/P の場合、使用中のポート番号を指定してもエラーが検出できないという問題があります。TP1/Client/P ではポート番号の重複使用をチェックしないので、この機能を使用する際には使用中のポート番号を指定しないよう注意してください。

2.7 XATMI インタフェース機能

RPC では、送受信できるデータの長さに制限があります。画像など、情報量の大きなデータは、RPC で規定されているデータ長を超えてしまう場合があります。TP1/Client では、XATMI インタフェース機能の一つである会話型サービスの通信を使用して、データを送受信できます。

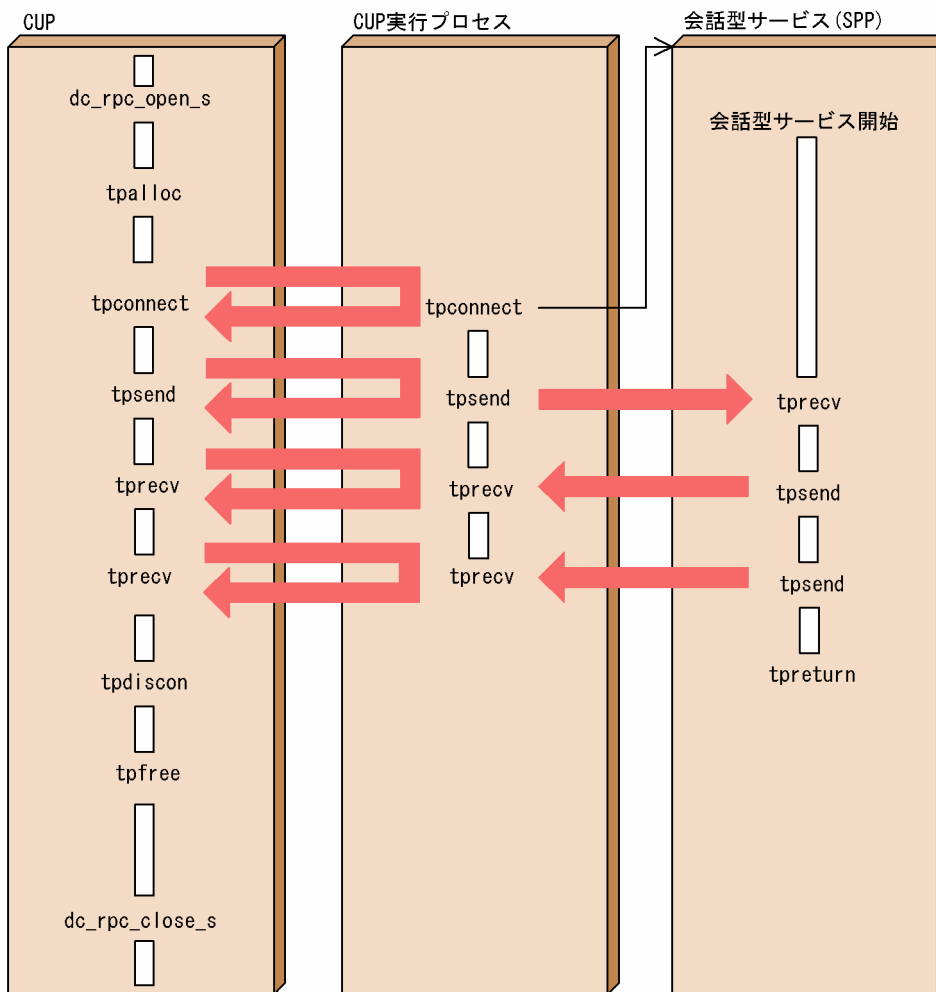
会話型サービスの通信では、大きなデータをパケットと呼ばれるまとまりに分割します。分割されたパケットを数回に分けて送信することで、全体として一つのデータが送信されたこととなります。

会話型サービスの通信を開始する場合、`tpalloc` 関数を呼び出して、型付きのバッファを割り当てます。割り当て完了後、`tpconnect` 関数を呼び出して、通常のメッセージ送受信と同様にコネクションを確立します。メッセージ送受信は、XATMI インタフェース機能の `tprecv` 関数および `tpsend` 関数を呼び出して行います。メッセージ送受信を終了するときは、`tpdiscon` 関数を呼び出してコネクションを切断し、`tpfree` 関数によって型付きのバッファを解放します。会話型サービス時にエラーが発生した場合は、リターン値が返されます。

なお、TP1/Client で使用できる XATMI インタフェース機能は、会話型サービスの通信だけです。

XATMI インタフェース機能の会話型サービスで通信する場合の処理の流れを、次の図に示します。

図 2-36 会話型サービスで通信する場合の処理の流れ



注 tpxxx関数には、末尾に_sの付く関数はありません。

2.7.1 会話型サービス

(1) コネクションの確立

CUP は会話型サービスとのコネクションを確立します。コネクションを確立する場合は、tpconnect 関数呼び出します。tpconnect 関数でコネクションを確立した UAP プロセスをオリジネータ、相手の UAP プロセスをサブオーディネータと呼びます。

コネクションを確立する OpenTP1 ノードはユーザ認証の有無によって次のようになります。ユーザ認証機能については、「2.1 ユーザ認証機能」を参照してください。

ユーザ認証を行う場合

ユーザ認証に成功した OpenTP1 ノードです。

ユーザ認証を抑止する場合

dc_clt_cltin_s 関数の引数 target_host に指定した OpenTP1 ノード、またはクライアント環境定義 DCHOST に指定した OpenTP1 ノードです。両方指定した場合、dc_clt_cltin_s 関数の引数 target_host に指定したノードが優先されます。

なお、ユーザ認証を抑止し、dc_clt_cltin_s 関数の引数 target_host に NULL を指定、かつクライアント環境定義 DCHOST の指定を省略した場合、常設コネクションの確立に失敗します。

tpconnect 関数が正常終了した場合には、確立したコネクションを識別する記述子が返されます。この記述子を、通信する関数に設定して通信します。

会話型サービスで tpreturn 関数を呼び出して処理を終了すると、コネクションは切断されます。

tpconnect 関数では、引数 flags に、制御権を持つかどうかを設定します。制御権ありで設定したプロセスは、tpsend 関数を呼び出し、データを送信できます。制御権なしで設定すると、通信相手のプロセスに制御権が渡され、tpconnect 関数を呼び出したプロセスは tprecv 関数を呼び出し、データを受信できます。

(2) データの送信

データを送信するときは、tpsend 関数を呼び出します。tpsend 関数を呼び出す場合は、引数には、tpconnect 関数で返された記述子を設定して、使用するコネクションを特定します。

tpsend 関数は、コネクションの制御権を持つ側だけが発行できます。制御権を持たない側が tpsend 関数を呼び出した場合、tpsend 関数はエラーリターンします。

コネクションの制御権を通信相手のプロセスに渡したい場合は、tpsend 関数の引数で設定します。

(3) データの受信

データを受信するときは、tprecv 関数を呼び出します。データは非同期に受信します。tprecv 関数はコネクションの制御権を持たないプロセスだけ発行できます。

引数の設定によっては、tprecv 関数はデータを受信するまで待ちます。

tprecv 関数を呼び出したプロセスがトランザクション下にある場合、最大応答待ち時間は、クライアント環境定義 DCCLTTREXPTM またはクライアントサービス定義 trn_expiration_time で指定した値となります。この場合、最大応答待ち時間を超えると、CUP 実行プロセスは異常終了します。ただし、tprecv 関数はエラーリターンしません。

tprecv 関数を呼び出したプロセスがトランザクション下でない場合、最大応答待ち時間は、クライアントサービス定義の watch_time で指定した値となります。この場合、最大応答待ち時間を超えると、tprecv 関数はエラーリターンします。

(4) コネクションの切断

会話型サービスの処理が終了して tpreturn 関数を呼び出すと、コネクションは正常に切断されます。また、通信中にエラーが発生した場合、コネクションが切断されることがあります。

(5) コネクションの強制切断

何らかの理由でコネクションを強制的に切断する場合は、`tpdiscon` 関数を呼び出します。`tpdiscon` 関数に設定された記述子は、それ以降の処理では無効になります。トランザクションはロールバックされます。

(6) トランザクションの生成

XATMI インタフェースを使用して通信を行う CUP で、CUP からトランザクションを生成する場合は、`dc_trn_begin_s` 関数を呼び出します。

トランザクションを生成するタイミングを次の表に示します。

表 2-9 トランザクションを生成するタイミング

dc_clt_connect_s 関数発行済み		dc_clt_connect_s 関数未発行	
tpconnect 関数発行済み	tpconnect 関数未発行	tpconnect 関数発行済み	tpconnect 関数未発行
○	○	○	×

(凡例)

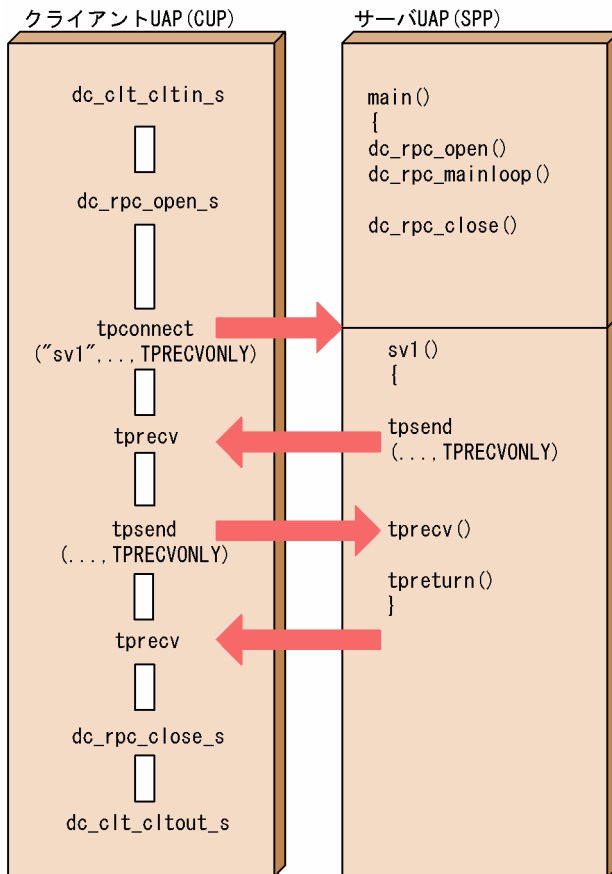
- ：トランザクションを生成し、XATMI インタフェースによる会話型サービスとの通信ができます。
- ×：トランザクションは生成できますが、XATMI インタフェースによる会話型サービスとの通信はできません。

(7) OpenTP1 の dc_rpc_call_s 関数との併用

`tpconnect` 関数を呼び出して会話型サービスとのコネクションを確立したあとも `dc_rpc_call_s` 関数は発行できます。

会話型サービスの通信形態を次の図に示します。

図 2-37 会話型サービスの通信形態



注 tpxxxx関数には、末尾に_sの付く関数はありません。

2.7.2 会話型サービスの時間監視

会話型サービスの通信では、時間監視はすべて OpenTP1 の定義で指定した値に従います。

会話型サービス時のタイムアウトは、次に示す時間を超えた場合に発生します。

- 最大応答待ち時間
- 常設コネクション問い合わせ間隔最大時間
- トランザクションブランチ限界経過時間

(1) 最大応答待ち時間のタイムアウト

最大応答待ち時間は、CUP および CUP 実行プロセスによって検出されます。CUP はクライアント環境定義 DCWATCHTIM で設定し、CUP 実行プロセスはクライアントサービス定義の `watch_time` で設定します。

システムの整合性を保つために、クライアント環境定義 DCWATCHTIM はクライアントサービス定義の `watch_time` よりも大きな値を指定する必要があります。CUP が CUP 実行プロセスよりも先にタイムア

ウトを検出した場合、`dc_rpc_close_s` 関数を呼び出さないかぎり、それ以降の通信はできません。この場合、CUP 実行プロセスは、問い合わせ間隔最大時間のタイムアウト後に該当するトランザクションを強制的にロールバック（トランザクション中だけ）し、CUP とのコネクションを切断します。

(2) 常設コネクション問い合わせ間隔最大時間のタイムアウト

常設コネクション問い合わせ間隔最大時間は、CUP 実行プロセスによって検出されます。時間監視の値は、クライアント環境定義 `DCCLTINQUIRETIME`、またはクライアントサービス定義の `clt_inquire_time` で設定します。定義の優先順位は次のとおりです。

クライアント環境定義>クライアントサービス定義

常設コネクション問い合わせ間隔最大時間とは、CUP が CUP 実行プロセスに対して行う要求（`tpsend` 関数など）と要求の間隔の最大時間です。

常設コネクション問い合わせ間隔最大時間のタイムアウトの場合、CUP 実行プロセスは該当するトランザクションを強制的にロールバック（トランザクション中だけ）し、CUP とのコネクションを切断します。

(3) トランザクションブランチ限界経過時間のタイムアウト

トランザクションブランチ限界経過時間は、CUP 実行プロセスによって検出されます。クライアント環境定義 `DCCLTTREXPTM`、またはクライアントサービス定義の `trn_expiration_time` で設定します。定義の優先順位は次のとおりです。

クライアント環境定義>クライアントサービス定義

トランザクションブランチ限界経過時間とは、トランザクションが生成されてから同期点の取得を行うまでの時間をいいます。

トランザクションブランチ限界経過時間のタイムアウトの場合は、CUP 実行プロセスが異常終了します。このときすべてのコネクションが強制的に切断され、トランザクションはロールバックされます。

また、設定によっては、トランザクションブランチ限界経過時間以外のタイムアウトを検出しないで、無限に待つこともできます。ただし、トランザクションブランチ限界経過時間満了によるタイムアウトは、定義の設定に関係なく発生します。

2.7.3 イベントの受信

コネクションを識別する記述子の中にイベントが設定されている場合、会話型サービスの通信関数（`tpsend` 関数、`tprecv` 関数）でそのイベントを受信できます。イベントには、データの送受信に関する情報が設定されます。

イベントの詳細については「[3.1 関数インタフェース](#)」を参照してください。

2.7.4 通信データの型

TP1/Client の会話型サービスで使用できる通信データの型は X_OCTET です。

X_OCTET は、内容と操作が完全にアプリケーションによって定義されているバイト（文字）の配列です。そのため、通信管理がどれくらいの長さの文字配列を送るかがわかるように、X_OCTET では、長さを示すパラメタを必ず設定します。ここで設定されるデータの長さは、会話型サービスで入力時 len パラメタとして提供され、出力時 len パラメタとして回収されます。ここで設定される型付きのバッファは、異機種マシン同士でデータを送受信するときも、データの符号化や復号化はしません。また、TP1/Client で使用する型 X_OCTET にはサブタイプはありません。

2.7.5 XATMI インタフェース機能を使用するときの注意事項

TP1/Client で XATMI インタフェースを使用するときの注意事項を示します。

- トランザクション下で XATMI インタフェースを使用する場合は、定義に次の値を必ず指定してください。
DCCLTTREXPTM, または trn_expiration_time に 0 以外の値を設定する
DCCLTTREXPSP, または trn_expiration_time_suspend に Y を設定する
- tpconnect 関数, tpSEND 関数でデータを送信するときにブロッキング状態が起こって、一定時間が経過してもブロッキング状態が解除されなかった場合、ブロッキングするかどうかの設定に関係なく、TPESYSTEM が返されます。
- トランザクションブランチ限界経過時間のタイムアウトが発生した場合、TPETIME は返されず、CUP 実行プロセスが異常終了します。このとき、以前に確立されていたコネクションはすべて切断され使用できなくなります。
- 常設コネクション問い合わせ間隔最大時間のタイムアウトが発生した場合、以前に確立されていたコネクションはすべて切断され使用できなくなります。
- CUP と CUP 実行プロセス間のコネクションで通信障害などが発生した場合、CUP 実行プロセスが異常終了します。このとき、以前に確立されていたコネクションはすべて切断され使用できなくなります。
- クライアント環境定義 DCCLTXATMI に Y を指定してください。

2.8 文字コード変換機能

この機能は、TP1/Client/P でだけ使用できます。

メインフレームをサーバとして使用する場合、サーバとクライアントでは使用する文字コードの体系が異なる場合があります、その場合は RPC をする時にコード変換が必要となります。コード変換には、コードマッピングテーブルを使用しない場合と使用する場合があります。

コード変換の仕様の詳細については、「付録 A コード変換の仕様」を参照してください。

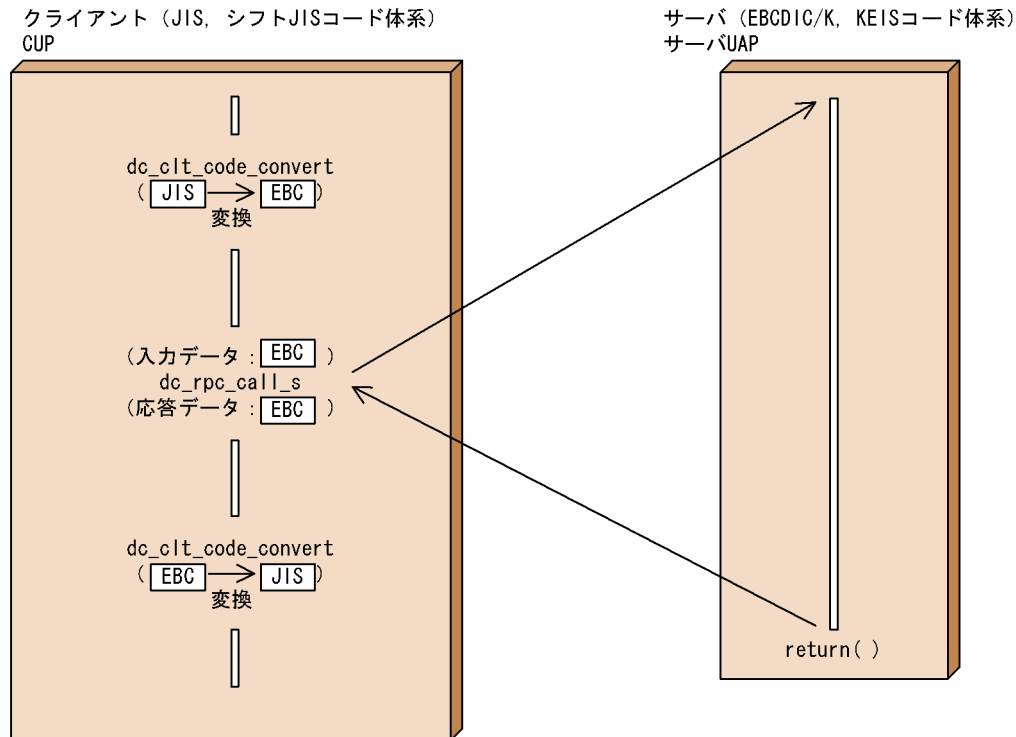
2.8.1 コードマッピングテーブルを使用しない場合

TP1/Client/P (Windows 環境) では、要求コードに DCCLT_JISSJIS_TO_EBCKEIS を指定して `dc_clt_code_convert` 関数を呼び出すと、JIS コードまたはシフト JIS コードで構成される文字列を、EBCDIC/EBCDIK コードまたは KEIS コードで構成される文字列に変換できます。この結果変換された文字列を `dc_rpc_call_s` 関数でサーバに問い合わせます。

`dc_rpc_call_s` 関数実行後の応答メッセージに対し、要求コードに DCCLT_EBCKEIS_TO_JISSJIS を指定して `dc_clt_code_convert` 関数を呼び出すと、EBCDIC/EBCDIK コードまたは KEIS コードで構成される文字列を、JIS コードまたはシフト JIS コードで構成される文字列に変換できます。

文字コード変換機能の概要を次の図に示します。

図 2-38 文字コード変換機能の概要



(凡例) **JIS**: JISコード, またはシフトJISコードで構成される文字列
EBC: EBCDIC/EBCDIKコードまたはKEISコードで構成される文字列

2.8.2 コードマッピングテーブルを使用する場合

コードマッピングテーブルを使用するコード変換では、CommuniNet と連携して、外字コードなどを特定のコードの文字列に変換します。変換の種類は、コードマッピングテーブルを使用しない場合と同様です。

変換対象となるコードがコードマッピングテーブルにない場合は、コードマッピングテーブルを使用しない場合と同様に処理を行います。ただし、不正コードを検出した場合、エラーとなります。

(1) 適用範囲

文字コード変換機能を使用する場合、CommuniNet でのコード変換対象となっている文字コードが変換できます。

(2) 使用方法

この機能を使用する場合、次に示す順序で関数発行、および呼び出しを実行してください。

1. 文字コード変換の開始

```
dc_clt_codeconv_open()
```

2. 文字コード変換の実行（複数回の実行可能）

`dc_clt_codeconv_exec()`

3. 文字コード変換の終了

`dc_clt_codeconv_close()`

文字コード変換の実行（`dc_clt_codeconv_exec()`）は、いったん文字コード変換の開始（`dc_clt_codeconv_open()`）をすれば、文字コード変換の終了（`dc_clt_codeconv_close()`）を行うまで、複数回実行できます。

(3) コードマッピングテーブルを使用するときの注意事項

- この機能を使用する場合は、CommuniNet のコードマッピングテーブルが必要です。CommuniNet のコードマッピングユーティリティで、コードマッピングテーブルを作成してから、この機能を使用してください。
- CommuniNet インストール後、一度も CommuniNet コードマッピングユーティリティで保存されていないコードマッピングテーブルは、使用できません。この機能を使用する前に、CommuniNet コードマッピングユーティリティでコードマッピングテーブルを保存してください。
- CommuniNet のコードマッピングテーブルのファイル名は、必ず `CMAPEX.TBL` とし、この機能を使用する前に Windows のディレクトリ下に格納してください。
- この機能を使用している途中で、CommuniNet のコードマッピングユーティリティによって、コードマッピングテーブルの内容を変更しても、文字コード変換機能の処理には反映されません。
- この機能では、エラーログ、および UAP トレースの情報は取得しません。
- 文字コード変換の開始（`dc_clt_codeconv_open()`または `CBLDCUTL('CNVOPN')`）は、1 回だけ発行して、文字コード変換の実行（`dc_clt_codeconv_exec()`または `CBLDCUTL('CNVEXEC')`）をしてください。メモリ不足となるおそれがあるため、文字コード変換の開始は複数回発行しないでください。複数回発行した場合は、発行した回数分、文字コード変換の終了（`dc_clt_codeconv_close()`または `CBLDCUTL('CNVCLS')`）を発行してください。

2.9 マルチスレッド機能

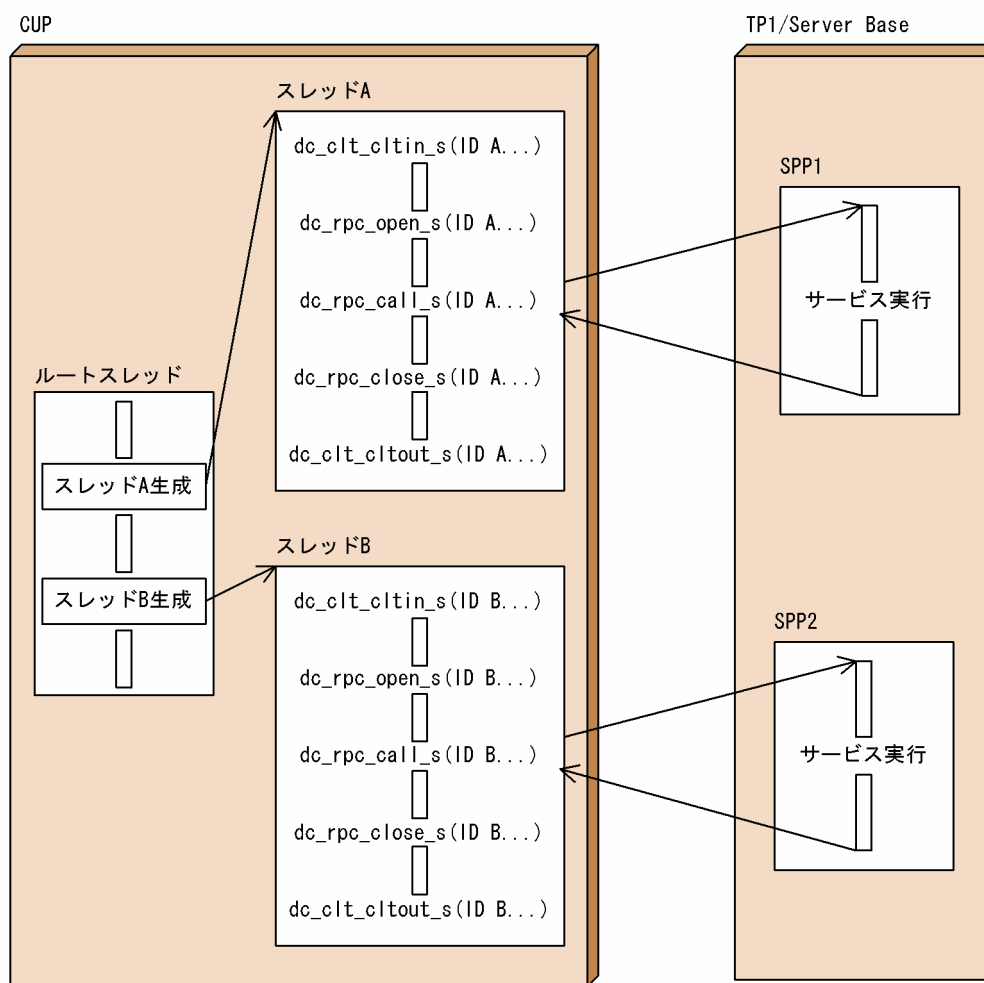
すべてのサービス呼び出しは、シリアルに実行されます。このため、処理に時間が掛かるサービス呼び出しを行うと、その後のサービス呼び出しが待たされてしまいます。

この場合、マルチスレッド機能を使用すると、処理に時間の掛かるサービス呼び出しは専用のスレッドに担当させることができるようになり、別スレッドではほかのサービス呼び出しができるようになります。

2.9.1 マルチスレッドに対応した CUP の処理の概要

マルチスレッド環境で動作する CUP の処理の概要を次の図に示します。

図 2-39 マルチスレッド環境で動作する CUP の処理の概要



図では、CUP のルートスレッドで、スレッド A およびスレッド B を生成します。生成されたスレッド A およびスレッド B が、それぞれ SPP1 および SPP2 に対してサービスを要求します。SPP1 および SPP2 は、それぞれ要求されたサービスを実行して、実行結果をスレッド A およびスレッド B に返します。

2.9.2 マルチスレッドに対応しない関数の実行

マルチスレッド環境で、dc_clt_cltin_s 関数を実行したスレッドとは別のスレッドで dc_rpc_call_s 関数を実行した場合、正常に動作しないことがあります。

マルチスレッド環境に対応しない関数の実行を、次の図に示します。

図 2-40 マルチスレッド環境に対応しない関数の実行 1

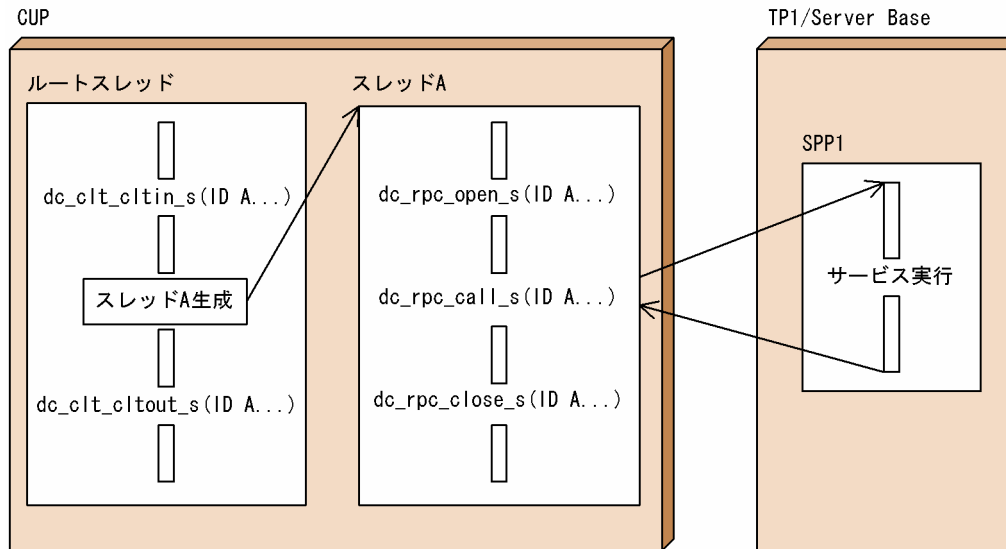
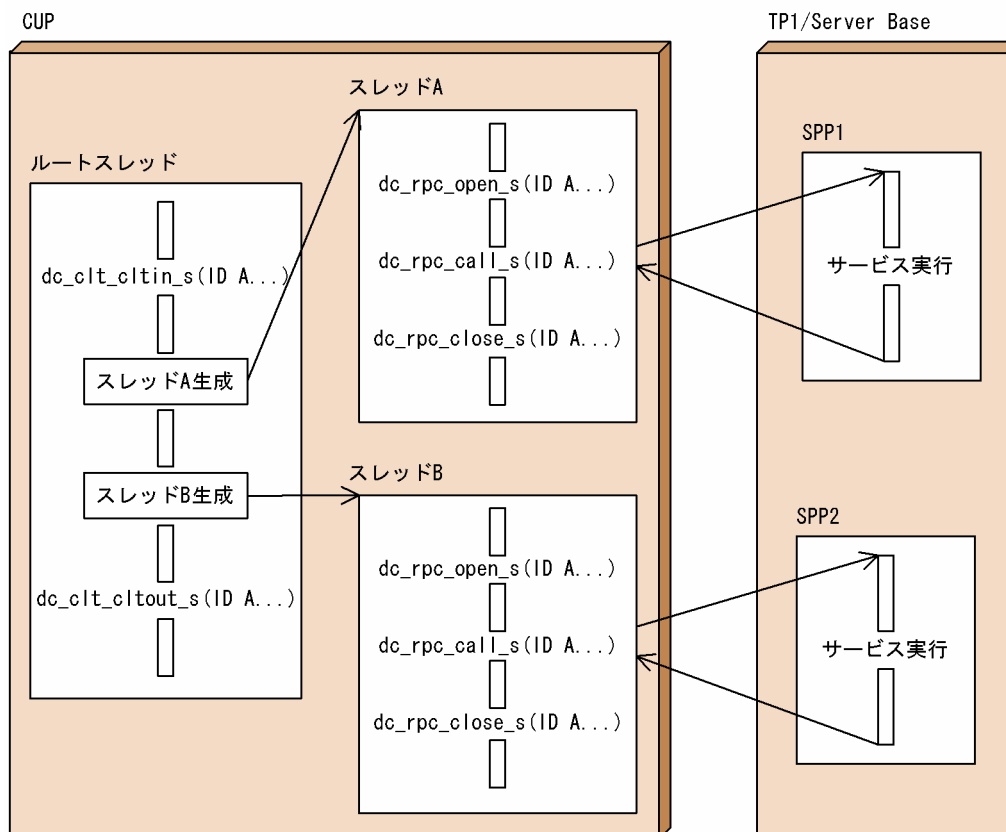


図 2-41 マルチスレッド環境に対応しない関数の実行 2



マルチスレッド環境に対応するためには、次の点に注意して関数を実行してください。

- それぞれのスレッドで、必ず `dc_clt_cltin_s` 関数を実行し、クライアント ID と呼ばれるディスクリプタを受け取る必要があります。各スレッドで実行するすべての関数には、`dc_clt_cltin_s` 関数で受け取ったクライアント ID を指定してください。

任意のプロセスまたはスレッドがユーザ認証を実行して取得したクライアント ID は、ほかのプロセスまたはスレッドでは使用できません。

- マルチスレッド環境で RPC を同時に実行する場合は、各スレッドで `dc_clt_cltin_s` 関数から実行する必要があります。

TP1/Client 提供関数のクライアント ID 引数には、同じスレッドの `dc_clt_cltin_s` 関数が返したクライアント ID を指定してください。クライアント ID は、スレッドをわたって使用できません。

2.9.3 マルチスレッド機能を使用するときの注意事項

- マルチスレッド環境で CUP を動作させる場合、`_s` 付き関数を使用してください。
- スレッド上で TP1/Client の関数の実行中（関数がリターンする前）に、同じスレッドから、再び TP1/Client の関数を実行しないでください。

- スレッドごとにクライアント環境定義の指定を分ける場合は、`dc_clt_cltin_s` 関数の引数 `defpath` に、異なるファイル名を指定してください。

受信ポート固定機能、TCP/IP 通信機能の受信関数、またはサーバからの一方通知受信機能を使用する場合は、スレッドごとに異なるポート番号を指定する必要があります。

- トレースファイルの出力先を同一にした場合、CPU の使用率が高くなり、RPC スループットが劣化するおそれがあります。この場合は、プロセス単位、またはスレッド単位でトレースファイルの出力先を分けて、この現象を回避してください。

- マルチスレッド機能を使用した CUP ではエラーログファイル、および各種トレースファイルが正しく出力されない場合があります。

次のすべての条件に該当する場合は、クライアント環境定義 `DCTRCPATH` に指定するディレクトリをスレッドごとに分けてください。

1. マルチスレッド機能を使用している。
2. 各スレッドでクライアント環境定義 `DCTRCPATH` に同一ディレクトリを指定している、または指定を省略している。
3. 各スレッドで発行している API が同時に動作する。

2.10 オンラインテスト機能

TP1/Server のオンラインテスト機能を使用できます。オンラインテスト機能を使用すると、CUP から起動する SPP はテストモードで実行できます。

オンラインテスト機能を使用する場合、あらかじめ、クライアント環境定義 DCUTOKEY にテストユーザ ID を指定しておく必要があります。また、CUP から起動する SPP は、ユーザサービス定義で test_mode に no 以外を指定しておく必要があります。

なお、TP1/Server のオンラインテスト機能を使用するには、TP1/Online Tester が必要です。

オンラインテスト機能については、マニュアル「OpenTP1 テスタ・UAP トレース使用の手引」を参照してください。

常設コネクション確立機能使用時は、常設コネクション確立先が CUP 実行プロセスのときのトランザクションの範囲内でだけ使用できます。

2.11 トラブルシュート機能

トラブルシュート機能には、エラーログ機能、UAP トレース機能、ソケットトレース機能、およびモジュールトレース機能があります。これらの機能で取得された情報は、ファイルに出力されます。UAP トレース機能、ソケットトレース機能、およびモジュールトレース機能で取得された情報は、バイナリ形式で出力されるため、編集する必要があります。

2.11.1 エラーログ機能

エラーログには、メッセージが出力されます。関数のリターン値だけではエラーの原因を特定できず、エラーを取り除けない場合があります。そのような場合にエラーログを見ると、エラーの原因を特定できるメッセージが出力されていることがあります。

エラーログは、通常カレントディレクトリ（CUP 実行ディレクトリ）に出力されますが、クライアント環境定義 DCTRCPATH を指定すると、DCTRCPATH に指定したディレクトリに出力されます。指定したディレクトリが存在しない場合、エラーログは出力されません。

エラーログ機能を使用するかどうかは、クライアント環境定義 DCTRCERR の指定に従います。DCTRCERR の指定を省略するか、またはエラーログを出力する指定にすると、二つのファイル（dcerr1.trc、dcerr2.trc）が作成されます。なお、出力する情報がない場合、ファイルは作成されません。

二つのファイルは、ラウンドロビン方式で切り替えられて時系列に出力されます。そのため、古い情報はファイル単位で消去されます。また、クライアント環境定義 DCTRCERR に指定したファイルサイズを超える書き込みが発生すると、ファイルは切り替えられます。ただし、書き込みを開始する時点で、指定したファイルサイズを超えていなければ、その情報は出力されます。そのため、実際のファイルサイズは、指定したファイルサイズよりも大きくなる場合があります。

なお、ファイルの先頭に出力される文字列は保守情報です。

2.11.2 UAP トレース機能

UAP トレースには、ユーザが発行した関数の情報が出力されます。UAP トレースを見ると、関数の発行順序、引数の指定値、関数のリターン値などがわかります。

UAP トレースは、通常カレントディレクトリ（CUP 実行ディレクトリ）に出力されますが、クライアント環境定義 DCTRCPATH を指定すると、DCTRCPATH に指定したディレクトリに出力されます。指定したディレクトリが存在しない場合、UAP トレースは出力されません。

UAP トレース機能を使用するかどうかは、クライアント環境定義 DCTRCUAP の指定に従います。DCTRCUAP の指定を、UAP トレースを出力する指定にすると、二つのファイル（dcuap1.trc、dcuap2.trc）が作成されます。なお、出力する情報がない場合、ファイルは作成されません。

二つのファイルは、ラウンドロビン方式で切り替えられて時系列に出力されます。そのため、古い情報はファイル単位で消去されます。また、クライアント環境定義 DCTRCUAP に指定したファイルサイズを超える書き込みが発生すると、ファイルは切り替えられます。ただし、書き込みを開始する時点で指定したファイルサイズを超えていなければ、その情報は出力されます。そのため、実際のファイルサイズは、指定したファイルサイズよりも大きくなる場合があります。

UAP トレースは、バイナリ形式で出力されます。そのため、トレース編集コマンド (cltdump コマンドまたは cltdmp32 コマンド) を使用して、テキスト形式に編集する必要があります。

2.11.3 ソケットトレース機能

TP1/Client は、ソケットトレースとして通信に関する情報をファイルに出力します。なお、ソケットトレースの出力内容は、公開しておりませんが、保守員が障害調査資料として使用場合がありますので、保守員の指示に従ってください。

ソケットトレースは、通常カレントディレクトリ (CUP 実行ディレクトリ) に出力されますが、クライアント環境定義 DCTRCPATH に出力先を指定すれば、指定したディレクトリに出力されます。指定したディレクトリが存在しない場合、ソケットトレースは出力されません。

ソケットトレース機能を使用するかどうかは、クライアント環境定義 DCTRCSOC の指定に従います。ファイルサイズについては、クライアント環境定義 DCTRCSOC の指定に従います。データサイズについては、クライアント環境定義 DCTRCSOCSIZE の指定に従います。ソケットトレースを出力する指定にすると、二つのファイル (dcsoc1.trc, dcsoc2.trc) が作成されます。なお、出力する情報がない場合、ファイルは作成されません。

出力されるファイルの切り替えは、ラウンドロビン方式で行われて、時系列に出力されます。そのため、古い情報はファイル単位で消去されます。また、指定したファイルサイズを超えた書き込みが発生すると、ファイルは切り替わります。書き込みを実行する時点で指定したファイルサイズを超えていなければ、その情報は出力されます。このため、指定したファイルサイズよりも大きくなる場合もあります。

ソケットトレースは、バイナリ形式で出力されます。このため、トレース編集コマンド (cltdump コマンド、または cltdmp32 コマンド) を使用して、テキスト形式に編集する必要があります。

2.11.4 モジュールトレース機能

TP1/Client は、モジュールトレースとして、処理の流れを追うために必要な情報をファイルに出力します。なお、モジュールトレースの出力内容は、公開しておりませんが、保守員が障害調査資料として使用場合がありますので、保守員の指示に従ってください。

モジュールトレースは、通常カレントディレクトリ (CUP 実行ディレクトリ) に出力されますが、クライアント環境定義 DCTRCPATH に出力先を指定すれば、指定した先に出力されます。指定した先のディレクトリが存在しなかった場合、モジュールトレースは出力されません。

モジュールトレース機能を使用するかどうかは、クライアント環境定義 DCTRCMDL の指定に従います。モジュールトレースを出力する指定にすると、二つのファイル (dcmdl1.trc, dcmdl2.trc) が作成されます。出力する情報がない場合、ファイルは作成されません。

出力されるファイルの切り替えは、ラウンドロビン方式で行われて、時系列に出力されます。そのため、古い情報はファイル単位で消去されます。また、指定したファイルサイズを超えた書き込みが発生すると、ファイルは切り替わります。書き込みを実行する時点で指定したファイルサイズを超えていなければ、その情報は出力されます。このため、指定したファイルサイズよりも大きくなる場合もあります。

モジュールトレースは、バイナリ形式で出力されます。このため、トレース編集コマンド (cltdump コマンド、または cltdmp32 コマンド) を使用して、テキスト形式に編集する必要があります。

2.11.5 TP1/Server の性能検証用トレース

TP1/Server の性能検証用トレース (PRF トレース) とは、TP1/Server 上で動作する各種サービスの、主なイベントのトレース情報です。この性能検証用トレースによって、性能検証、およびトラブルシューティングの効率を向上できます。

TP1/Client では、TP1/Server の性能検証用トレースに TP1/Client が設定した性能検証用の識別情報を出力できます。また、この性能検証用の識別情報は、TP1/Client の UAP トレースにも出力されます。このため、TP1/Client の関数実行時間 (UAP トレースで取得) と、TP1/Server のサービスの実行時間 (性能検証用トレースで取得) とを照合したり、障害が発生したときに、処理がどこまで到達したかを知ることができます。

(1) TP1/Server への性能検証用の識別情報の伝播

TP1/Server へ性能検証用の識別情報を伝播する場合は、クライアント環境定義 DCCLTPRFINFOSEND に Y を指定します。

ただし、TP1/Server のバージョンによっては、伝播できない場合があります。詳細については、マニュアル「OpenTP1 運用と操作」、またはマニュアル「TP1/LiNK 使用の手引」を参照してください。

(2) TP1/Server と TP1/Client とのトレースの照合

クライアント環境定義 DCCLTPRFINFOSEND に Y を指定した場合、TP1/Server に送信する電文中に、dc_clt_cltin_s 関数ごとに一意となる識別情報 (IP アドレスなど) を付加できます。付加した情報は、TP1/Client の UAP トレースに出力されます。また、これと同じ情報は TP1/Server の性能検証用トレースにも出力されます。

この TP1/Client の UAP トレースと、TP1/Server の性能検証用トレースとを照合することで、TP1/Client と TP1/Server との間の、一連の処理の流れを知ることができます。

ただし、クライアント環境定義 DCTRCUAP の指定が有効でない場合は、UAP トレースに性能検証用の識別情報は出力されません (TP1/Server へ性能検証用の識別情報は伝播します)。

(3) 性能検証用トレースの識別情報

TP1/Client の性能検証用の識別情報は、TP1/Client の UAP トレース、および TP1/Server の性能検証用トレースに出力されます。

(a) 性能検証用の識別情報として取得する情報

性能検証用の識別情報として取得する情報について、次に示します。

ノード ID : aa[bb] (4 バイトの英数字)

aa : 次の値が出力されます。

- TP1/Client/W の場合 : _W
- TP1/Client/P の場合 : _P

bb : ランダムな 2 文字の英数字 (数字 (0~9) またはアルファベット (A~Z, a~z))

ルート通信通番 : [xxxxxxxx] (4 バイトの 16 進表示のデータ)

xxxxxxxx : IP アドレス

RPC 通信通番 : [yyyy][zzzz] (4 バイトの 16 進表示のデータ)

yyyy : ランダムな 2 バイトの 16 進数字

zzzz : 通信通番 (該当する関数の呼び出しごとにインクリメントされます)

(b) トレースの出力例

ノード ID 「_WOX」、ルート通信通番 「f784d10a」、RPC 通信通番 「4e880002」の場合の出力例を次に示します。

TP1/Client の UAP トレースの出力例

関数入り口情報 (EVENT=BEGIN) と関数出口情報 (EVENT=END) の間で出力されます。取得ポイントは、送信前です。

```
DATE = 2008/08/11 TIME = 05:08:35.603 PID = 2072:3152 SIZE = 26
FUNC = dc_rpc_call_s                EVENT = PRF
Address  +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +a +b +c +d +e +f  0123456789abcdef
00000000  5f 57 4f 58 2f 30 78 66 37 38 34 64 31 30 61 2f  _WOX/0xf784d10a/
00000010  30 78 34 65 38 38 30 30 30 32                      0x4e880002
```

TP1/Server の性能検証用トレースの出力例

```
PRF: Rec Node: smpl Run-ID: 0x4743dfcc Process: 53000          Trace: 4          Event:
0x1003 Time: 2008/08/11 05:08:36 583.000.000 Server-name: svgrp
Rc: 0 Client: - 0x4e880002      Server: **** Root: _WOX - 0xf784d10a  Svc-Grp: *****
***** Svc: ***** Trn: *
```

(4) トレース情報の取得ポイント

TP1/Client では、次に示す関数を実行すると、UAP トレースに性能検証用の識別情報が出力されます。

- dc_rpc_call_s 関数
- dc_rpc_call_to_s 関数
- dc_clt_connect_s 関数
- dc_clt_disconnect_s 関数
- dc_trn_begin_s 関数
- dc_trn_chained_commit_s 関数
- dc_trn_chained_rollback_s 関数
- dc_trn_unchained_commit_s 関数
- dc_trn_unchained_rollback_s 関数

2.11.6 トラブルシュート機能を使用するときの注意事項

トレース編集出力コマンド (cltdump コマンド, または cltdmp32 コマンド) は, トレースを出力した TP1/Client のバージョンと同じバージョンのコマンドの使用をお勧めします。異なるバージョンのコマンドを使用した場合, 正しく編集されないことがあります。

正しい情報をファイル (エラーログおよび各種トレース) に出力するために, 次の点に注意してください。

1. ファイルを出力する CUP が動作中は, ファイルの移動および削除は行わないでください。
2. 複数の CUP のトレースファイルの出力先に同じディレクトリを指定すると, 次のような影響があるためプロセス単位, またはスレッド単位でトレースファイルの出力先を分けてください。
 - RPC スループットが劣化するおそれがあります。
 - 複数の CUP のトレース情報が混在し, トラブルシュートが困難になります。

表 2-10 ファイルサイズを指定するクライアント環境定義と出力されるファイル名

クライアント環境定義	ファイル名
DCTRRCERR	dcerr1.trc dcerr2.trc
DCTRUCUAP	dcuap1.trc dcuap2.trc
DCTRCSOC	dcsoc1.trc dcsoc2.trc
DTRCMDL	dcmdl1.trc dcmdl2.trc

なお, ファイルの出力先は, クライアント環境定義 DTRCPATH に指定したディレクトリです。指定を省略した場合は, カレントディレクトリ (CUP 実行ディレクトリ) です。

2.12 ホスト名拡張機能

TP1/Client で扱えるホスト名長は 63 文字を制限としていますが、この機能を使用することで 255 文字まで拡張できます。この機能を使用する場合、クライアント環境定義 DCCLTOPTION に 00000008（論理和）を指定してください。

TP1/Client が提供しているヘッダファイル dcvclt.h に、MAXHOSTNAME (=64) と DCMAXDNSNAME (=256) が定義されています。CUP を作成するとき、必要に応じて使用してください。

2.12.1 C 言語の関数の引数に指定できるホスト名長およびホスト名格納領域長

ホスト名拡張機能を使用する場合、C 言語の関数の引数に指定できるホスト名長およびホスト名格納領域長を拡張できます。

C 言語の関数の引数に指定できるホスト名長を次の表に示します。

表 2-11 C 言語の関数の引数に指定できるホスト名長

関数	引数	複数ホストの指定	指定可能なホスト名長	
			拡張しない場合	拡張する場合
dc_clt_cltin_s	target_host	○	63 文字	255 文字
dc_clt_set_raphost_s	raphost		(255 文字) ※	(1023 文字) ※
DCRPC_DIRECT_SCHEDULE	hostnm	×	63 文字	255 文字
dc_clt_cancel_notification_s	hostname			
dc_clt_send_s	hostname			

(凡例)

- ：複数のホストを指定できます。
- ×：複数のホストを指定できません。

注※

括弧内の文字数は、引数に指定できる最大文字数（ポート番号および区切り文字を含む）です。

C 言語の関数の引数に指定できるホスト名格納領域長を次の表に示します。

表 2-12 C 言語の関数の引数に指定できるホスト名格納領域長

関数	引数	格納するホスト数	ホスト名格納領域長	
			拡張しない場合 (単位: バイト)	拡張する場合 (単位: バイト)
dc_clt_get_raphost_s	raphost	複数	256 以上	1024 以上
dc_clt_cltin_s	set_host	1 個	64 以上	256 以上
dc_clt_accept_notification_s	hostname			
dc_clt_chained_accept_notification_s	hostname			

注

上記の関数の引数には、必ず上記規定以上の領域を用意してください。領域が規定値未満の場合、CUP が異常終了するおそれがあります。

2.12.2 ホスト名拡張機能使用時の COBOL-UAP 作成用プログラム

COBOL 言語で CUP を作成する場合、ホスト名拡張機能を使用するときには、次に示すプログラムを利用してください。なお、ホスト名拡張機能を使用する場合のデータ領域長については、「6. TP1/Client で使用できる要求文 (COBOL 言語編)」の各プログラムの UAP で値を設定するデータ領域の説明を参照してください。

表 2-13 ホスト名拡張機能利用時に使用する CALL 文で呼び出す COBOL-UAP 作成用プログラム

機能		CALL 文で呼び出す COBOL-UAP 作成用プログラム
ユーザ認証機能	クライアントユーザの認証要求	CBLDCCLS(' EXCLTIN ')
		CBLDCCLT(' EXCLTIN ')
常設コネクション	常設コネクション確立要求先の指定	CBLDCCLS(' STRAPHST')*
		CBLDCCLT(' STRAPHST')*
	常設コネクション確立要求先の取得	CBLDCCLS(' GTRAPHST')*
		CBLDCCLT(' GTRAPHST')*
TCP/IP 通信機能	メッセージの送信	CBLDCCLS(' EXSEND ')
		CBLDCCLT(' EXSEND ')
サーバからの一方通知受信機能	一方通知メッセージの受信	CBLDCCLS(' EXNACPT ')
		CBLDCCLT(' EXNACPT ')
	一方通知待ち状態のキャンセル	CBLDCCLS(' EXNCANCL')
		CBLDCCLT(' EXNCANCL')
	一方通知受信	CBLDCCLS(' EXNCACPT')

機能		CALL 文で呼び出す COBOL-UAP 作成用プログラム
サーバからの一方通知受信機能	一方通知受信	CBLDCCLT('EXNCACPT')

注※

クライアント環境定義 DCCLTOPTION に 00000008 を指定した場合、データ領域を大きくする必要があります。

2.12.3 クライアント環境定義のオペランドで指定できる文字数

ホスト名拡張機能を使用する場合、次のクライアント環境定義のオペランドで指定するホスト名に、最大 255 文字のホスト名を指定できます。

- DCHOST※
- DCCLTRAPHOST※
- DCCLTDCCMHOST※
- DCSNDHOST
- DCCLTCUPSNDHOST

注※

オペランドに指定できる長さは最大 1023 文字です（このほかの定義の場合、オペランドに指定できる長さは最大 255 文字です）。

また、クライアント環境定義 DCCLTSERVICEGROUPLIST で指定するファイルには、最大 255 文字のホスト名を指定できます。

2.12.4 ホスト名拡張機能を使用するときの注意事項

ホスト名を指定する場合、次の点に注意してください。

- ホスト名に、区切り文字として扱っているコロン (:) およびコンマ (,)、コメントとして扱っているセミコロン (;) を使用することはできません。
- ホスト名は英数字（必ず英字を含める）で記述してください。
- ホスト名の最後には'¥0'を付けてください。

なお、ホスト名拡張機能を使用しない場合は、63 文字+'¥0'がホスト名の最大長です。

2.13 送信元ホスト指定機能

TP1/Client では、サーバとのコネクション確立要求時に、送信元ホストを指定できます。この機能を、送信元ホスト指定機能といいます。

CUP が動作するホスト上に、複数のネットワークアダプタが接続されている場合、CUP がこれらのコネクション確立要求時に使用する送信元ホストは、TCP/IP の制御で任意に決まります。しかし、送信元ホスト指定機能を使用すると、コネクション確立要求時の送信元ホストを指定できます。

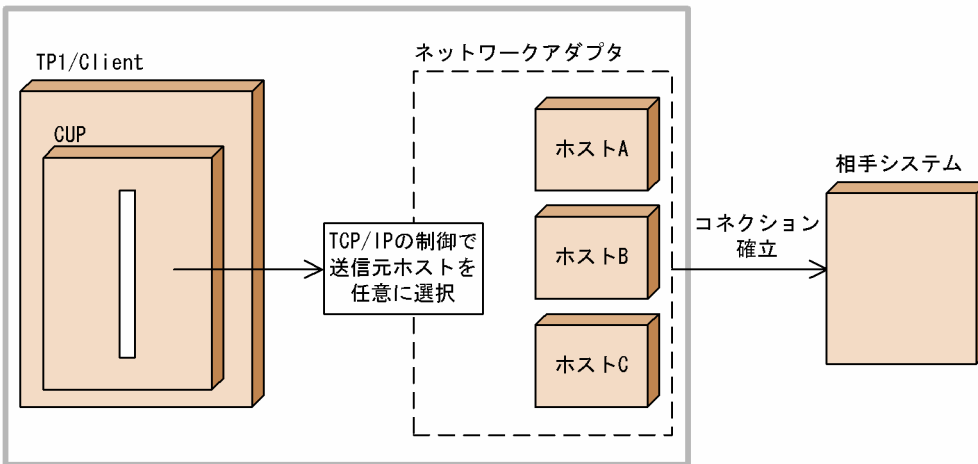
送信元ホストは、クライアント環境定義 DCCLTCUPSNDHOST に指定します。

送信元ホスト指定機能を使用しない場合と使用する場合について、次の図に示します。

図 2-42 送信元ホスト指定機能を使用しない場合と使用する場合

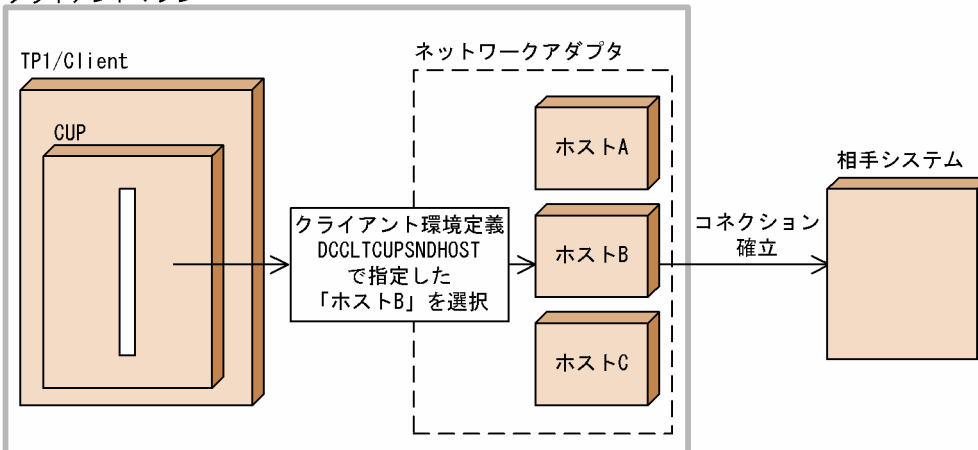
●CUPの送信元ホスト指定機能を使用しない場合

クライアントマシン



●CUPの送信元ホスト指定機能を使用する場合

クライアントマシン



2.14 受信ポート固定機能

TP1/Client では、サーバからの接続確立受け付け時の受信ポートを固定できます。この機能を、受信ポート固定機能といいます。

この機能は、TP1/Server から TP1/Client へ接続を確立する場合で、TP1/Server と TP1/Client との間に設置したファイアウォールで TP1/Client の受信ポートにだけ通知を許可するようにフィルタリングしたいときに使用します。

この機能を使用する場合は、クライアント環境定義 DCCLTCUPRCVPORT に受信ポートを指定します。

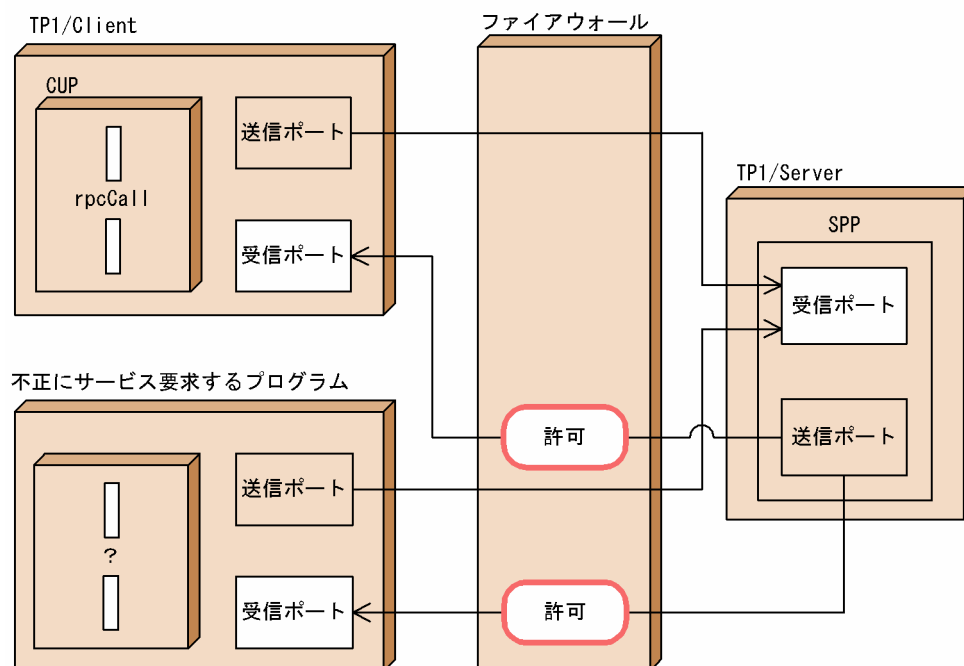
受信ポート固定機能を使用しない場合と使用する場合について説明します。

2.14.1 受信ポート固定機能を使用しない場合

RPC の応答通信で、受信ポートに対するフィルタリングはありません。OS が、TP1/Client の RPC の受信ポートとして不定のポートを自動的に割り当てます。

受信ポート固定機能を使用しない場合について、スケジューラダイレクト機能を使用した RPC を例に、次の図に示します。

図 2-43 受信ポート固定機能を使用しない場合（スケジューラダイレクト機能を使用した RPC）

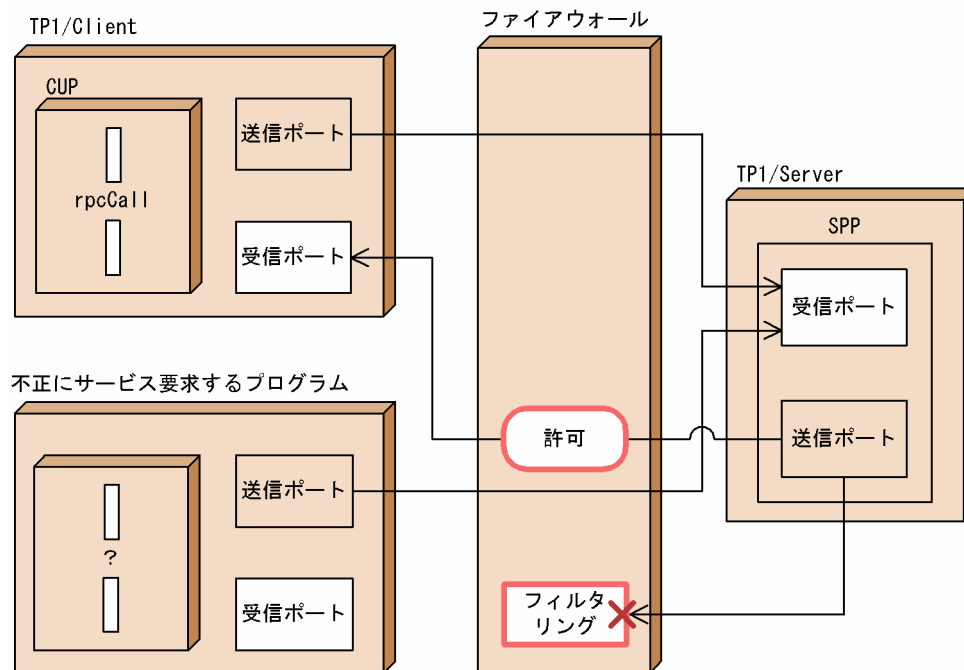


2.14.2 受信ポート固定機能を使用する場合

RPC への応答通信で許可する受信ポートを、クライアント環境定義 DCCLTCUPRCVPORT で指定したポートとし、それ以外はフィルタリング対象にします。このため、不正なサービス要求に対する TP1/Server からの応答通信を、ファイアウォールでフィルタリングできます。

受信ポート固定機能を使用する場合について、スケジューラダイレクト機能を使用した RPC を例に、次の図に示します。

図 2-44 受信ポート固定機能を使用する場合（スケジューラダイレクト機能を使用した RPC）



なお、受信ポート固定機能は、次の場合には適用されません。

- TCP/IP 通信機能を使用する場合
クライアント環境定義 DCRCVPORT に指定された受信ポートを使用します。
- サーバからの一方通知受信機能を使用する場合
dc_clt_accept_notification_s 関数、または dc_clt_open_notification_s 関数の引数に指定された受信ポートを使用します。
- rap サーバとの通信をする場合
TP1/Client から確立したコネクションを使用して送受信します。受信ポート固定機能を使用しなくても、安全にファイアウォールを通過させられます。

3

ユーザアプリケーションプログラムの作成 (C 言語編)

ユーザアプリケーションプログラムの C 言語での関数インタフェース、および翻訳と結合について説明します。

この章では、各関数を、`_s` 付き関数名 (`dc_xxx_xxx_s`) を用いて説明します。`_s` 無し関数を使う場合は、`_s` 無し関数名 (`dc_xxx_xxx`) に置き換えて読んでください。

3.1 関数インタフェース

TP1/Client の関数インタフェースについて説明します。

CUP は、ANSI C 仕様の C 言語で作成してください。

CUP は OpenTP1 のサービス利用プログラム (SUP) と同様にスタブを使用しません。そのため、ユーザデータのコード (コード体系, およびバイトオーダ) はユーザプログラムで変換してください。

3.1.1 関数の一覧

関数の一覧を、表 3-1 に示します。

dc_xxx_xxx_s 関数, および文字コード変換関数は、マルチスレッド環境で使用できます。そのため、文字コード変換関数以外で dc_xxx_xxx 関数と dc_xxx_xxx_s 関数をサポートしている関数は、dc_xxx_xxx_s 関数を使用することをお勧めします。

ただし、ご使用になられる TP1/Client 製品のプログラムプロダクトによっては、_s 付きの関数をサポートしていない場合がありますので、「リリースノート」でご確認ください。

各関数の詳細については「4. TP1/Client で使用できる関数 (C 言語編)」を参照してください。

表 3-1 関数の一覧

機能		関数名
ユーザ認証機能	クライアントユーザの認証要求	dc_clt_cltin_s
		dc_clt_cltin
	クライアントユーザの認証解除	dc_clt_cltout_s
		dc_clt_cltout
リモートプロシジャコール	UAP の開始	dc_rpc_open_s
		dc_rpc_open
	UAP の終了	dc_rpc_close_s
		dc_rpc_close
	遠隔サービスの要求	dc_rpc_call_s
		dc_rpc_call
	通信先を指定した遠隔サービスの要求	dc_rpc_call_to_s
		dc_rpc_call_to
	サービス応答待ち時間の更新	dc_rpc_set_watch_time_s
		dc_rpc_set_watch_time

機能		関数名	
リモートプロシジャコール	サービス応答待ち時間の参照	dc_rpc_get_watch_time_s	
		dc_rpc_get_watch_time	
	DCRPC_BINDING_TBL 構造体の作成	DCRPC_DIRECT_SCHEDULE	
常設コネクション	常設コネクションの確立	dc_clt_connect_s	
		dc_clt_connect	
	常設コネクションの解放	dc_clt_disconnect_s	
		dc_clt_disconnect	
	常設コネクション確立要求先の指定	dc_clt_set_raphost_s	
		dc_clt_set_raphost	
	常設コネクション確立要求先の取得	dc_clt_get_raphost_s	
		dc_clt_get_raphost	
	端末識別情報の設定	dc_clt_set_connect_inf_s	
		dc_clt_set_connect_inf	
	トランザクション制御	トランザクションの開始	dc_trn_begin_s
			dc_trn_begin
連鎖モードのコミット		dc_trn_chained_commit_s	
		dc_trn_chained_commit	
連鎖モードのロールバック		dc_trn_chained_rollback_s	
		dc_trn_chained_rollback	
非連鎖モードのコミット		dc_trn_unchained_commit_s	
		dc_trn_unchained_commit	
非連鎖モードのロールバック		dc_trn_unchained_rollback_s	
		dc_trn_unchained_rollback	
現在のトランザクションに関する情報の報告		dc_trn_info_s	
		dc_trn_info	
現在のトランザクションに関する識別子の取得		dc_clt_get_trnid_s	
		dc_clt_get_trnid	
TCP/IP 通信機能		メッセージの送信	dc_clt_send_s
			dc_clt_send
	メッセージの受信	dc_clt_receive_s	
		dc_clt_receive	

機能		関数名
TCP/IP 通信機能	メッセージの受信（障害時メッセージ受信）	dc_clt_receive2_s
		dc_clt_receive2
	組み立てメッセージの送信	dc_clt_assem_send_s
		dc_clt_assem_send
	組み立てメッセージの受信	dc_clt_assem_receive_s
		dc_clt_assem_receive
サーバからの一方通知受信機能	一方通知メッセージの受信	dc_clt_accept_notification_s
		dc_clt_accept_notification
	一方通知待ち状態のキャンセル	dc_clt_cancel_notification_s
		dc_clt_cancel_notification
	一方通知受信の開始	dc_clt_open_notification_s
		dc_clt_open_notification
	一方通知受信の終了	dc_clt_close_notification_s
		dc_clt_close_notification
	一方通知受信	dc_clt_chained_accept_notification_s
		dc_clt_chained_accept_notification
XATMI インタフェース機能	型付きバッファの割り当て	tpalloc
	型付きバッファの解放	tpfree
	会話型サービスとの接続の確立	tpconnect
	会話型サービスとの接続の切断	tpdiscon
	会話型サービスへのメッセージの送信	tpsend
	会話型サービスからのメッセージの受信	tprecv
文字コード変換機能（コードマッピングテーブルを使用しない場合）※	文字コード変換	dc_clt_code_convert
文字コード変換機能（コードマッピングテーブルを使用する場合）※	文字コード変換の開始	dc_clt_codeconv_open
	文字コード変換の終了	dc_clt_codeconv_close
	文字コード変換の実行	dc_clt_codeconv_exec

注※

TP1/Client/P でだけ使用できる機能です。

3.1.2 関数の記述形式

「4. TP1/Client で使用できる関数 (C 言語編)」では、使用する関数を次の形式で説明します。

形式

TP1/Client のライブラリで提供する関数の定義形式と、引数のデータ型を示します。引数に値を指定するときは、ここで示すデータ型に従ってください。

機能

ここで説明する関数の機能を説明します。

UAP で値を設定する引数

関数の実行時に、値を指定しておく引数を示します。各引数の説明に従って、値を指定してください。

値が返される引数

関数を実行したあとに、OpenTP1、サーバ UAP、TP1/Client などから値が返される引数です。

リターン値

関数を実行したときに戻ってくる値を、表形式で説明します。このリターン値によって、関数が正常に実行されたかがわかります。エラーが発生したときは、エラーの内容を示します。

UAP を作成するときは、数値ではなく、必ずここで示す定義名でリターン値を使用してください。リターン値の定数名はヘッダファイルで定義されています。

注意事項

関数を使用するときに注意しなければならないことを示します。

(1) 引数に指定する値の説明で使う記号

関数の引数に指定する値の説明で使う記号の一覧を示します。

記号	説明
{ }	この記号で囲まれている複数の項目のうちから一つを選択することを示します。項目の区切りは で示します。 (例) {DCCLT_CNV_EBCDIC DCCLT_CNV_EBCDIK} DCCLT_CNV_EBCDIC と DCCLT_CNV_EBCDIK の二つの項目で、どちらかを指定することを示します。
[]	この記号で囲まれている項目は、省略しても良いことを示します。 (例) [DCNOFLAGS] DCNOFLAGS は省略できることを示します。
— (下線)	括弧 [] で囲まれた複数の項目のうちで、括弧内のすべてを省略したときに、TP1/Client で仮定する標準値を示します。 (例) [{DCCLT_CNV_SPCHAN <u>DCCLT_CNV_SPCZEN</u> }] DCCLT_CNV_SPCHAN, DCCLT_CNV_SPCZEN の両方とも省略した場合、DCCLT_CNV_SPCZEN を仮定します。

記号	説明
...	記述が省略されていることを示します。この記号の直前に示された項目を繰り返し複数個指定できます。 (例) ホスト名 [:ポート番号] [,ホスト名 [:ポート番号] ,...] ホスト名 [:ポート番号] を続けて指定できることを示します。
~	この記号の前に示された項目が、~に続く < >, (())に従うことを示します。
<文字列>	任意の文字の配列を示します。
<符号なし整数>	数字 (0~9) を示します。
(())	指定値の指定範囲を示します。

(2) 引数として指定する記号の説明

引数として指定する記号を示します。

記号	説明
 (ストローク)	一つの引数に複数の項目を指定するときに、項目同志の区切りを示します。 この記号を項目と項目の間に入れて指定してください。 (例) {DCCLT_CNV_EBCDIC DCCLT_CNV_EBCDIK} [{DCCLT_CNV_SPCHAN DCCLT_CNV_SPCZEN}] DCCLT_CNV_EBCDIC と DCCLT_CNV_SPCHAN を指定するときは、 「DCCLT_CNV_EBCDIC DCCLT_CNV_SPCHAN」と指定します。

3.2 ユーザアプリケーションプログラムの翻訳と結合

翻訳と結合の方法は、OS 環境によって異なります。

3.2.1 UNIX 環境の場合の翻訳と結合

(1) 翻訳

CUP は ANSI C 仕様の C 言語で記述します。C 言語で作成した CUP のオブジェクトファイルを作成するために、ソースプログラムを翻訳します。翻訳は C コンパイラで実行します。

AIX および Linux でソースプログラムを翻訳するときのコマンド入力例を次に示します。

〈例〉

C 言語で作成した CUP のソースプログラム名

- cupmain.c (メイン関数)
- cupfnc1.c (内部関数 1)
- cupfnc2.c (内部関数 2)

この場合、それぞれのソースプログラムは次のように翻訳します。

<AIX の場合>

```
xlc -c -I/usr/include cupmain.c
xlc -c -I/usr/include cupfnc1.c
xlc -c -I/usr/include cupfnc2.c
```

<Linux の場合>

```
cc -c -I/usr/include -ansi cupmain.c
cc -c -I/usr/include -ansi cupfnc1.c
cc -c -I/usr/include -ansi cupfnc2.c
```

マルチスレッド対応の `_s` 付き関数を使用している場合は、次のように翻訳します。

<AIX の場合>

```
xlc_r -c cupmain.c
xlc_r -c cupfnc1.c
xlc_r -c cupfnc2.c
```

<Linux の場合>

```
cc -c -I/usr/include -ansi -lpthread cupmain.c
cc -c -I/usr/include -ansi -lpthread cupfnc1.c
cc -c -I/usr/include -ansi -lpthread cupfnc2.c
```

上記 cc コマンドを実行すると、次のオブジェクトファイルが作成されます。

- cupmain.o (メイン関数のオブジェクトファイル)
- cupfnc1.o (内部関数 1 のオブジェクトファイル)
- cupfnc2.o (内部関数 2 のオブジェクトファイル)

(2) 結合

CUP の実行形式ファイルは、次に示すファイルを結合させて作成します。結合は cc コマンドで実行します。

- CUP のオブジェクトファイル (メイン関数と内部関数)
- TP1/Client のライブラリ

上記のファイルを結合するときのコマンドの入力例を次に示します。

〈例〉

CUP の実行形式ファイル「example」を作成する場合

- メイン関数のオブジェクトファイル名…cupmain.o
- 内部関数のオブジェクトファイル名…cupfnc1.o, cupfnc2.o

この場合、次のようにファイルを結合します。

```
cc -o example cupmain.o cupfnc1.o cupfnc2.o -L/usr/lib -lclt
```

マルチスレッド対応の CUP を作成する場合、次のように結合します。

```
xlcr -o example cupmain.o cupfnc1.o cupfnc2.o -L/usr/lib -lclt
```

注

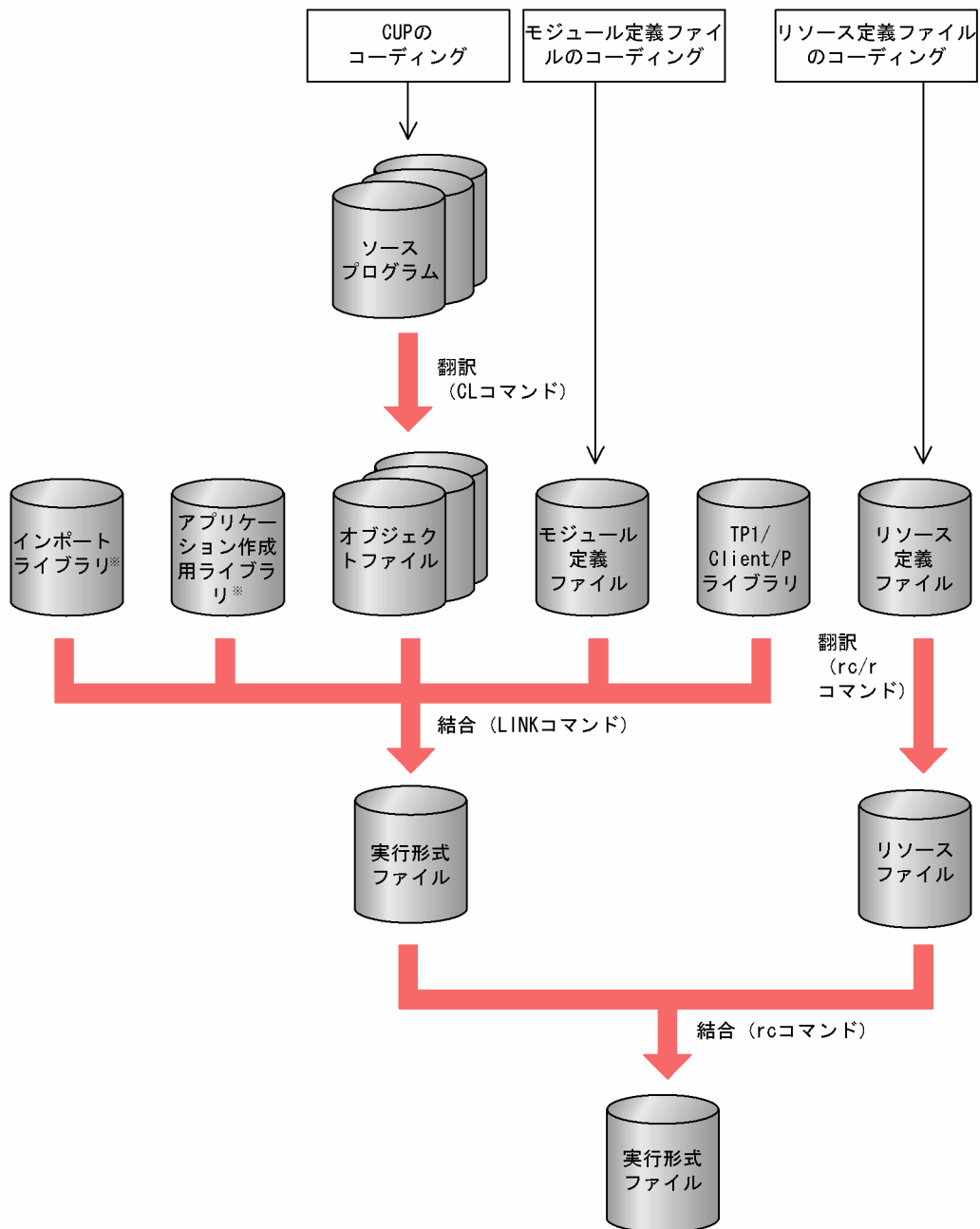
-L オプションの指定は省略できます。

3.2.2 Windows 環境の場合の翻訳と結合

(1) 手順

CUP 作成時の手順を次の図に示します。

図 3-1 CUP 作成時の手順



注※ Windows SDKが提供するライブラリです。

(2) 翻訳と結合

(a) ソースプログラムの翻訳

Windows 環境で CUP のオブジェクトファイルを作成するには、Microsoft Visual C++を使用します。翻訳は CL コマンドで実行します。

翻訳時に必須となるオプションを次の表に示します。

表 3-2 翻訳時の必須オプション

クライアント機能	オプション	オプションの意味
TP1/Client/P	/DDCCLTDLL	TP1/Client/P 提供のヘッダファイルを展開します。

ソースプログラムを翻訳するときのコマンドの入力例を次に示します。

〈例〉

C 言語で作成した CUP のソースプログラム名…

- cup.c (メイン関数)
- cupsub1.c (内部関数 1)
- cupsub2.c (内部関数 2)

この場合、それぞれのソースプログラムを次のように翻訳します。

```
CL /DDCCLTDLL /c cup.c  
CL /DDCCLTDLL /c cupsub1.c  
CL /DDCCLTDLL /c cupsub2.c
```

上記 CL コマンドを実行すると、次のオブジェクトファイルが作成されます。

- cup.obj (メイン関数のオブジェクトファイル)
- cupsub1.obj (内部関数 1 のオブジェクトファイル)
- cupsub2.obj (内部関数 2 のオブジェクトファイル)

(b) リソース定義ファイルの作成

この例では、リソースとしてアイコンを定義します。リソース定義ファイル cup.rc を次のように作成します。

```
CUPI ICON cup.ico
```

注

CUPI はアイコンに付けた任意の名称です。

(c) リソースの翻訳

Microsoft Visual C++ のリソースコンパイラを使用して、リソース定義ファイル (cup.rc) を次のように翻訳します。

```
rc /r cup.rc
```

上記 rc コマンドを実行すると、リソースファイル cup.res が生成されます。

(d) モジュール定義ファイルの作成

モジュール定義ファイル cup.def の作成例を次に示します。

NAME	CUPEXEC
HEAPSIZE	1024
STACKSIZE	8192

注

STACKSIZE は、8192 以上を指定してください。

(e) CUP の結合

CUP の実行形式ファイルは、次に示すファイルを結合させて作成します。結合は、LINK コマンドで実行します。

- CUP のオブジェクトファイル（メイン関数と内部関数）
- TP1/Client/P のライブラリ（CLTW32.LIB, CLTWS32.LIB または CLTW64.LIB）※
- インポートライブラリ（Windows SDK および Visual Studio が提供）
- Windows アプリケーション作成用のライブラリ（Windows SDK および Visual Studio が提供）
- モジュール定義ファイル

注※

文字コード変換機能を使用する場合は、CLTCNV32.LIB または CLTCNV64.LIB も必要です。

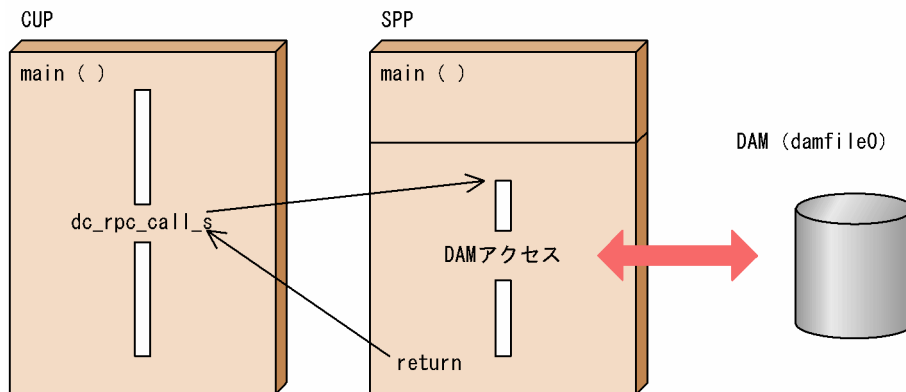
3.3 ユーザアプリケーションプログラムの作成例

ここでは、UAP を作成する場合の CUP や SPP などのコーディングについて、例を使って説明します。

3.3.1 CUP と SPP の作成

CUP と SPP の構成例を次の図に示します。例題では、Windows 環境以外の場合を示します。

図 3-2 CUP と SPP の構成例



この例題で使用する CUP のコーディング例を次に示します。

```
000010 #include <stdio.h>
000020 #include <stdlib.h>
000030 #include <string.h>
000040 #include <dcvclt.h>
000050 #include <dcvrpc.h>
000060
000070 #define BUFSIZE 512
000080 #define SERVICE "spp01"
000090
000100 main()
000110 {
000120     char    in[BUFSIZE];
000130     DCULONG in_len;
000140     char    out[BUFSIZE];
000150     DCULONG out_len;
000160     char    indata[BUFSIZE];
000170     DCLONG  rc;
000180     DCCLT_ID cltid;
000190     char    clt_flag = 0;
000200     char    rpc_flag = 0;
000210
000220     /*
000230     * クライアントユーザの認証要求
000240     */
000250     if((rc = dc_clt_cltin_s(NULL, &cltid, NULL, NULL, "user01", "puser01",
000260         NULL, DCNOFLAGS)) != DC_OK){
000270         printf("cup01: dc_clt_cltin_sに失敗しました。CODE=%d\n", rc);
000280         goto PROG_EXIT;
```

```

000290     }
000300     clt_flag = 1;
000310
000320     /*
000330     *   RPC-OPEN(RPC環境の初期設定)
000340     */
000350     if((rc = dc_rpc_open_s(cltid, DCNOFLAGS)) != DC_OK) {
000360         printf("cup01: dc_rpc_open_sに失敗しました。CODE=%d\n", rc);
000370         goto PROG_END;
000380     }
000390     rpc_flag = 1;
000400
000410     while (1) {
000420         printf("***** 伝言板メニュー *****\n");
000430         printf("伝言の取り出し ... [1]   伝言の書き込み ... [2]\n");
000440         printf("終了 ..... [9]\n");
000450         printf("番号を入力して下さい。=>");
000460         gets(indata);
000470
000480         if(indata[0] == '1') {
000490
000500             /*
000510             *   RPC-CALL(RPCの実行)
000520             */
000530             strcpy(in, "cup01");
000540             in_len = strlen(in) + 1;
000550             out_len = sizeof(out);
000560             if((rc = dc_rpc_call_s(cltid, SERVICE, "get", in, &in_len, out,
000570                 &out_len, DCNOFLAGS)) != DC_OK) {
000580                 printf("cup01: dc_rpc_call_sに失敗しました。CODE=%d\n", rc);
000590                 goto PROG_END;
000600             }
000610             printf("伝言の内容: %s\n", out);
000620         }
000630
000640         else if(indata[0] == '2') {
000650             printf("伝言を入力して下さい =>");
000660             gets(indata);
000670             if(indata[0] == '\0') {
000680                 strcpy(indata, "伝言はありません。");
000690             }
000700
000710             /*
000720             *   RPC-CALL(RPCの実行)
000730             */
000740             strcpy(in, indata);
000750             in_len = strlen(in) + 1;
000760             out_len = sizeof(out);
000770             if((rc = dc_rpc_call_s(cltid, SERVICE, "put", in, &in_len, out,
000780                 &out_len, DCNOFLAGS)) != DC_OK) {
000790                 printf("cup01: dc_rpc_call_sに失敗しました。CODE=%d\n", rc);
000800                 goto PROG_END;
000810             }
000820             printf("%s\n", out);
000830         }
000840
000850         else if(indata[0] == '9') {
000860             break;

```



```

000870     }
000880
000890     else {
000900         continue;
000910     }
000920 }
000930
000940 PROG_END:
000950 /*
000960  * RPC-CLOSE(RPC環境の解除)
000970 */
000980 if(rpc_flag) {
000990     dc_rpc_close_s(cltid, DCNOFLAGS);
001000 }
001010
001020 PROG_EXIT:
001030 if(clt_flag) {
001040     dc_clt_cltout_s(cltid, DCNOFLAGS);
001050 }
001060 exit(0);
001070 }

```

この例題で使用する、SPP のコーディング例（メイン関数 DAM アクセス）を次に示します。

```

000010 #include <stdio.h>
000020 #include <dcrpc.h>
000030 #include <dcdam.h>
000040 #define DAMFILE "damfile0"
000050
000060 int damfd;
000070
000080 main()
000090 {
000100     int rc;
000110
000120     /*
000130     * RPC-OPEN(UAPの開始)
000140     */
000150     if ((rc = dc_rpc_open(DCNOFLAGS)) != DC_OK) {
000160         printf("spp01: dc_rpc_openに失敗しました。CODE=%d\n", rc);
000170         goto PROG_END;
000180     }
000190     /*
000200     * DAM-OPEN(論理ファイルのオープン)
000210     */
000220     if ((rc = dc_dam_open(DAMFILE, DCDAM_BLOCK_EXCLUSIVE)) < 0) {
000230         printf("spp01: dc_dam_openに失敗しました。CODE=%d\n", rc);
000240         goto PROG_END;
000250     }
000260     damfd = rc;
000270     /*
000280     * RPC-MAINLOOP(SPPのサービス開始)
000290     */
000300     printf("spp01: mainloopに入ります。¥n");
000310     if ((rc = dc_rpc_mainloop(DCNOFLAGS)) != DC_OK) {
000320         printf("spp01: dc_rpc_mainloopに失敗しました。CODE=%d\n", rc);
000330     }

```

```

000340 /*
000350 * DAM-CLOSE(論理ファイルのクローズ)
000360 */
000370 if ((rc = dc_dam_close(damfd, DCNOFLAGS)) != DC_OK) {
000380     printf("spp01: dc_dam_closeに失敗しました。CODE=%d\n", rc);
000390 }
000400 PROG_END:
000410 /*
000420 * RPC-CLOSE(UAPの終了)
000430 */
000440 dc_rpc_close(DCNOFLAGS);
000450 printf("spp01: SPPのサービス処理を終了します。¥n");
000460 exit(0);
000470 }

```

この例題で使用する、SPP のコーディング例（サービス関数 DAM アクセス）を次に示します。

```

000010 #include <stdio.h>
000020 #include <string.h>
000030 #include <dcrpc.h>
000040 #include <dctrn.h>
000050 #include <dcdam.h>
000060 #define DAMBLKSIZE 504
000070
000080 extern int damfd;
000090 static char damblk[DAMBLKSIZE];
000100
000110 void get(in, in_len, out, out_len)
000120     char          *in;
000130     DCULONG      *in_len;
000140     char          *out;
000150     DCULONG      *out_len;
000160 {
000170     int          rc;
000180     struct DC_DAMKEY keyptr;
000190     static char  *service = "get";
000200
000210     printf("%s: %sからのサービス要求を受け付けました。¥n", service, in);
000220
000230     /*
000240     * TRN-BEGIN(トランザクションの開始)
000250     */
000260     if ((rc = dc_trn_begin()) != DC_OK) {
000270         sprintf(out, "%s: dc_trn_beginに失敗しました。CODE=%d\n",
000280             service, rc);
000290         printf("%s", out);
000300         goto PROG_END;
000310     }
000320     /*
000330     * DAM_READ(DAMファイルの読み込み)
000340     */
000350     keyptr.fstblkno = 0;
000360     keyptr.endblkno = 0;
000370     if ((rc = dc_dam_read(damfd, &keyptr, 1, damblk, DAMBLKSIZE,
000380         DCDAM_REFERENCE | DCDAM_NOWAIT)) != DC_OK) {
000390         sprintf(out, "%s: dc_dam_readに失敗しました。CODE=%d\n",
000400             service, rc);

```

```

000410     printf("%s", out);
000420     goto TRN_COMMIT;
000430 }
000440 strncpy(out, damblk, *out_len);
000450
000460 TRN_COMMIT:
000470 /*
000480  * TRN_UNCHAINED_COMMIT(非連鎖モードのコミット)
000490 */
000500 if ((rc = dc_trn_unchained_commit()) != DC_OK) {
000510     sprintf(out, "%s: dc_trn_unchained_commitに失敗しました。CODE=%d\n",
000520             service, rc);
000530     printf("%s", out);
000540 }
000550 PROG_END:
000560     *out_len = strlen(out) + 1;
000570     return;
000580 }
000590
000600 void put(in, in_len, out, out_len)
000610     char             *in;
000620     DCULONG         *in_len;
000630     char             *out;
000640     DCULONG         *out_len;
000650 {
000660     int             rc;
000670     struct DC_DAMKEY keyptr;
000680     static char     *service = "put";
000690
000700     printf("%s: サービス要求を受け付けました。%n", service);
000710
000720     /*
000730     * TRN-BEGIN(トランザクションの開始)
000740     */
000750     if ((rc = dc_trn_begin()) != DC_OK) {
000760         sprintf(out, "%s: dc_trn_beginに失敗しました。CODE=%d\n",
000770                 service, rc);
000780         printf("%s", out);
000790         goto PROG_END;
000800     }
000810     /*
000820     * DAM_WRITE(DAMファイルへ書き込み)
000830     */
000840     keyptr.fstblkno = 0;
000850     keyptr.endblkno = 0;
000860     strncpy(damblk, in, DAMBLKSIZE);
000870     if ((rc = dc_dam_write(damfd, &keyptr, 1, damblk,
000880                           DAMBLKSIZE, DCDAM_WAIT)) != DC_OK) {
000890         sprintf(out, "%s: dc_dam_writeに失敗しました。CODE=%d\n",
000900                 service, rc);
000910         printf("%s", out);
000920         dc_trn_unchained_rollback();
000930         goto PROG_END;
000940     }
000950     sprintf(out, "%s: 正常に処理を終了しました。%n", service);
000960     /*
000970     * TRN_UNCHAINED_COMMIT(非連鎖モードのコミット)
000980     */

```

```

000990     if ((rc = dc_trn_unchained_commit()) != DC_OK) {
001000         sprintf(out, "%s: dc_trn_unchained_commitに失敗しました。CODE=%d¥n",
001010             service, rc);
001020         printf("%s", out);
001030     }
001040 PROG_END:
001050     *out_len = strlen(out) + 1;
001060     return;
001070 }

```

3.3.2 マルチスレッド対応のユーザアプリケーションプログラムの作成

マルチスレッドで動作する CUP のコーディング例を示します。この例は、CUP から送信したメッセージをエコーバックする SPP を呼び出すプログラムです。

```

000010 #include <stdio.h>
000020 #include <string.h>
000030 #include <dcvclt.h>
000040 #include <dcvrpc.h>
000050 #include <pthread.h>
000060 #include <sys/errno.h>
000070 #define BUFSIZE 512
000080 #define SERVICE "spp01"
000090 #define THDMAX 5
000100
000110 void *CUP_thread(void *arg)
000120 {
000130     char    in[BUFSIZE];
000140     DCULONG in_len;
000150     char    out[BUFSIZE];
000160     DCULONG out_len;
000170     int     rc = DC_OK;
000180     DCCLT_ID cltid;
000190     int     myid;
000200
000210     myid = *((int *)arg);
000220
000230     /*--- クライアントユーザの認証要求 ---*/
000240     if ((rc = dc_clt_cltin_s(NULL, &cltid, NULL, NULL,
000250         "user01", "puser01", NULL, DCNOFLAGS)) != DC_OK) {
000260         printf("cup%d: dc_clt_cltinに失敗しました。CODE=%d¥n", myid, rc);
000270         goto PROG_EXIT;
000280     }
000290
000300     /*--- RPC-OPEN (RPC環境の初期設定) ---*/
000310     if ((rc = dc_rpc_open_s(cltid, DCNOFLAGS)) != DC_OK) {
000320         printf("cup%d: dc_rpc_openに失敗しました。CODE=%d¥n", myid, rc);
000330         goto PROG_END;
000340     }
000350
000360     /*--- RPC-CALL (RPCの実行) ---*/
000370     strcpy(in, "HELLO SPP !!");
000380     in_len = strlen(in) + 1;
000390     out_len = sizeof(out);

```

```

000400     if ((rc = dc_rpc_call_s(cltid, SERVICE, "echo", in, &in_len,
000410         out, &out_len, DCNOFLAGS)) != DC_OK) {
000420         printf("cup%d: dc_rpc_callに失敗しました。CODE=%d\n", myid, rc);
000430         goto PROG_END;
000440     }
000450     printf("%s\n", out);
000460 PROG_END:
000470
000480     /*--- RPC-CLOSE (RPC環境の解除) ---*/
000490     dc_rpc_close_s(cltid, DCNOFLAGS);
000500
000510 PROG_EXIT:
000520     /*--- クライアントユーザの認証解除 ---*/
000530     dc_clt_cltout_s(cltid, DCNOFLAGS);
000540
000550     /*--- スレッドを終了させる ---*/
000560     pthread_exit(arg);
000570 }
000580
000590 main()
000600 {
000610     int         i;
000620     int         rc;
000630     int         *exit_value;
000640     pthread_t   threads[THDMAX];
000650
000660     /*--- スレッドを生成する ---*/
000670     for (i = 1; i < THDMAX; i++) {
000680         rc = pthread_create((pthread_t *)&threads[i],
000690                             NULL,
000700                             CUP_thread,
000710                             (void *)&i);
000720         if (rc < 0) {
000730             printf("cup0: pthread_createに失敗しました。CODE=%d\n", errno);
000740         }
000750     }
000760
000770     /*--- スレッドの終了を待ち合わせる ---*/
000780     for (i = 1; i < THDMAX; i++) {
000790         rc = pthread_join(threads[i], (void **)&exit_value);
000800         if (rc < 0) {
000810             printf("cup0: pthread_joinに失敗しました。CODE=%d\n", errno);
000820         }
000830     }
000840 }

```

4

TP1/Client で使用できる関数 (C 言語編)

TP1/Client で使用できる関数について説明します。

この章では、各関数を、`_s` 付き関数名 (`dc_xxx_xxx_s`) を用いて説明します。`_s` 無し関数を使う場合は、`_s` 無し関数名 (`dc_xxx_xxx`) に置き換えて読んでください。

4.1 関数を使用するときの注意事項

関数を使用するときの注意事項を次に示します。

- マルチスレッド環境でも動作する、`_s` 付き関数をご使用になられることをお勧めします。ただし、ご使用される TP1/Client のプログラムプロダクトによっては、`_s` 付き関数をサポートしていない場合がありますので、「リリースノート」でご確認ください。
- TP1/Client/W の場合、CLTFAR ポインタは不要です（ただし、指定しても問題ありません）。また、TP1/Client/P の場合、CLTFAR ポインタは必須です。CLTFAR ポインタの動作は、`dcvclt.h` に定義されています。
- TP1/Client/P の `_s` 無し関数の場合、CLTFAR ポインタを指定しても、`far` ポインタを指定しても、問題ありません。
- `_s` 付き関数と `_s` 無し関数では、引数の数が異なります。マルチスレッド環境で動作させる場合は、`_s` 付き関数を使う必要があります。この環境で `_s` 無し関数を使うと、マルチスレッド環境では正しく動作しませんのでご注意ください。なお、文字コード変換機能で提供する関数は、`_s` 無し関数しかありませんが、これらの関数はマルチスレッド環境でも正しく動作します。

4.2 ユーザ認証機能

4.2.1 dc_clt_cltin_s – クライアントユーザの認証要求

(1) 形式

(a) TP1/Client/W の場合

■ _s 付き関数

```
#include <dcvclt.h>
DCLONG dc_clt_cltin_s(HWND hWnd, DCCLT_ID *cltid,
                    char *defpath,
                    char *target_host,
                    char *logname,
                    char *passwd,
                    char *set_host, DCLONG flags)
```

■ _s 無し関数

```
#include <dcvclt.h>
int dc_clt_cltin(char *target_host,
                char *logname,
                char *passwd,
                char *set_host, DCLONG flags)
```

(b) TP1/Client/P の場合

■ _s 付き関数

```
#include <dcvclt.h>
DCLONG dc_clt_cltin_s(HWND hWnd, DCCLT_ID CLTFAR *cltid,
                    char CLTFAR *defpath,
                    char CLTFAR *target_host,
                    char CLTFAR *logname,
                    char CLTFAR *passwd,
                    char CLTFAR *set_host, DCLONG flags)
```

■ _s 無し関数

```
#include <dcvclt.h>
int dc_clt_cltin(char CLTFAR *target_host,
                char CLTFAR *logname,
                char CLTFAR *passwd,
                char CLTFAR *set_host, DCLONG flags)
```


(2) 機能

指定された窓口となる TP1/Server に対して、ログイン名で指定されたクライアントユーザの認証を要求します。

ユーザ認証を抑止する場合でも、dc_clt_cltin_s 関数は必ず実行してください。

(3) UAP で値を設定する引数

- hWnd

NULL を指定します。

- cltid

クライアント ID (ヘッダファイル dcvcclt.h で定義されていて、型は DCCLT_ID) を受け取る領域へのポインタを指定します。

クライアントユーザの認証が正常に終了すると、指定した領域にクライアント ID が設定されます。設定されたクライアント ID は、dc_clt_cltout_s 関数を呼び出すまで破壊してはなりません。

ここで取得したクライアント ID は、この関数を発行したスレッド内でのみ使用できません。別のスレッドに引き渡して使用 (dc_rpc_call_s 関数などを発行するなど) した場合の動作は保証しません。

- defpath

クライアント環境定義ファイルへのパス名を指定します。パス名には完全パス、またはカレントドライブ・ディレクトリからの相対パスが指定できます。パス名を指定した場合のファイルの読み込み順序を次に示します。

- TP1/Client/P の場合

クライアント環境定義ファイルの読み込み順序は次のとおりです。

- 1.Windows ディレクトリの BETRAN.INI ファイル
- 2.引数 defpath に指定したクライアント環境定義ファイル

定義は、クライアント環境定義ファイルおよび BETRAN.INI ファイルのどちらのファイルに指定しても有効です。

両方のファイルに同じ定義を異なる値で指定した場合は、クライアント環境定義ファイルに指定した値が有効となります。

クライアント環境定義ファイルおよび BETRAN.INI ファイルのどちらにも指定がない場合は、デフォルト値で動作します。

- TP1/Client/W の場合

環境変数に指定されている定義は、すべて無効となります。引数 defpath に指定したクライアント環境定義ファイルに指定されていない定義はデフォルト値で動作します。

また、引数 defpath の先頭に NULL を指定することでパス名を省略できます。省略時の動作を次に示します。

- TP1/Client/P の場合

Windows ディレクトリの BETRAN.INI ファイルをクライアント環境定義ファイルとして動作します。BETRAN.INI ファイルがない場合、または定義ファイルの内容が不正な場合はデフォルト値で動作します。

- TP1/Client/W の場合

環境変数の指定で動作します。環境変数が指定されていない場合は、デフォルト値で動作します。

引数 defpath に指定したクライアント環境定義ファイルがない場合、または定義ファイルの内容が不正な場合の動作を次に示します。

- TP1/Client/P の場合

Windows ディレクトリの BETRAN.INI ファイルをクライアント環境定義ファイルとして動作します。BETRAN.INI ファイルがない場合、または定義ファイルの内容が不正な場合は、デフォルト値で動作します。

- TP1/Client/W の場合

デフォルト値で動作します。環境変数の指定は無効となります。

- target_host

認証要求を実行する時に窓口となる TP1/Server のホスト名およびポート番号を指定します。複数の窓口となる TP1/Server を指定できます（区切り文字は','を使用）。

ホスト名として指定できる長さは、63 文字*までです。ホスト名を複数指定する場合、引数 target_host に指定できる長さ（ポート番号なども含む）は、255 文字*までです。

形式

ホスト名 [:ポート番号] [,ホスト名 [:ポート番号] ,...]

- ホスト名 ~ 〈文字列〉
- ポート番号 ~ 〈符号なし整数〉 ((5001~65535))

区切り文字','の後ろ以外は空白文字（スペースまたはタブ）を入れないでください。

ホスト名として、10 進ドット記法の IP アドレスを指定することもできます。

ポート番号省略時は、クライアント環境定義 DCNAMPORT の値を仮定します。

引数 target_host に TP1/Server を二つ以上指定し、窓口となる TP1/Server の障害を検出した場合、クライアント環境定義 DCHOSTSELECT が N のとき、現在窓口となっている TP1/Server の次の TP1/Server を参照して切り替えを試みます。クライアント環境定義 DCHOSTSELECT が Y のとき、障害を検出した TP1/Server を除いて、窓口となる TP1/Server をランダムに選択して切り替えを試みます。

NULL が指定されていた場合は、クライアント環境定義 DCHOST を参照します。ただし、引数 target_host に NULL が指定されていて、DCHOST が設定されていなかった場合は、ブロードキャストを行い、窓口となる TP1/Server を決定します。TP1/Client/P でブロードキャストを行う場合、hosts ファイルにブロードキャストアドレスを指定する必要があります（ホスト名は broadcast としてください）。指定がなかった場合、dc_clt_cltin_s 関数は DCCLTER_SYSERR でエラーリターンします。

注※

クライアント環境定義 DCCLTOPTION に 00000008 を指定した場合、ホスト名として指定できる長さは、255 文字までです。ホスト名を複数指定する場合、引数 target_host に指定できる長さ（ポート番号なども含む）は 1023 文字までとなります。

- logname

クライアントユーザのログイン名を指定します。ログイン名として指定できる長さは、15 文字までです。TP1/Server 以外のサーバと通信する場合は、NULL 以外の任意の値を指定します。NULL を指定した場合、関数はエラーリターンします。

- passwd

logname で指定したログイン名に対するパスワードを指定します。パスワードとして指定できる長さは、15 文字までです。パスワードが設定されていない場合は、NULL を指定します。

- set_host

実際に認証要求をしたホストのホスト名を格納する 64 バイト※の領域へのポインタを指定します。NULL が指定された場合は、ホスト名を格納しません。

注※

クライアント環境定義 DCCLTOPTION に 00000008 を指定した場合、64 バイトではなく、256 バイトになります。

- flags

リモート API 機能を使用するためにユーザ認証を抑止する場合は、DCCLT_NO_AUTHENT を指定します。ユーザ認証を抑止しない場合は、DCNOFLAGS を指定します。

(4) 値が返される引数

- cltid

クライアント ID が返されます。

- set_host

ユーザ認証したサーバのホスト名（または 10 進ドット記法の IP アドレス）が返されます。ユーザ認証を抑止した場合は、返されません。

(5) リターン値

リターン値	数値 (10 進数)	意味
DC_OK	0	正常終了しました。
DCCLTER_INVALID_ARGS	-2501	引数が誤っています。
DCCLTER_PROTO	-2502	すでに dc_clt_cltin 関数が実行されています。dc_clt_cltin_s 関数実行時は、このリターン値は返りません。

リターン値	数値 (10進数)	意味
DCCLTER_FATAL	-2503	通信路の初期化に失敗しました。または、クライアント環境定義の指定が誤っています。
DCCLTER_NO_BUFS	-2504	必要なバッファが確保できませんでした。または、リソース不足が発生しました。
DCCLTER_NET_DOWN	-2506	通信障害が発生しました。
DCCLTER_OLTF_NOT_UP	-2515	設定したサービスがあるノードの OpenTP1 が起動されていません。
DCCLTER_SYSERR	-2518	システムエラーが発生しました。
DCCLTER_REJECT	-2527	指定されたログイン名が対象とするホストに登録されていないか、またはパスワードが一致しません。または、ユーザ認証機能をサポートしていない OpenTP1 サーバである可能性があります。システム共通定義 client_uid_check の指定が正しいか見直してください。
DCCLTER_PORT_IN_USE	-2547	指定したポート番号は使用されています。または、OS が自動的に割り当てるポート番号が不足しています。

(6) 注意事項

- TP1/Server 側が UNIX で、機密保護機構によって暗号化パスワードフィールドに*がセットされている場合、ユーザ認証を受けられないのでご注意ください。この場合、dc_clt_cltin_s 関数は DCCLTER_REJECT でエラーリターンします。
- dc_clt_cltin_s 関数がエラーリターンした場合、引数に指定した cltid は無効となるため、cltid を指定して TP1/Client の関数を発行できません。再度、dc_clt_cltin_s 関数から実行し直してください。
- 引数 set_host に値を受け取る場合は、64 バイト※以上の領域を指定してください。領域が 64 バイト※未満の場合、TP1/Client 内部の処理で領域破壊を起こすおそれがあります。なお、ユーザ認証を抑制した場合、値は格納されません。

注※

クライアント環境定義 DCCLTOPTION に 00000008 を指定した場合、64 バイトではなく、256 バイトになります。

- TP1/Client では、dc_clt_cltin_s 関数の呼び出しごとにクライアント環境定義を定義できます。dc_clt_cltin_s 関数の呼び出しごとにクライアント環境定義を定義するには、dc_clt_cltin_s 関数の呼び出しごとに、異なるファイルをクライアント環境定義ファイルとして作成して、そのファイル名を dc_clt_cltin_s 関数の引数 defpath に指定してください。

4.2.2 dc_clt_cltout_s – クライアントユーザの認証解除

(1) 形式

(a) _s 付き関数の場合

```
#include <dcvclt.h>
void dc_clt_cltout_s(DCCLT_ID cltid, DCLONG flags)
```

(b) _s 無し関数の場合

```
#include <dcvclt.h>
void dc_clt_cltout(DCLONG flags)
```

(2) 機能

クライアントユーザの認証を解除し、以降 OpenTP1 のサービスを受けられないようにします。

dc_clt_cltout_s 関数は、CUP を終了する前に必ず実行してください。dc_clt_cltout_s 関数は、dc_clt_cltin_s 関数と対になるように実行する必要があります。

(3) UAP で値を設定する引数

- cltid
dc_clt_cltin_s 関数で受け取ったクライアント ID を指定します。
- flags
DCNOFLAGS を指定します。

(4) 注意事項

引数 cltid に不正な値を指定した場合、CUP が異常終了または予期しない動作をするおそれがあります。

4.3 リモートプロシジャコール

4.3.1 dc_rpc_open_s – UAP の開始

(1) 形式

(a) `_s` 付き関数の場合

```
#include <dcvrpc.h>
DCLONG dc_rpc_open_s(DCCLT_ID cltid, DCLONG flags)
```

(b) `_s` 無し関数の場合

```
#include <dcvrpc.h>
int dc_rpc_open(DCLONG flags)
```

(2) 機能

OpenTP1 の SPP を呼び出すための環境, または TCP/IP 通信機能を使用するための環境を初期化します。

`dc_rpc_open_s` 関数は, RPC, トランザクション制御, トランザクション制御の各種関数を実行する前に実行してください。

(3) UAP で値を設定する引数

- `cltid`

`dc_clt_cltin_s` 関数で受け取ったクライアント ID を指定します。

- `flags`

初期化する環境を指定します。

DCNOFLAGS : SPP を呼び出すための環境

DCCLT_ONEWAY_SND : メッセージを一方送信するための環境

DCCLT_ONEWAY_RCV : メッセージを一方受信するための環境

DCCLT_SNDRCV : メッセージを送受信するための環境

DCNOFLAGS を指定した場合は, TCP/IP 通信機能を使用できません。

DCNOFLAGS 以外を指定した場合でも, RPC の機能は使用できます。

DCCLT_SNDRCV を指定した場合は, 同時に DCCLT_ONEWAY_SND, および

DCCLT_ONEWAY_RCV を指定してはなりません。

(4) リターン値

リターン値	数値 (10進数)	意味
DC_OK	0	正常終了しました。
DCRPCER_INVALID_ARGS	-2401	引数に指定した値が誤っています。
DCRPCER_PROTO	-2402	dc_rpc_open_s 関数はすでに実行されています。 または、dc_clt_cltin_s 関数が実行されていません。
DCRPCER_FATAL	-2403	次のどれかの要因で、このリターン値が戻りました。 ・初期化に失敗しました。 ・クライアント環境定義の指定が誤っています。
DCRPCER_PORT_IN_USE	-2447	指定したポート番号は使用されています。
DCCLTER_INVALID_CLTID	-2544	cltid に指定したクライアント ID は、dc_clt_cltin_s 関数で受け取ったクライアント ID と異なっています。

(5) 注意事項

次に示す場合は、dc_rpc_close_s 関数を実行した直後に、再び flags に DCCLT_ONEWAY_RCV を指定した dc_rpc_open_s 関数は実行できません。この場合、15 秒から 20 秒後に dc_rpc_open_s 関数を実行してください。

- flags に DCCLT_ONEWAY_RCV を指定した dc_rpc_open_s 関数を実行したあと、dc_clt_receive_s 関数を実行してメッセージを受信し、相手システムが接続を解放する前に CUP 側で dc_rpc_close_s 関数を実行して接続を解放した場合

4.3.2 dc_rpc_close_s – UAP の終了

(1) 形式

(a) _s 付き関数の場合

```
#include <dcvrpc.h>
void dc_rpc_close_s(DCCLT_ID cltid, DCLONG flags)
```

(b) _s 無し関数の場合

```
#include <dcvrpc.h>
void dc_rpc_close(DCLONG flags)
```

(2) 機能

OpenTP1 の SPP を呼び出すための環境，または TCP/IP 通信機能を使用するための環境を解除します。

dc_rpc_close_s 関数は，dc_rpc_open_s 関数と対になるように実行する必要があります。

dc_rpc_close_s 関数を実行したあとに実行できる関数を，次に示します。

- dc_rpc_open_s 関数
- dc_clt_cltout_s 関数

(3) UAP で値を設定する引数

- cltid
dc_clt_cltin_s 関数で受け取ったクライアント ID を指定します。
- flags
DCNOFLAGS を指定します。

(4) 注意事項

dc_rpc_close_s 関数は，リターン値を返しません。引数に不正な値が指定された場合は，環境は解除されないのので，注意してください。

4.3.3 dc_rpc_call_s – 遠隔サービスの要求

(1) 形式

(a) TP1/Client/W の場合

■ _s 付き関数

```
#include <dcvrpc.h>
DCLONG dc_rpc_call_s(DCCLT_ID cltid, char *group,
                    char *service, char *in,
                    DCULONG *in_len, char *out,
                    DCULONG *out_len, DCLONG flags)
```

■ _s 無し関数

```
#include <dcvrpc.h>
int dc_rpc_call(char *group, char *service,
               char *in, DCULONG *in_len,
               char *out, DCULONG *out_len, DCLONG flags)
```


(b) TP1/Client/P の場合

■ _s 付き関数

```
#include <dcvrpc.h>
DCLONG dc_rpc_call_s(DCCLT_ID cltid, char CLTFAR *group,
                    char CLTFAR *service, char CLTFAR *in,
                    DCULONG CLTFAR *in_len, char CLTFAR *out,
                    DCULONG CLTFAR *out_len, DCLONG flags)
```

■ _s 無し関数

```
#include <dcvrpc.h>
int dc_rpc_call(char CLTFAR *group, char CLTFAR *service,
               char CLTFAR *in, DCULONG CLTFAR *in_len,
               char CLTFAR *out, DCULONG CLTFAR *out_len,
               DCLONG flags)
```

(2) 機能

SPP のサービスを要求します。「サービスグループ名+サービス名」に該当するサービス関数を呼び、その応答を受け取ります。

サービス要求されたサーバ UAP が存在するノードの OpenTP1 は、稼働していなければなりません。OpenTP1 が稼働していない場合（開始処理中を含む）は、dc_rpc_call_s 関数は DCRPCER_NET_DOWN, DCRPCER_OLTF_NOT_UP, DCRPCER_OLTF_INITIALIZING でエラーリターンします。

dc_rpc_call_s 関数を実行したときに、目的のサービスグループが閉塞されている場合は、DCRPCER_SERVICE_CLOSED でエラーリターンします。

dc_rpc_call_s 関数を実行したときに、目的のサービスグループが dcsvstop コマンドなどで終了処理中、または終了している場合、DCRPCER_SERVICE_TERMINATING, DCRPCER_SERVICE_CLOSED, DCRPCER_NO_SUCH_SERVICE_GROUP のうちのどれかでエラーリターンします。どのリターン値が戻るかは、dc_rpc_call_s 関数を実行したタイミングで決まります。

ソケット受信型サーバでは、ユーザサービス定義の max_socket_msg と max_socket_msglen の指定でデータの輻輳制御をしています。そのため、サービス要求を受信できない場合があります。このとき、dc_rpc_call_s 関数は、DCRPCER_SERVER_BUSY でエラーリターンします。この値が戻った場合、CUP は適当な時間をおいてから再実行すれば、サービス要求できる場合があります。

XDM/DCCM3 と通常の通信形態で通信する場合、クライアント環境定義 DCCLTSERVICEGROUPLIST に XDM/DCCM3 論理端末のホスト名およびポート番号を指定し、dc_rpc_call_s 関数を実行します。

(a) 引数の設定

CUP ではサービス関数の応答の領域（out）を確保しておきます。さらに、CUP では dc_rpc_call_s 関数に次の値を指定します。

- 入力パラメタ (in)
- 入力パラメタ長 (in_len)
- 応答の長さ (out_len)

入力パラメタ、入力パラメタ長、応答の長さは、CUP の `dc_rpc_call_s` 関数で指定した値がそのままサービス関数に渡されます。応答を返さないサービス関数のサービスを呼ぶときは、応答の長さを指定しても無視されます。

入力パラメタ長と応答の長さの最大値 `DCRPC_MAX_MESSAGE_SIZE`*は、ヘッダファイル `dcrpc.h` に定義されています。

注※

クライアント環境定義 `DCCLTRPCMAXMSGSIZE` に 2 以上を指定した場合、`DCRPC_MAX_MESSAGE_SIZE` の値 (1 メガバイト) ではなく、クライアント環境定義 `DCCLTRPCMAXMSGSIZE` に指定した値になります。

(b) 引数の参照

サービス関数の処理終了後に、次の値が参照できます。

- サービス関数の応答 (out)
- サービス関数の応答の長さ (out_len)

`out_len` は、サービス関数から実際に返ってきた応答の長さです。

同期応答型 RPC (flags に `DCNOFLAGS` を設定) の場合、`dc_rpc_call_s` 関数がリターンしたあと、`out` と `out_len` を参照できます。非応答型 RPC (flags に `DCRPC_NOREPLY` を設定) の場合、`out` と `out_len` は参照できません。また、`dc_rpc_call_s` 関数がエラーリターンした場合も `out` と `out_len` は参照できません。

返ってきた応答が CUP で確保した応答の領域 (out) よりも大きい場合は、リターン値 `DCRPCER_REPLY_TOO_BIG` でエラーリターンします。

(3) UAP で値を設定する引数

- `cltid`
`dc_clt_cltin_s` 関数で受け取ったクライアント ID を指定します。
- `group`
サービスグループ名を 31 バイト以内の文字列で指定します。文字列の最後には、NULL 文字を指定してください。
- `service`
サービス名を 31 バイト以内の文字列で指定します。文字列の最後には、NULL 文字を指定してください。

- **in**

サービスの入力パラメタを指定します。

- **in_len**

サービスの入力パラメタ長を指定します。1 から DCRPC_MAX_MESSAGE_SIZE*までの範囲の長さが指定できます。

注※

クライアント環境定義 DCCLTRPCMAXMSGSIZE に 2 以上を指定した場合、DCRPC_MAX_MESSAGE_SIZE の値 (1 メガバイト) ではなく、クライアント環境定義 DCCLTRPCMAXMSGSIZE に指定した値になります。

- **out**

サービスの応答を受け取る領域を指定します。

- **out_len**

サービスの応答を受け取る領域の長さを指定します。1 から DCRPC_MAX_MESSAGE_SIZE*までの範囲の長さが指定できます。

注※

クライアント環境定義 DCCLTRPCMAXMSGSIZE に 2 以上を指定した場合、DCRPC_MAX_MESSAGE_SIZE の値 (1 メガバイト) ではなく、クライアント環境定義 DCCLTRPCMAXMSGSIZE に指定した値になります。

- **flags**

RPC の形態を指定します。

DCNOFLAGS : 同期応答型 RPC

DCRPC_NOREPLY : 非応答型 RPC

DCRPC_CHAINED : 連鎖 RPC

flags に DCNOFLAGS, または DCRPC_CHAINED を指定すると、応答が返されてくるか、または応答待ち時間 (クライアント環境定義 DCWATCHTIM の値) 切れ (タイムアウト) エラーになるまで、dc_rpc_call_s 関数は戻りません。ただし、サービス要求先の SPP がアボートした場合は、即時にエラーリターンします。

この場合、DCWATCHTIM で指定された応答待ち時間によって、次の二つのリターン値が返されます。

- DCWATCHTIM に 1~65535 を指定した場合 : DCRPCER_TIMED_OUT
- DCWATCHTIM に 0 (無限に待つ) を指定した場合 : DCRPCER_SERVICE_NOT_UP

また、CUP 実行中に応答待ち時間を変更することもできます。この場合、dc_rpc_call_s 関数を実行する前に、dc_rpc_set_watch_time_s 関数を実行してください。

なお、トランザクション中、または常設コネクション中に限り、DCRPC_CHAINED を指定できます。

flags に DCRPC_NOREPLY を指定すると、要求したサービスは応答を返さないサービスとみなされます。この場合、dc_rpc_call_s 関数はサービスの実行終了を待たないで、すぐに戻ります。この指定をした場合は、応答 (out) と応答の長さ (out_len) は参照できません。さらに、サービス関数が実行されたかどうかは、CUP ではわかりません。

トランザクションの処理からの RPC を、トランザクションとしないサービス要求にできます。RPC の形態を示すパラメタに DCRPC_TPNOTRAN を合わせて指定すると、該当する dc_rpc_call_s 関数のサービス要求は、トランザクションの処理でないサービス要求になります。

(例) DCNOFLAGS|DCRPC_TPNOTRAN

このサービス要求は、トランザクションの処理からだけ指定できます。トランザクションの範囲外での指定をした場合、dc_rpc_call_s 関数は DCRPCER_INVALID_ARGS でエラーリターンします。

(4) 値が返される引数

- out

サービス関数で指定したサービスの応答が返されます。引数 flags に DCRPC_NOREPLY を指定した場合は、返されません。

- out_len

サービス関数で指定したサービスの応答の長さが返されます。引数 flags に DCRPC_NOREPLY を指定した場合は、返されません。

(5) リターン値

リターン値	数値 (10 進数)	意味
DC_OK	0	正常終了しました。
DCRPCER_INVALID_ARGS	-2401	引数に指定した値が誤っています。
DCRPCER_PROTO	-2402	dc_rpc_open_s 関数が実行されていません。
DCRPCER_NO_BUFS	-2404	必要なバッファが確保できませんでした。または、リソース不足が発生しました。
DCRPCER_NET_DOWN	-2406	ネットワーク障害が発生しました。
DCRPCER_TIMED_OUT	-2407	dc_rpc_call_s 関数の処理で時間切れ (タイムアウト) が発生しました。または、サービス要求した SPP が処理完了前に異常終了しました。
DCRPCER_MESSAGE_TOO_BIG	-2408	入力パラメタ長が最大値を超えました。
DCRPCER_REPLY_TOO_BIG	-2409	返ってきた応答の長さが、CUP で用意した領域を超えています。
DCRPCER_NO_SUCH_SERVICE_GROUP	-2410	次のどちらかの要因が考えられます。 <ul style="list-style-type: none"> • group に指定したサービスグループ名は定義されていません。 • TP1/Server と通信するにもかかわらず、クライアント環境定義 DCCLTNOSERVER に Y が指定されています。
DCRPCER_NO_SUCH_SERVICE	-2411	service に指定したサービス名は定義されていません。

リターン値	数値 (10進数)	意味
DCRPCER_SERVICE_CLOSED	-2412	service に指定したサービスが存在するサービスグループは、閉塞されています。
DCRPCER_SERVICE_TERMINATING	-2413	指定したサービスは終了処理中です。
DCRPCER_SERVICE_NOT_UP	-2414	サービス要求先の SPP が未起動であるか、または処理完了前に異常終了しました。この値はクライアント環境定義 DCWATCHTIM に 0 を指定 (応答無限待ち指定) した場合は戻ります。
DCRPCER_OLTF_NOT_UP	-2415	指定したサービスが存在するノードの OpenTP1 が実行されていません。または、トランザクション処理中、サーバとのコネクションが切断されているため、通信できません。
DCRPCER_SYSERR_AT_SERVER	-2416	指定したサービスでシステムエラーが発生しました。
DCRPCER_NO_BUFS_AT_SERVER	-2417	指定したサービスでメモリ不足が発生しました。
DCRPCER_SYSERR	-2418	システムエラーが発生しました。
DCRPCER_INVALID_REPLY	-2419	サービス関数が OpenTP1 に返した応答長が、1 から DCRPC_MAX_MESSAGE_SIZE*までの範囲にありません。
DCRPCER_OLTF_INITIALIZING	-2420	サービス要求されたノードにある OpenTP1 は、開始処理中です。
DCRPCER_NO_BUFS_RB	-2423	メモリ不足が発生しました。
DCRPCER_SYSERR_RB	-2424	システムエラーが発生しました。
DCRPCER_SYSERR_AT_SERVER_RB	-2425	指定したサービスでシステムエラーが発生しました。
DCRPCER_REPLY_TOO_BIG_RB	-2426	返ってきた応答が、CUP で用意した領域に収まりません。
DCRPCER_TRNCHK	-2427	ノード間負荷バランス機能の環境で、複数の SPP のトランザクション属性が一致していません。または、負荷を分散する先のノードにある OpenTP1 のバージョンが古い場合、ノード間負荷バランス機能を実行できません。このリターン値は、ノード間負荷バランス機能を使用している SPP にサービスを要求した場合にだけ戻ります。
DCRPCER_CONNFREE	-2442	常設コネクションが解放されました。
DCRPCER_SERVER_BUSY	-2456	サービス要求先のソケット受信型サーバが、サービス要求を受信できません。
DCRPCER_TESTMODE	-2466	クライアント環境定義 DCUTOKEY を指定している環境下で、ユーザサービス定義で test_mode=no と指定した SPP に対してサービス要求しました。 または、次の条件が重なった環境下から関数を呼び出しています。 <ul style="list-style-type: none"> クライアント環境定義 DCUTOKEY を指定しています。 CUP 実行プロセスとの常設コネクションが確立中です。 トランザクションの範囲外です。

4. TP1/Client で使用できる関数 (C 言語編)

リターン値	数値 (10進数)	意味
DCRPCER_TESTMODE	-2466	<ul style="list-style-type: none"> ユーザサービス定義で test_mode=no 以外を指定した SPP に対してサービス要求をしました。
DCRPCER_NOT_TRN_EXTEND	-2467	トランザクション処理の連鎖 RPC を使ったあとで、flags に DCRPC_TPNOTRAN を設定した dc_rpc_call_s 関数でサービスを要求しています。
DCRPCER_SECCHK	-2470	サービス要求先の SPP は、セキュリティ機能で保護されています。dc_rpc_call_s 関数を呼び出した UAP には、サーバ UAP へのアクセス権がありません。
DCRPCER_TRNCHK_EXTEND	-2472	次のどれかの要因が考えられます。 <ul style="list-style-type: none"> 同時に起動できるトランザクションブランチの数を越えたため、トランザクションブランチを開始できません。 一つのトランザクションブランチから開始できる子トランザクションブランチの最大数を越えたため、トランザクションブランチを開始できません。 トランザクション内でドメイン修飾をしたサービス要求で、flags に DCRPC_TPNOTRAN を設定していません。
DCRPCER_SERVICE_TERMINATED	-2478	サービス要求先の SPP が処理完了前に異常終了しました。この値はクライアント環境定義 DCEXTENDFUNCTION に 00000001 を指定した場合に戻ります。00000000 を指定しているか指定を省略していると、リターン値には DCRPCER_TIMED_OUT、または DCRPCER_SERVICE_NOT_UP が戻ります。
DCRPCER_VERSION_CHECK	-2479	サービス要求先の TP1/Server Base のバージョンが古い (03-03 以降でない) ため、データ圧縮機能は使用できません。このリターン値は、トランザクションの範囲内でサービス要求した場合に戻ります。
DCCLTER_INVALID_CLTID	-2544	cltid に指定したクライアント ID は、dc_clt_cltin_s 関数で受け取ったクライアント ID と異なります。
DCRPCER_PORT_IN_USE	-2547	指定したポート番号は使用されています。または、OS が自動的に割り当てるポート番号が不足しています。

注※

クライアント環境定義 DCCLTRPCMAXMSGSIZE に 2 以上を指定した場合、DCRPC_MAX_MESSAGE_SIZE の値 (1 メガバイト) ではなく、クライアント環境定義 DCCLTRPCMAXMSGSIZE に指定した値になります。

(6) 注意事項

- 入力パラメタ (in), およびサービス関数の応答 (out) に同じバッファを指定しないでください。
- flags に DCRPC_NOREPLY を指定した場合、次のリターン値は戻りません。

発生しないエラー

DCRPCER_REPLY_TOO_BIG

DCRPCER_INVALID_REPLY

発生しても検出できないエラー

DCRPCER_NO_SUCH_SERVICE

DCRPCER_SERVICE_CLOSED

DCRPCER_SERVICE_TERMINATING

DCRPCER_SYSERR_AT_SERVER

DCRPCER_NO_BUFS_AT_SERVER

DCRPCER_OLTF_INITIALIZING

- リターン値 DCRPCER_TIMED_OUT が戻る場合、次に示す要因が考えられます。

クライアント環境定義で指定した最大応答待ち時間が短い

サービス要求先の SPP から発行したサービス関数の異常終了

サービス要求先の SPP が存在するノードの障害

サービス要求先の SPP の処理完了前での異常終了

ネットワーク障害

上記の場合、サービス要求先の SPP から開始したトランザクションの処理はコミットされて、データベースが更新されていることがあります。データベースが更新されているかどうか確認してください。

- CUP から dc_trn_begin_s 関数を実行後、dc_rpc_call_s 関数を実行して、次に示すリターン値が戻った場合は、必要があればロールバック要求の関数を実行してください。

DCRPCER_TIMED_OUT

DCRPCER_NO_SUCH_SERVICE

DCRPCER_NO_BUFS_AT_SERVER

DCRPCER_INVALID_REPLY

DCRPCER_NO_BUFS_RB

DCRPCER_SYSERR_RB

DCRPCER_SYSERR_AT_SERVER_RB

DCRPCER_REPLY_TOO_BIG_RB

4.3.4 dc_rpc_call_to_s – 通信先を指定した遠隔サービスの要求

(1) 形式

(a) TP1/Client/W の場合

■ _s 付き関数

```
#include <dcvrpc.h>
DCLONG dc_rpc_call_to_s(
    DCCLT_ID cltid, struct DCRPC_BINDING_TBL *direction,
```

```
char *group, char *service, char *in,  
DCULONG *in_len, char *out,  
DCULONG *out_len,  
DCLONG flags)
```

■ `_s` 無し関数

```
#include <dcvrpc.h>  
DCLONG dc_rpc_call_to(  
    struct DCRPC_BINDING_TBL *direction,  
    char *group, char *service,  
    char *in, DCULONG *in_len, char *out,  
    DCULONG *out_len, DCLONG flags)
```

(b) TP1/Client/P の場合

■ `_s` 付き関数

```
#include <dcvrpc.h>  
DCLONG dc_rpc_call_to_s(  
    DCCLT_ID cltid, struct DCRPC_BINDING_TBL CLTFAR *direction,  
    char CLTFAR *group, char CLTFAR *service, char CLTFAR *in,  
    DCULONG CLTFAR *in_len, char CLTFAR *out,  
    DCULONG CLTFAR *out_len,  
    DCLONG flags)
```

■ `_s` 無し関数

```
#include <dcvrpc.h>  
DCLONG dc_rpc_call_to(  
    struct DCRPC_BINDING_TBL CLTFAR *direction, char CLTFAR *group,  
    char CLTFAR *service, char CLTFAR *in, DCULONG CLTFAR *in_len,  
    char CLTFAR *out, DCULONG CLTFAR *out_len, DCLONG flags)
```

(2) 機能

`dc_rpc_call_s` 関数と同様に、SPP のサービスを要求します。`dc_rpc_call_to_s` 関数では、サービスグループ名とサービス名に加え、ホスト名を該当するサービス関数の検索のキーとして使用し、サービスの要求先を限定します。

この関数を発行する前に `DCRPC_DIRECT_SCHEDULE()` を発行し、`DCRPC_BINDING_TBL` 構造体を作成しておく必要があります。引数 `direction` に `DCRPC_BINDING_TBL` 構造体へのアドレスを指定します。それ以外のインタフェースは、`dc_rpc_call_s` 関数と同じです。

(3) UAP で値を設定する引数

- `cltid`
`dc_clt_cltin_s` 関数で受け取ったクライアント ID を指定します。
- `direction`

DCRPC_BINDING_TBL 構造体のアドレスを指定します。

この関数を発行する前に DCRPC_DIRECT_SCHEDULE() を発行し、DCRPC_BINDING_TBL 構造体に値を設定しておきます。

- **group**

サービスグループ名を 31 バイト以内の文字列で指定します。文字列の最後には、NULL 文字を指定してください。

- **service**

サービス名を 31 バイト以内の文字列で設定します。文字列の最後には、NULL 文字を指定してください。

- **in**

サービスの入力パラメタを指定します。

- **in_len**

サービスの入力パラメタ長を指定します。1 から DCRPC_MAX_MESSAGE_SIZE* までの範囲の長さが指定できます。

注※

クライアント環境定義 DCCLTRPCMAXMSGSIZE に 2 以上を指定した場合、DCRPC_MAX_MESSAGE_SIZE の値 (1 メガバイト) ではなく、クライアント環境定義 DCCLTRPCMAXMSGSIZE に指定した値になります。

- **out**

サービスの応答を格納する領域のアドレスを指定します。

- **out_len**

サービスの応答の長さを指定します。1 から DCRPC_MAX_MESSAGE_SIZE* までの範囲の長さが指定できます。

注※

クライアント環境定義 DCCLTRPCMAXMSGSIZE に 2 以上を指定した場合、DCRPC_MAX_MESSAGE_SIZE の値 (1 メガバイト) ではなく、クライアント環境定義 DCCLTRPCMAXMSGSIZE に指定した値になります。

- **flags**

RPC の形態を設定します。

DCNOFLAGS : 同期応答型 RPC

DCRPC_NOREPLY : 非応答型 RPC

flags に DCNOFLAGS を指定すると、応答が返されてくるか、または応答待ち時間 (クライアント環境定義 DCWATCHTIM の値) 切れ (タイムアウト) エラーになるまで、dc_rpc_call_to_s 関数は戻りません。ただし、サービス要求先の SPP がアボートした場合は、即時にエラーリターンします。

この場合、クライアント環境定義 DCWATCHTIM で指定された応答待ち時間によって、次の二つのリターン値が返されます。

- DCWATCHTIM に 1~65535 を指定した場合：DCRPCER_TIMED_OUT
- DCWATCHTIM に 0（無限に待つ）を指定した場合：DCRPCER_SERVICE_NOT_UP

また、CUP 実行中に応答待ち時間を変更することもできます。この場合、dc_rpc_call_to_s 関数を実行する前に、dc_rpc_set_watch_time_s 関数を実行してください。

flags に DCRPC_NOREPLY を指定すると、要求したサービスは応答を返さないサービスとみなされます。この場合、dc_rpc_call_to_s 関数はサービスの実行終了を待たないで、すぐに戻ります。この指定をした場合は、応答 (out) と応答の長さ (out_len) は参照できません。さらに、サービス関数が実行されたかどうかは、CUP ではわかりません。

(4) 値が返される引数

- out
サービス関数で指定したサービスの応答を格納する領域に応答が返されます。引数 flags に DCRPC_NOREPLY を指定したときは、返されません。
- out_len
サービス関数で指定したサービスの応答の長さが返されます。引数 flags に DCRPC_NOREPLY を指定したときは、返されません。

(5) リターン値

リターン値	数値 (10 進数)	意味
DC_OK	0	正常終了しました。
DCRPCER_INVALID_ARGS	-2401	引数に指定した値が誤っています。
DCRPCER_PROTO	-2402	dc_rpc_open_s 関数が実行されていません。 または、常設コネクション確立中にこの関数を発行しています。 または、トランザクションの範囲内でこの関数を発行しています。
DCRPCER_NO_BUFS	-2404	必要なバッファが確保できませんでした。または、リソース不足が発生しました。
DCRPCER_NET_DOWN	-2406	ネットワーク障害が発生しました。
DCRPCER_TIMED_OUT	-2407	dc_rpc_call_to_s 関数の処理で時間切れ（タイムアウト）が発生しました。または、サービス要求した SPP が処理完了前に異常終了しました。
DCRPCER_MESSAGE_TOO_BIG	-2408	入力パラメタ長が最大値を超えました。
DCRPCER_REPLY_TOO_BIG	-2409	返ってきた応答の長さが、CUP で用意した領域を超えています。
DCRPCER_NO_SUCH_SERVICE_GROUP	-2410	group に指定したサービスグループ名は定義されていません。

リターン値	数値 (10進数)	意味
DCRPCER_NO_SUCH_SERVICE_GROUP	-2410	または、ソケット受信型（ユーザサービス定義 receive_from=socket 指定）のユーザサーバへサービス要求を行いました。 または、クライアント環境定義 DCCLTONLYTHISNODE の指定が N のとき、サービス要求先の SPP が未起動です。
DCRPCER_NO_SUCH_SERVICE	-2411	service に指定したサービス名は定義されていません。
DCRPCER_SERVICE_CLOSED	-2412	service に指定したサービスが存在するサービスグループは、閉塞されています。
DCRPCER_SERVICE_TERMINATING	-2413	指定したサービスは終了処理中です。
DCRPCER_SERVICE_NOT_UP	-2414	クライアント環境定義 DCCLTONLYTHISNODE の指定が Y のとき、サービス要求先の SPP が未起動です。 または、クライアント環境定義 DCWATCHTIM の指定が 0 のとき、処理完了前に異常終了しました。
DCRPCER_OLTF_NOT_UP	-2415	指定したサービスが存在するノードの OpenTP1 が実行されていません。
DCRPCER_SYSERR_AT_SERVER	-2416	指定したサービスでシステムエラーが発生しました。
DCRPCER_NO_BUFS_AT_SERVER	-2417	指定したサービスでメモリ不足が発生しました。
DCRPCER_SYSERR	-2418	システムエラーが発生しました。
DCRPCER_INVALID_REPLY	-2419	サービス関数が OpenTP1 に返した応答長が、1 から DCRPC_MAX_MESSAGE_SIZE*までの範囲にありません。
DCRPCER_OLTF_INITIALIZING	-2420	サービス要求されたノードにある OpenTP1 は、開始処理中です。
DCRPCER_TRNCHK	-2427	負荷を分散する先のノードにある OpenTP1 のバージョンが古いため、ノード間負荷バランス機能を実行できません。このリターン値は、ノード間負荷バランス機能を使用している SPP にサービスを要求した場合にだけ戻ります。
DCRPCER_TESTMODE	-2466	ユーザサービス定義で test_mode=no と指定した SPP に対してサービス要求しました。
DCRPCER_SECCHK	-2470	サービス要求先の SPP は、セキュリティ機能で保護されています。dc_rpc_call_to_s 関数を呼び出した UAP には、サーバ UAP へのアクセス権限がありません。
DCRPCER_SERVICE_TERMINATED	-2478	サービス要求先の SPP が処理完了前に異常終了しました。この値はクライアント環境定義 DCEXTENDFUNCTION に 00000001 を指定した場合に戻ります。00000000 を指定しているか指定を省略していると、リターン値には DCRPCER_TIMED_OUT、または DCRPCER_SERVICE_NOT_UP が戻ります。
DCCLTER_INVALID_CLTID	-2544	cltid に指定したクライアント ID は、dc_clt_cltin_s 関数で受け取ったクライアント ID と異なっています。

リターン値	数値 (10進数)	意味
DCRPCER_PORT_IN_USE	-2547	指定したポート番号は使用されています。または、OSが自動的に割り当てるポート番号が不足しています。

注※

クライアント環境定義 DCCLTRPCMAXMSGSIZE に 2 以上を指定した場合、DCRPC_MAX_MESSAGE_SIZE の値 (1 メガバイト) ではなく、クライアント環境定義 DCCLTRPCMAXMSGSIZE に指定した値になります。

(6) 注意事項

- ソケット受信型 (ユーザサービス定義 receive_from=socket 指定) のユーザサーバへサービス要求を行った場合は、この関数は、DCRPCER_NO_SUCH_SERVICE_GROUP でエラーリターンします。
- サービス要求先の OpenTP1 のバージョンは、03-02 以降でなければなりません。これ以前の OpenTP1 をサービス要求先とした場合は、動作の保証はできません。
- 常設コネクション確立中、またはトランザクションの範囲内でこの関数を発行した場合、この関数は DCRPCER_PROTO でエラーリターンします。
- クライアント環境定義 DCCACHE, DCCLTCACHETIM に値を指定しても有効になりません。
- スケジュールサービスに直接送信するため、クライアント環境定義 DCCLTLOADBALANCE に値を指定しても有効になりません。
- dc_rpc_call_to_s 関数発行時は、クライアント環境定義 DCCLTSERVICEGROUPLIST は参照しません。
- dc_rpc_call_to_s 関数発行時は、クライアント環境定義 DCSCDDIRECT, DCSCDPORT, DCSCDMULTI および DCSCDMULTICOUNT は参照しません。
- クライアント環境定義 DCCLTONLYTHISNODE の指定値が N, または指定値を省略したときは、サービス要求を受け付けたスケジュールサービスのノード優先で負荷分散を行います。Y のときは、負荷分散を行いません。
- サービス要求先のホスト名に誤りがあった場合、この関数は DCRPCER_INVALID_ARGS でエラーリターンします。
- クライアント環境定義 DCCLTONLYTHISNODE の指定値によって、この関数の動作に次のような相違があります。

SPP の状態		DCCLTONLYTHISNODE 指定値	
指定ノード	他ノード	N (省略値)	Y
負荷が低い	—	指定ノードの SPP にスケジュールされる。	指定ノードの SPP にスケジュールされる。
負荷が高い、または閉塞 (受付可)	負荷が低い	他ノードの SPP にスケジュールされる。	
	負荷が高い	指定ノードの SPP にスケジュールされる。	

SPP の状態		DCCLTONLYTHISNODE 指定値	
指定ノード	他ノード	N (省略値)	Y
負荷が高い, または 閉塞 (受付可)	閉塞	指定ノードの SPP にスケジュールされる。	指定ノードの SPP にスケジュールされる。
	停止		
閉塞 (受付不可)	起動	他ノードの SPP にスケジュールされる。	関数は, DCRPCER_SERVICE_CLOSED でエラーリターンする。
	閉塞 (受付可)		
	閉塞 (受付不可)	関数は, DCRPCER_SERVICE_CLOSED でエラーリターンする。	
	停止		
停止	起動	他ノードの SPP にスケジュールされる。	関数は, DCRPCER_SERVICE_NOT_UP でエラーリターンする。
	閉塞 (受付可)		
	閉塞 (受付不可)	関数は, DCRPCER_NO_SUCH_SERVICE_GROUP でエラーリターンする。	
	停止		

(凡例)

— : 該当しません。

注

負荷が低い状態とは、負荷レベルが LEVEL0, または LEVEL1 のときを指します。

負荷が高い状態とは、負荷レベルが LEVEL2 のときを指します。

4.3.5 dc_rpc_set_watch_time_s – サービス応答待ち時間の更新

(1) 形式

(a) _s 付き関数の場合

```
#include <dcrpc.h>
DCLONG dc_rpc_set_watch_time_s(DCCLT_ID cltid, DCLONG var)
```

(b) _s 無し関数の場合

```
#include <dcrpc.h>
DCLONG dc_rpc_set_watch_time(DCLONG var)
```

(2) 機能

サービス要求の応答待ち時間を変更します。この関数を使用した場合、以降の dc_rpc_call_s 関数では、この関数で指定したサービス要求の応答待ち時間が使用されます。この関数で指定した値は、dc_rpc_close_s

関数を実行するまで有効です。ただし、この関数は、クライアント環境定義 DCWATCHTIM に指定した値を変更しません。

dc_rpc_set_watch_time_s 関数で値を変更する前に dc_rpc_get_watch_time_s 関数を実行して、変更前の値を記録しておく必要があります。

(3) UAP で値を設定する引数

- cltid

dc_clt_cltin_s 関数で受け取ったクライアント ID を指定します。

- var

変更後のサービス応答待ち時間を設定します。1 から 65535 の範囲で設定します。無制限に待ち続ける場合は、0 を設定します。

(4) リターン値

リターン値	数値 (10 進数)	意味
DC_OK	0	正常終了しました。
DCRPCER_INVALID_ARGS	-2401	var に設定した値が間違っています。
DCRPCER_PROTO	-2402	dc_rpc_open_s 関数が実行されていません。
DCRPCER_NO_BUFS	-2404	メモリ不足が発生しました。
DCRPCER_INVALID_CLTID	-2544	cltid に指定したクライアント ID は、dc_clt_cltin_s 関数で受け取ったクライアント ID と異なります。

4.3.6 dc_rpc_get_watch_time_s – サービス応答待ち時間の参照

(1) 形式

(a) _s 付き関数の場合

```
#include <dcvrpc.h>
DCLONG dc_rpc_get_watch_time_s(DCCLT_ID cltid)
```

(b) _s 無し関数の場合

```
#include <dcvrpc.h>
DCLONG dc_rpc_get_watch_time()
```

(2) 機能

現在のサービス要求の応答待ち時間を参照します。この関数は、dc_rpc_set_watch_time_s 関数でサービス応答待ち時間を一時的に変更する前に、元の値を退避するために使用します。

この関数は、dc_rpc_set_watch_time_s 関数で変更したサービス応答待ち時間をリターンします。変更していない場合は、クライアント環境定義 DCWATCHTIM の値をリターンします。

この関数で得られる値は、dc_rpc_call_s 関数に対して有効です。

(3) UAP で値を設定する引数

- cltid

dc_clt_cltin_s 関数で受け取ったクライアント ID を指定します。

(4) リターン値

リターン値	数値 (10 進数)	意味
—	正の整数	現在のサービス応答待ち時間を示します。
—	0	サービス応答待ち時間は、無制限に待ち続ける指定です。
DCRPCER_PROTO	-2402	dc_rpc_open_s 関数が実行されていません。
DCRPCER_NO_BUFS	-2404	メモリ不足が発生しました。
DCRPCER_INVALID_CLTID	-2544	cltid に指定したクライアント ID は、dc_clt_cltin_s 関数で受け取ったクライアント ID と異なっています。

(凡例)

— : 該当しません。

4.3.7 DCRPC_DIRECT_SCHEDULE – DCRPC_BINDING_TBL 構造体の作成

(1) 形式

(a) TP1/Client/W の場合

```
#include <dcvrpc.h>
DCRPC_DIRECT_SCHEDULE(
    struct DCRPC_BINDING_TBL *direction,
    char *hostnm,
    unsigned short scdport, DCLONG flags)
```

(b) TP1/Client/P の場合

```
#include <dcvrpc.h>
DCRPC_DIRECT_SCHEDULE(
    struct DCRPC_BINDING_TBL CLTFAR *direction,
    char CLTFAR *hostnm,
    unsigned short scdport, DCLONG flags)
```

(2) 機能

dc_rpc_call_to_s 関数の引数 direction に指定する DCRPC_BINDING_TBL 構造体を作成します。

(3) UAP で値を設定する引数

- direction

DCRPC_BINDING_TBL 構造体のアドレスを指定します。DCRPC_BINDING_TBL 構造体はスレッドごとに作成してください。

- hostnm

サービス要求先のホスト名を格納した領域のアドレスを指定します。文字列の最後には、NULL 文字を指定してください。

NULL が指定された場合、続いて発行する dc_rpc_call_to_s 関数は、DCRPCER_INVALID_ARGS でエラーリターンします。

ホスト名として指定できる長さは、63 文字*までです。

ホスト名として、10 進ドット記法の IP アドレスを指定することもできます。

注※

クライアント環境定義 DCCLTOPTION に 00000008 を指定した場合、ホスト名に指定できる長さは 255 文字までとなります。

- scdport

サービス要求先のホストで起動しているスケジューラのポート番号を 0、または 5001 から 65535 の範囲で指定します。

0 が指定された場合、続いて発行する dc_rpc_call_to_s 関数は、窓口ホストのネームサービスに、hostnm に指定したサービス要求先ホストのスケジュールサービスのポート番号を問い合わせます。このため、窓口ホストの TP1/Server と、hostnm に指定したサービス要求先ホストの TP1/Server は、OpenTP1 のドメイン構成（システム共通定義の all_node オペランドで指定）である必要があります。

- flags

DCNOFLAGS を指定します。

4.4 常設コネクション

4.4.1 dc_clt_connect_s – 常設コネクションの確立

(1) 形式

(a) `_s` 付き関数の場合

```
#include <dcvclt.h>
DCLONG dc_clt_connect_s(DCCLT_ID cltid, DCLONG flags)
```

(b) `_s` 無し関数の場合

```
#include <dcvclt.h>
DCLONG dc_clt_connect(DCLONG flags)
```

(2) 機能

CUP 実行プロセス, rap サーバまたは DCCM3 の論理端末との間に常設コネクションを確立します。

常設コネクションを確立する CUP 実行プロセスが起動されている OpenTP1 ノードは, `dc_clt_cltin_s` 関数の引数 `target_host` に指定した OpenTP1 ノード, クライアント環境定義 `DCCLTRAPHOST` または `DCHOST` に指定した OpenTP1 ノードです。

DCCM3 の論理端末との間に常設コネクションを確立する場合, クライアント環境定義に `DCCLTDCCMHOST` および `DCCLTDCCMPORT` を定義し, `dc_clt_connect_s` 関数の引数 `flags` に `DCCLT_DCCM3` を指定します。

また, リモート API 機能を使用する場合, DCCM3 の論理端末との間に常設コネクションを確立するには, `DCCLTRAPHOST` に DCCM3 の論理端末のホスト名およびポート番号を指定し, `dc_clt_connect_s` 関数の引数 `flags` に `DCNOFLAGS` を指定します。

(3) UAP で値を設定する引数

- `cltid`

`dc_clt_cltin_s` 関数で受け取ったクライアント ID を指定します。

- `flags`

常設コネクションを確立する通信相手を指定します。

`DCNOFLAGS`

CUP 実行プロセス, rap サーバまたは DCCM3 の論理端末との間に常設コネクションを確立します。

DCCLT_DCCM3

DCCM3 の論理端末との間に常設コネクションを確立します。

(4) リターン値

リターン値	数値 (10 進数)	意味
DC_OK	0	正常終了しました。または、すでに常設コネクションが確立されています。
DCCLTER_INVALID_ARGS	-2501	引数が誤っています。
DCCLTER_PROTO	-2502	次のどれかの要因が考えられます。 <ul style="list-style-type: none">• トランザクション内で発行されています。• dc_rpc_open_s 関数が発行されていません。• OpenTP1 に対する確立要求が発行されましたが、すでに DCCM3 との常設コネクションが確立されています。• DCCM3 に対する確立要求が発行されましたが、すでに OpenTP1 との常設コネクションが確立されています。
DCCLTER_NO_BUFS	-2504	必要なバッファが確保できませんでした。または、リソース不足が発生しました。
DCCLTER_NET_DOWN	-2506	通信障害が発生しました。
DCCLTER_TIMED_OUT	-2507	常設コネクション確立時に時間切れ（タイムアウト）が発生しました。
DCCLTER_OLTF_NOT_UP	-2515	次のどれかの要因が考えられます。 <ul style="list-style-type: none">• OpenTP1 サーバ、または DCCM3 論理端末が起動されていません。• クライアント拡張サービスが起動されていません。システムサービス構成定義 clt_conf の指定が正しくありません。• CUP 実行プロセスが起動されていません。クライアントサービス定義 clt_cup_conf の指定が正しくありません。
DCCLTER_SYSERR	-2518	システムエラーが発生しました。
DCCLTER_WRONG_HOST	-2539	DCCM3 の論理端末に対する確立要求が発行されましたが、ホスト名が不正です。
DCCLTER_INVALID_CLTID	-2544	cltid に指定したクライアント ID は、dc_clt_cltin_s 関数で受け取ったクライアント ID と異なっています。
DCCLTER_PORT_IN_USE	-2547	指定したポート番号は使用されています。または、OS が自動的に割り当てるポート番号が不足しています。

(5) 注意事項

- dc_clt_connect_s 関数がエラーリターンした場合、常設コネクションは確立されません。ただし、次のリターン値が返された場合、CUP 実行プロセスまたは DCCM3 の論理端末側では常設コネクションが確立状態になることがあります。

- DCCLTER_NET_DOWN
- DCCLTER_TIMED_OUT
- DCCLTER_SYSERR

このとき、CUP 実行プロセスまたは DCCM3 の論理端末は、CUP からの要求を待ち続ける場合があります。要求待ちを避けるため、常設コネクション問い合わせ間隔最大時間（DCCM3 論理端末の場合は端末放置監視時間）に適切な値を設定してください。

- dc_clt_connect_s 関数をトランザクション内で発行することはできません。
- CUP 実行プロセス、rap サーバ、またはクライアント環境定義 DCCLTRAPHOST に指定した DCCM3 の論理端末との間に常設コネクションを確立した場合は、dc_clt_disconnect_s 関数を呼び出すまでクライアント環境定義 DCCLTDCCMHOST に指定した DCCM3 の論理端末との通信はできません。

また、クライアント環境定義 DCCLTDCCMHOST に指定した DCCM3 の論理端末との間に常設コネクションを確立した場合は、dc_clt_disconnect_s 関数を呼び出すまで CUP 実行プロセス、rap サーバ、またはクライアント環境定義 DCCLTRAPHOST に指定した DCCM3 の論理端末との通信はできません。

- DCCM3 の論理端末との間に常設コネクションを確立する場合、データ圧縮機能は使用できません。クライアント環境定義 DCCLTDATACOMP の指定を省略するか、または DCCLTDATACOMP に N を指定する必要があります。

4.4.2 dc_clt_disconnect_s – 常設コネクションの解放

(1) 形式

(a) _s 付き関数の場合

```
#include <dcvclt.h>
DCLONG dc_clt_disconnect_s(DCCLT_ID cltid, DCLONG flags)
```

(b) _s 無し関数の場合

```
#include <dcvclt.h>
DCLONG dc_clt_disconnect(DCLONG flags)
```

(2) 機能

CUP 実行プロセス, rap サーバまたは DCCM3 の論理端末との間の常設コネクションを解放します。

(3) UAP で値を設定する引数

- `cltid`
dc_clt_cltin_s 関数で受け取ったクライアント ID を指定します。
- `flags`
DCNOFLAGS を指定します。

(4) リターン値

リターン値	数値 (10 進数)	意味
DC_OK	0	正常終了しました。または, TP1/Client/W の場合, すでに常設コネクションが切断されています。
DCCLTER_INVALID_ARGS	-2501	引数が誤っています。
DCCLTER_PROTO	-2502	dc_rpc_open_s 関数が発行されていません。
DCCLTER_NO_BUFS	-2504	必要なバッファが確保できませんでした。
DCCLTER_NET_DOWN	-2506	通信障害が発生しました。または, TP1/Client/P の場合, すでに常設コネクションが切断されています。
DCCLTER_TIMED_OUT	-2507	常設コネクション解放時に時間切れ (タイムアウト) が発生しました。
DCCLTER_SYSERR	-2518	システムエラーが発生しました。
DCCLTER_INVALID_CLTID	-2544	cltid に指定したクライアント ID は, dc_clt_cltin_s 関数で受け取ったクライアント ID と異なります。

(5) 注意事項

- dc_clt_disconnect_s 関数が次のリターン値でエラーリターンした場合, 常設コネクションは解放されません。
 - DCCLTER_INVALID_ARGS
 - DCCLTER_PROTO
 - DCCLTER_NO_BUFS (クライアント側で検知した場合)
 - DCCLTER_INVALID_CLTID
- dc_clt_disconnect_s 関数が次のリターン値でエラーリターンした場合, TP1/Client 側で強制的に常設コネクションを解放します。

- DCCLTER_NO_BUFS (サーバ側で検知した場合)
- DCCLTER_NET_DOWN
- DCCLTER_TIMED_OUT
- DCCLTER_SYSERR

このとき、CUP 実行プロセスまたは DCCM3 の論理端末は、TP1/Client で常設コネクションが解放されたことを検知しないで、CUP からの要求を待ち続けることがあります。要求待ちを避けるため、常設コネクション問い合わせ間隔最大時間 (DCCM3 論理端末の場合は端末放置監視時間) に適切な値を設定してください。

- dc_clt_disconnect_s 関数をトランザクション内で発行した場合、トランザクションはコミットされません。

4.4.3 dc_clt_set_raphost_s – 常設コネクション確立要求先の指定

(1) 形式

(a) TP1/Client/W の場合

■ _s 付き関数

```
#include <dcvclt.h>
DCLONG dc_clt_set_raphost_s(DCCLT_ID cltid,
                           char *raphost,
                           DCLONG flags)
```

■ _s 無し関数

```
#include <dcvclt.h>
DCLONG dc_clt_set_raphost(char *raphost, DCLONG flags)
```

(b) TP1/Client/P の場合

■ _s 付き関数

```
#include <dcvclt.h>
DCLONG dc_clt_set_raphost_s(DCCLT_ID cltid,
                             char CLTFAR *raphost,
                             DCLONG flags)
```

■ _s 無し関数

```
#include <dcvclt.h>
DCLONG dc_clt_set_raphost(char CLTFAR *raphost, DCLONG flags)
```

(2) 機能

常設コネクション確立要求先のホスト名およびポート番号を指定します。この関数を使用した場合、クライアント環境定義 DCCLTRAPHOST に定義したホスト名およびポート番号は無視され、以降の dc_clt_connect_s 関数では、この関数で指定したホスト名およびポート番号が使用されます。

常設コネクション確立要求先のホスト名およびポート番号をこの関数の実行前に戻すときは、dc_clt_get_raphost_s 関数で返された元の値を、この関数で再設定してください。

(3) UAP で値を設定する引数

- cltid

dc_clt_cltin_s 関数で受け取ったクライアント ID を指定します。

- raphost

常設コネクション確立要求先のホスト名およびポート番号を指定します。

形式

ホスト名:ポート番号 [,ホスト名:ポート番号,...]

- ホスト名 ~ 〈文字列〉
- ポート番号 ~ 〈符号なし整数〉 ((5001~65535))

ホスト名として指定できる長さは、63 文字*までです。ホスト名を複数指定する場合、引数 raphost に指定できる長さ（ポート番号なども含む）は、255 文字*までです。

区切り文字','の後ろ以外は空白文字（スペースまたはタブ）を入れないでください。

注※

クライアント環境定義 DCCLTOPTION に 00000008 を指定した場合、ホスト名として指定できる長さは 255 文字、ホスト名を複数指定する場合に引数 raphost に指定できる長さ（ポート番号なども含む）は 1023 文字までとなります。

- flags

DCNOFLAGS を指定します。

(4) リターン値

リターン値	数値 (10 進数)	意味
DC_OK	0	正常終了しました。
DCCLTER_INVALID_ARGS	-2501	引数に指定した値が誤っています。
DCCLTER_PROTO	-2502	トランザクション内で発行されているか、常設コネクション確立中です。または、dc_rpc_open_s 関数が発行されていません。
DCCLTER_NO_BUFS	-2504	必要なバッファが確保できませんでした。

リターン値	数値 (10進数)	意味
DCCLTER_INVALID_CLTID	-2544	cltid に指定したクライアント ID は、dc_clt_cltin_s 関数で受け取ったクライアント ID と異なっています。

(5) 注意事項

- この関数は、クライアント環境定義 DCCLTRAPHOST に指定した値を変更しません。
- raphost に NULL 文字へのポインタを指定した場合、DCCLTRAPHOST が未定義の状態になります。この場合、以降の dc_clt_connect_s 関数では、CUP 実行プロセスまたは DCCM3 の論理端末に常設コネクションを確立します。

4.4.4 dc_clt_get_raphost_s – 常設コネクション確立要求先の取得

(1) 形式

(a) TP1/Client/W の場合

■ _s 付き関数

```
#include <dcvclt.h>
DCLONG dc_clt_get_raphost_s(DCCLT_ID cltid,
                           char *raphost,
                           DCLONG flags)
```

■ _s 無し関数

```
#include <dcvclt.h>
DCLONG dc_clt_get_raphost(char *raphost, DCLONG flags)
```

(b) TP1/Client/P の場合

■ _s 付き関数

```
#include <dcvclt.h>
DCLONG dc_clt_get_raphost_s(DCCLT_ID cltid,
                           char CLTFAR *raphost,
                           DCLONG flags)
```

■ _s 無し関数

```
#include <dcvclt.h>
DCLONG dc_clt_get_raphost(char CLTFAR *raphost, DCLONG flags)
```

(2) 機能

常設コネクション確立要求先のホスト名およびポート番号を取得します。この関数は、`dc_clt_set_raphost_s` 関数で常設コネクション確立要求先を変更する前に、元の値を退避するために使用します。

この関数は、`dc_clt_set_raphost_s` 関数で変更した常設コネクション確立要求先を `raphost` に返します。変更していない場合は、クライアント環境定義 `DCCLTRAPHOST` の値を `raphost` に返します。

(3) UAP で値を設定する引数

- `cltid`

`dc_clt_cltin_s` 関数で受け取ったクライアント ID を指定します。

- `raphost`

現在設定されている常設コネクション確立要求先のホスト名およびポート番号を格納する 256 バイト
*以上の領域へのポインタを指定します。

注※

クライアント環境定義 `DCCLTOPTION` に `00000008` を指定した場合、256 バイトではなく、
1024 バイトになります。

- `flags`

`DCNOFLAGS` を指定します。

(4) 値が返される引数

- `raphost`

現在設定されている常設コネクション確立要求先のホスト名およびポート番号が返されます。クライアント環境定義 `DCCLTRAPHOST` を定義しないで、かつ `dc_clt_set_raphost_s` 関数で常設コネクション確立要求先を指定していない場合は、`raphost` の先頭に `NULL` 文字が返されます。

形式

ホスト名 [:ポート番号] [,ホスト名 [:ポート番号] ,...]

・ホスト名 ~ 〈文字列〉

・ポート番号 ~ 〈符号なし整数〉((5001~65535))

(5) リターン値

リターン値	数値 (10進数)	意味
<code>DC_OK</code>	0	正常終了しました。
<code>DCCLTER_INVALID_ARGS</code>	-2501	引数に指定した値が誤っています。
<code>DCCLTER_PROTO</code>	-2502	<code>dc_rpc_open_s</code> 関数が発行されていません。

リターン値	数値 (10進数)	意味
DCCLTER_NO_BUFS	-2504	必要なバッファが確保できませんでした。
DCCLTER_INVALID_CLTID	-2544	cltid に指定したクライアント ID は、dc_clt_cltin_s 関数で受け取ったクライアント ID と異なっています。

(6) 注意事項

引数 raphost には、256 バイト*以上の領域を指定してください。領域が 256 バイト*未満の場合、TP1/Client 内部の処理で領域破壊を起こすおそれがあります。

注※

クライアント環境定義 DCCLTOPTION に 00000008 を指定した場合、256 バイトではなく、1024 バイトになります。

4.4.5 dc_clt_set_connect_inf_s – 端末識別情報の設定

(1) 形式

(a) TP1/Client/W の場合

■ _s 付き関数

```
#include <dcvclt.h>
DCLONG dc_clt_set_connect_inf_s(DCCLT_ID cltid,
                                char *inf,
                                unsigned short inf_len,
                                DCLONG flags)
```

■ _s 無し関数

```
#include <dcvclt.h>
DCLONG dc_clt_set_connect_inf(char *inf,
                               unsigned short inf_len,
                               DCLONG flags)
```

(b) TP1/Client/P の場合

■ _s 付き関数

```
#include <dcvclt.h>
DCLONG dc_clt_set_connect_inf_s(DCCLT_ID cltid,
                                char CLTFAR *inf,
                                unsigned short inf_len,
                                DCLONG flags)
```

■ _s 無し関数

```
#include <dcvclt.h>
DCLONG dc_clt_set_connect_inf(char CLTFAR *inf,
                               unsigned short inf_len,
                               DCLONG flags)
```

(2) 機能

端末識別情報を動的に設定します。

常設コネクションを使用して DCCM3 論理端末と通信する場合、端末識別情報を DCCM3 論理端末に通知することで、DCCM3 の端末固定割り当て機能を利用できます。

この関数に指定した端末識別情報は、クライアント環境定義 DCCLTRAPHOST に DCCM3 論理端末のホスト名およびポート番号を指定して、dc_clt_connect_s 関数の引数 flags に DCNOFLAGS を指定した場合だけ、有効です。

この関数を実行した場合、クライアント環境定義 DCCLTCONNECTINF に指定した端末識別情報は、dc_rpc_open_s 関数を再び実行するまで参照されません。

この関数に指定した端末識別情報は、この関数のあとに実行する dc_clt_connect_s 関数で参照され、DCCM3 論理端末に通知されます。

この関数を複数回実行した場合は、dc_clt_connect_s 関数実行直前に指定した端末識別情報が、有効となります。

(3) UAP で値を設定する引数

- cltid

dc_clt_cltin_s 関数で受け取ったクライアント ID を指定します。

- inf

端末識別情報を指定します。16 進数で指定する場合は、64 バイト以内で指定します。文字列で指定する場合は、64 文字以内 (NULL 文字を含めない) で指定します。

常設コネクションを使用して DCCM3 論理端末と通信する場合は、端末識別情報として DCCM3 論理端末の論理端末名称を EBCDIK コードで指定します。ただし、DCCM3 側では先頭 8 バイト目までに指定した値だけが有効になります (9 バイト目以降に指定された値は無視されます)。

- inf_len

端末識別情報長を指定します。1 から DCCLT_MAX_CONNECT_INF_SIZE までの範囲の長さが指定できます。DCCLT_MAX_CONNECT_INF_SIZE はヘッダファイル中で規定しています。ヘッダファイルは、TP1/Client/W の場合は dcvclt.h です。TP1/Client/P の場合は DCVCLT.H です。

- flags

DCNOFLAGS を指定します。

(4) リターン値

リターン値	数値 (10進数)	意味
DC_OK	0	正常終了しました。
DCCLTER_INVALID_ARGS	-2501	引数に指定した値が誤っています。
DCCLTER_PROTO	-2502	dc_rpc_open_s 関数が実行されていません。
DCCLTER_NO_BUFS	-2504	必要なバッファが確保できませんでした。
DCCLTER_INVALID_CLTID	-2544	cltid に指定したクライアント ID は、dc_clt_cltin_s 関数で受け取ったクライアント ID と異なっています。

(5) 注意事項

- 端末識別情報の通知によって、DCCM3 の端末固定割り当て機能を利用できるのは、DCCM3 のバージョン 09-03 以降です。端末固定割り当て機能については、マニュアル「VOS3 データマネジメントシステム XDM E2 系 解説」を参照してください。
- dc_clt_set_connect_inf_s 関数で定義した端末識別情報と一致する DCCM3 論理端末の論理端末名称が DCCM3 側で定義されていなかった場合、dc_clt_connect_s 関数は DCCLTER_NET_DOWN でエラーリターンします。

4.5 トランザクション制御

4.5.1 dc_trn_begin_s – トランザクションの開始

(1) 形式

(a) `_s` 付き関数の場合

```
#include <dcvtrn.h>
DCLONG dc_trn_begin_s(DCCLT_ID cltid)
```

(b) `_s` 無し関数の場合

```
#include <dcvtrn.h>
DCLONG dc_trn_begin()
```

(2) 機能

グローバルトランザクションを、`dc_trn_begin_s` 関数を実行する CUP のプロセスから開始します。

`dc_trn_begin_s` 関数は、`dc_rpc_open_s` 関数のあとに実行してください。

`dc_trn_begin_s` 関数を実行してから、トランザクションの同期点（コミットの要求）までが一つのグローバルトランザクションとなります。また、グローバルトランザクションの中では、`dc_trn_begin_s` 関数は重複して実行できません（SPP での `dc_trn_begin_s` 関数を含みます）。実行した場合はエラーリターンします。

SPP のトランザクション属性は、ユーザーサービス定義の `atomic_update` の指定に従います。

(3) UAP で値を設定する引数

- `cltid`

`dc_clt_cltin_s` 関数で受け取ったクライアント ID を指定します。

(4) リターン値

リターン値	数値 (10 進数)	意味
DC_OK	0	正常終了しました。
DCCLTER_PROTO	-2502	誤ったコンテキスト（例 すでにトランザクション内にいる）から関数を呼び出しています。

リターン値	数値 (10進数)	意味
DCCLTER_PROTO	-2502	または、次の条件が重なった環境下から関数を呼び出しています。 <ul style="list-style-type: none"> クライアント環境定義 DCUTOKEY を指定しています。 rap サーバとの常設コネクションが確立中です。
DCCLTER_NO_BUFS	-2504	メモリ不足が発生しました。または、リソース不足が発生しました。
DCCLTER_NET_DOWN	-2506	ネットワーク障害が発生しました。
DCCLTER_TIMED_OUT	-2507	dc_trm_begin_s 関数の処理時間で時間切れ（タイムアウト）が発生しました。
DCCLTER_NO_SUCH_SERVICE_GROUP	-2510	クライアント拡張サービスが起動されていません。システムサービス構成定義 clt_conf の指定が正しいか見直してください。または、トランザクショナル RPC 実行プロセスが起動されていません。クライアントサービス定義 clt_tm_conf の指定が正しいか見直してください。
DCCLTER_OLTF_NOT_UP	-2515	OpenTP1 が起動されていません。
DCCLTER_NO_BUFS_AT_SERVER	-2517	トランザクションプロセス内でメモリ不足が発生しました。
DCCLTER_SYSERR	-2518	システムエラーが発生しました。
DCCLTER_CONNFREE	-2542	常設コネクションが解放されました。
DCCLTER_INVALID_CLTID	-2544	cltid に指定したクライアント ID は、dc_clt_cltin_s 関数で受け取ったクライアント ID と異なっています。
DCCLTER_BUSY_AT_SERVER	-2545	サーバのトランザクション処理に負荷が掛かり過ぎているため、トランザクションを開始できません。 このリターン値が戻った場合は、再び実行すれば成功する可能性が高いので、再実行してください。
DCCLTER_PORT_IN_USE	-2547	指定したポート番号は使用されています。または、OS が自動的に割り当てるポート番号が不足しています。
DCTRNER_RM	-3406	リソースマネージャ (RM) でエラーが発生しました。 トランザクションは発生できませんでした。
DCTRNER_TM	-3407	トランザクションサービスでエラーが発生したので、トランザクションを開始できませんでした。 このリターン値が戻った場合は、再び実行すれば成功する可能性が高いので、再実行してください。

4.5.2 dc_trn_chained_commit_s – 連鎖モードのコミット

(1) 形式

(a) _s 付き関数の場合

```
#include <dcvtrn.h>
DCLONG dc_trn_chained_commit_s(DCCLT_ID cltid)
```

(b) _s 無し関数の場合

```
#include <dcvtrn.h>
DCLONG dc_trn_chained_commit()
```

(2) 機能

トランザクションの同期点を取得します。

dc_trn_chained_commit_s 関数が正常終了すると新しいグローバルトランザクションが発生し、以降実行する関数は新しいグローバルトランザクションの範囲になります。

(3) UAP で値を設定する引数

- cltid

dc_clt_cltin_s 関数で受け取ったクライアント ID を指定します。

(4) リターン値

リターン値	数値 (10 進数)	意味
DC_OK	0	正常終了しました。
DCCLTER_PROTO	-2502	誤ったコンテキスト (例 トランザクションの範囲外) から関数を呼び出しています。
DCCLTER_NO_BUFS	-2504	メモリ不足が発生しました。
DCCLTER_NET_DOWN	-2506	ネットワーク障害が発生しました。
DCCLTER_TIMED_OUT	-2507	dc_trn_chained_commit_s 関数の処理時間で時間切れ (タイムアウト) が発生しました。
DCCLTER_OLTF_NOT_UP	-2515	OpenTP1 が起動されていません。または、サーバとのコネクションが切断されているため、通信できません。
DCCLTER_NO_BUFS_AT_SERVER	-2517	トランザクションプロセス内でメモリ不足が発生しました。
DCCLTER_SYSERR	-2518	システムエラーが発生しました。

リターン値	数値 (10進数)	意味
DCCLTER_CONNFREE	-2542	常設コネクションが解放されました。
DCCLTER_INVALID_CLTID	-2544	cltid に指定したクライアント ID は、dc_clt_cltin_s 関数で受け取ったクライアント ID と異なっています。
DCTRNER_ROLLBACK	-3402	現在のトランザクションは、コミットに失敗したためロールバックしました。完了後、このプロセスはトランザクション下であって、グローバルトランザクションの範囲内です。
DCTRNER_HEURISTIC	-3403	ヒューリスティック決定のため、あるトランザクションブランチはコミットとなり、あるトランザクションはロールバックとなりました。このリターン値は、ヒューリスティック決定の結果が、グローバルトランザクションの同期点の結果と一致しなかった場合に返ります。このリターン値が戻る原因およびグローバルトランザクションの同期点の結果は、メッセージログファイルを参照してください。 このリターン値が返ったあとも、このプロセスはトランザクション下であって、グローバルトランザクションの範囲内です。
DCTRNER_HAZARD	-3404	グローバルトランザクションのトランザクションブランチがヒューリスティックに完了しました。しかし、障害のためヒューリスティックに完了したトランザクションブランチの同期点の結果が判明しません。このリターン値が返る原因およびグローバルトランザクションの同期点の結果は、メッセージログファイルを参照してください。このリターン値が返ったあともこのプロセスはトランザクション下であって、グローバルトランザクションの範囲内です。
DCTRNER_NO_BEGIN	-3424	コミットは正常終了しましたが、新しいトランザクションは開始できませんでした。このリターン値が返ったあとは、このプロセスはトランザクション下にはありません。
DCTRNER_ROLLBACK_NO_BEGIN	-3425	現在のトランザクションは、コミットできないでロールバックしました。その後、新しいトランザクションを開始できなかったため、このプロセスはトランザクション下にありません。
DCTRNER_HEURISTIC_NO_BEGIN	-3426	dc_trn_chained_commit_s 関数を実行したグローバルトランザクションは、ヒューリスティック決定のため、トランザクションブランチにはコミットされたものと、されていないものがあります。このリターン値は、ヒューリスティック決定の結果が、グローバルトランザクションの同期点の結果と一致しなかった場合にリターン値が返る原因となった UAP、リソースマネージャ、およびグローバルトランザクションの同期点の結果は、メッセージログファイルを参照してください。その後、新しいトランザクションを開始できなかったため、このプロセスはトランザクション下にありません。
DCTRNER_HAZARD_NO_BEGIN	-3427	グローバルトランザクションのトランザクションブランチが、ヒューリスティックに完了しました。しかし、障害のため

リターン値	数値 (10進数)	意味
DCTRNER_HAZARD_NO_BEGIN	-3427	め、ヒューリスティックに完了したトランザクションブランチの同期点の結果が判明しません。このリターン値が返る原因となった UAP、リソースマネージャ、およびグローバルトランザクションの同期点の結果は、メッセージログファイルを参照してください。その後、新しいトランザクションを開始できなかったため、このプロセスはトランザクション下ではありません。

(5) 注意事項

トランザクションをコミットしてから CUP のプロセスを終了させるときは、`dc_trn_unchained_commit_s` 関数を必ず実行してください。

4.5.3 `dc_trn_chained_rollback_s` – 連鎖モードのロールバック

(1) 形式

(a) `_s` 付き関数の場合

```
#include <dcvtrn.h>
DCLONG dc_trn_chained_rollback_s(DCCLT_ID cltid)
```

(b) `_s` 無し関数の場合

```
#include <dcvtrn.h>
DCLONG dc_trn_chained_rollback()
```

(2) 機能

トランザクションをロールバックします。

`dc_trn_chained_rollback_s` 関数が正常終了すると、新しいグローバルトランザクションが発生し、以降実行する関数は新しいグローバルトランザクションの範囲になります。

(3) UAP で値を設定する引数

- `cltid`
`dc_clt_cltin_s` 関数で受け取ったクライアント ID を指定します。

(4) リターン値

リターン値	数値 (10進数)	意味
DC_OK	0	正常終了しました。
DCCLTER_PROTO	-2502	誤ったコンテキスト (例 トランザクションの範囲外) から関数を呼び出しています。
DCCLTER_NO_BUFS	-2504	メモリ不足が発生しました。
DCCLTER_NET_DOWN	-2506	ネットワーク障害が発生しました。
DCCLTER_TIMED_OUT	-2507	dc_trn_chained_rollback_s 関数の処理時間で時間切れ (タイムアウト) が発生しました。
DCCLTER_OLTF_NOT_UP	-2515	OpenTP1 が起動されていません。または、サーバとのコネクションが切断されているため、通信できません。
DCCLTER_NO_BUFS_AT_SERVER	-2517	トランザクションプロセス内でメモリ不足が発生しました。
DCCLTER_SYSERR	-2518	システムエラーが発生しました。
DCCLTER_CONNFREE	-2542	常設コネクションが解放されました。
DCCLTER_INVALID_CLTID	-2544	cltid に指定したクライアント ID は、dc_clt_cltin_s 関数で受け取ったクライアント ID と異なっています。
DCTRNER_HEURISTIC	-3403	<p>ヒューリスティック決定のため、あるトランザクションブランチはコミットとなり、あるトランザクションはロールバックとなりました。</p> <p>このリターン値は、ヒューリスティック決定の結果が、グローバルトランザクションの同期点の結果と一致しなかった場合に返ります。このリターン値が戻る原因およびグローバルトランザクションの同期点の結果は、メッセージログファイルを参照してください。</p> <p>このリターン値が戻ったあとも、このプロセスはトランザクション下であって、グローバルトランザクションの範囲内です。</p>
DCTRNER_HAZARD	-3404	<p>グローバルトランザクションのトランザクションブランチがヒューリスティックに完了しました。しかし、障害のためヒューリスティックに完了したトランザクションブランチの同期点の結果が判明しません。このリターン値が戻る原因およびグローバルトランザクションの同期点の結果は、メッセージログファイルを参照してください。</p> <p>このリターン値が戻ったあともこのプロセスはトランザクション下であって、グローバルトランザクションの範囲内です。</p>
DCTRNER_NO_BEGIN	-3424	ロールバックは正常終了しましたが、新しいトランザクションは開始できませんでした。このリターン値が返ったあとは、このプロセスはトランザクション下にはありません。

リターン値	数値 (10進数)	意味
DCTRNER_HEURISTIC_NO_BEGIN	-3426	<p>dc_trn_chained_rollback_s 関数を実行したグローバルトランザクションは、ヒューリスティック決定のため、トランザクションブランチにはコミットされたものと、されていないものがあります。</p> <p>このリターン値は、ヒューリスティック決定の結果が、グローバルトランザクションの同期点の結果と一致しなかった場合に返ります。</p> <p>このリターン値が返る原因となった UAP、リソースマネージャ、およびグローバルトランザクションの同期点の結果は、メッセージログファイルを参照してください。</p> <p>その後、新しいトランザクションを開始できなかったため、このプロセスはトランザクション下にありません。</p>
DCTRNER_HAZARD_NO_BEGIN	-3427	<p>グローバルトランザクションのトランザクションブランチが、ヒューリスティックに完了しました。しかし、障害のため、ヒューリスティックに完了したトランザクションブランチの同期点の結果が判明しません。</p> <p>このリターン値が返る原因となった UAP、リソースマネージャ、およびグローバルトランザクションの同期点の結果は、メッセージログファイルを参照してください。</p> <p>その後、新しいトランザクションを開始できなかったため、このプロセスはトランザクション下にありません。</p>

(5) 注意事項

トランザクションをロールバックしてから CUP のプロセスを終了させるときは、dc_trn_unchained_rollback_s 関数を必ず実行してください。

4.5.4 dc_trn_unchained_commit_s – 非連鎖モードのコミット

(1) 形式

(a) _s 付き関数の場合

```
#include <dcvtrn.h>
DCLONG dc_trn_unchained_commit_s(DCCLT_ID cltid)
```

(b) _s 無し関数の場合

```
#include <dcvtrn.h>
DCLONG dc_trn_unchained_commit()
```

(2) 機能

トランザクションの同期点を取得します。

dc_trn_unchained_commit_s 関数が正常に終了すると、グローバルトランザクションは終了します。グローバルトランザクションの範囲外からは、SPP をトランザクションとして実行できません。

(3) UAP で値を設定する引数

- cltid

dc_clt_cltin_s 関数で受け取ったクライアント ID を指定します。

(4) リターン値

リターン値	数値 (10 進数)	意味
DC_OK	0	正常終了しました。
DCCLTER_PROTO	-2502	誤ったコンテキスト (例 トランザクションの範囲外) から関数を実行しています。
DCCLTER_NO_BUFS	-2504	メモリ不足が発生しました。
DCCLTER_NET_DOWN	-2506	ネットワーク障害が発生しました。
DCCLTER_TIMED_OUT	-2507	dc_trn_unchained_commit_s 関数の処理時間で時間切れ (タイムアウト) が発生しました。
DCCLTER_OLTF_NOT_UP	-2515	OpenTP1 が起動されていません。または、サーバとの接続が切断されているため、通信できません。
DCCLTER_NO_BUFS_AT_SERVER	-2517	トランザクションプロセス内でメモリ不足が発生しました。
DCCLTER_SYSERR	-2518	システムエラーが発生しました。
DCCLTER_CONNFREE	-2542	常設接続が解放されました。
DCCLTER_INVALID_CLTID	-2544	cltid に指定したクライアント ID は、dc_clt_cltin_s 関数で受け取ったクライアント ID と異なっています。
DCTRNER_ROLLBACK	-3402	コミットに失敗したため、トランザクションをロールバックしました。このリターン値が戻ったあと、このプロセスはグローバルトランザクションの範囲外となります。
DCTRNER_HEURISTIC	-3403	ヒューリスティック決定によって、一部、またはすべてのトランザクションブランチがロールバックされました。詳細は、メッセージログファイルを参照してください。このリターン値が戻ったあと、このプロセスはグローバルトランザクションの範囲外となります。
DCTRNER_HAZARD	-3404	ヒューリスティック決定でトランザクションが完了しましたが、障害のため結果がわかりません。詳細は、メッセージログファイルを参照してください。このリターン値が戻ったあ

リターン値	数値 (10進数)	意味
DCTRNER_HAZARD	-3404	と、このプロセスはグローバルトランザクションの範囲外となります。

(5) 注意事項

CUP のプロセスを正常に終了させるときは、dc_trn_unchained_commit_s 関数を実行してトランザクションをコミットしてください。

4.5.5 dc_trn_unchained_rollback_s – 非連鎖モードのロールバック

(1) 形式

(a) _s 付き関数の場合

```
#include <dcvtrn.h>
DCLONG dc_trn_unchained_rollback_s(DCCLT_ID cltid)
```

(b) _s 無し関数の場合

```
#include <dcvtrn.h>
DCLONG dc_trn_unchained_rollback()
```

(2) 機能

トランザクションをロールバックします。

dc_trn_unchained_rollback_s 関数が正常終了すると、グローバルトランザクションは終了します。グローバルトランザクションの範囲外からは、SPP をトランザクションとして実行できません。

(3) UAP で値を設定する引数

- cltid

dc_clt_cltin_s 関数で受け取ったクライアント ID を指定します。

(4) リターン値

リターン値	数値 (10進数)	意味
DC_OK	0	正常終了しました。

リターン値	数値 (10進数)	意味
DCCLTER_PROTO	-2502	誤ったコンテキスト（例 トランザクションの範囲外）から関数を実行しています。
DCCLTER_NO_BUFS	-2504	メモリ不足が発生しました。
DCCLTER_NET_DOWN	-2506	ネットワーク障害が発生しました。
DCCLTER_TIMED_OUT	-2507	dc_trn_unchained_rollback_s 関数の処理時間で時間切れ（タイムアウト）が発生しました。
DCCLTER_OLTF_NOT_UP	-2515	OpenTP1 が起動されていません。または、サーバとの接続が切断されているため、通信できません。
DCCLTER_NO_BUFS_AT_SERVER	-2517	トランザクションプロセス内でメモリ不足が発生しました。
DCCLTER_SYSERR	-2518	システムエラーが発生しました。
DCCLTER_CONNFREE	-2542	常設接続が解放されました。
DCCLTER_INVALID_CLTID	-2544	cltid に指定したクライアント ID は、dc_clt_cltin_s 関数で受け取ったクライアント ID と異なっています。
DCTRNER_HEURISTIC	-3403	ヒューリスティック決定によって、一部、またはすべてのトランザクションブランチがロールバックされました。詳細は、メッセージログファイルを参照してください。このリターン値が戻ったあと、このプロセスはグローバルトランザクションの範囲外となります。
DCTRNER_HAZARD	-3404	ヒューリスティック決定でトランザクションが完了しましたが、障害のため結果がわかりません。詳細は、メッセージログファイルを参照してください。このリターン値が戻ったあと、このプロセスはグローバルトランザクションの範囲外となります。

(5) 注意事項

トランザクションをロールバックしてから CUP のプロセスを終了させるときは、必ず dc_trn_unchained_rollback_s 関数を実行してください。

4.5.6 dc_clt_get_trnid_s – 現在のトランザクションに関する識別子の取得

(1) 形式

(a) TP1/Client/W の場合

■ _s 付き関数

```
#include <dcvclt.h>
DCLONG dc_clt_get_trnid_s(DCCLT_ID cltid, char *trngid,
                          char *trnbid)
```

■ _s 無し関数

```
#include <dcvclt.h>
DCLONG dc_clt_get_trnid(char *trngid, char *trnbid)
```

(b) TP1/Client/P の場合

■ _s 付き関数

```
#include <dcvclt.h>
DCLONG dc_clt_get_trnid_s(DCCLT_ID cltid, char CLTFAR *trngid,
                          char CLTFAR *trnbid)
```

■ _s 無し関数

```
#include <dcvclt.h>
DCLONG dc_clt_get_trnid(char CLTFAR *trngid, char CLTFAR *trnbid)
```

(2) 機能

現在のトランザクショングローバル識別子、およびトランザクションブランチ識別子を取得します。

この識別子は、次に示す関数を実行してトランザクションが起動されたときに、OpenTP1 が割り当てたものです。

- dc_trn_begin_s 関数
- dc_trn_chained_commit_s 関数
- dc_trn_chained_rollback_s 関数

(3) UAP で値を設定する引数

- cltid
dc_clt_cltin_s 関数で受け取ったクライアント ID を指定します。

- trngid

トランザクショングローバル識別子を受け取る領域を指定します。

トランザクショングローバル識別子は 16 文字です。最後に NULL 文字が付けられるため、17 バイト以上の領域を確保してください。

- trnbid

トランザクションブランチ識別子を受け取る領域を指定します。

トランザクションブランチ識別子は 16 文字です。最後に NULL 文字が付けられるため、17 バイト以上の領域を確保してください。

(4) 値が返される引数

- trngid

トランザクショングローバル識別子が返されます。

- trnbid

トランザクションブランチ識別子が返されます。

(5) リターン値

リターン値	数値 (10 進数)	意味
DC_OK	0	正常終了しました。
DCCLTER_INVALID_ARGS	-2501	trngid, または trnbid のポインタが NULL です。
DCCLTER_PROTO	-2502	誤ったコンテキスト (例 トランザクションの範囲外) から関数を呼び出しています。
DCCLTER_NO_BUFS	-2504	メモリ不足が発生しました。
DCCLTER_INVALID_CLTID	-2544	cltid に指定したクライアント ID は、dc_clt_cltin_s 関数で受け取ったクライアント ID と異なります。

(6) 注意事項

引数 trngid および trnbid には、17 バイト以上の領域を指定してください。領域が 17 バイト未満の場合、TP1/Client 内部の処理で領域破壊を起こすおそれがあります。

4.5.7 dc_trn_info_s – 現在のトランザクションに関する情報の報告

(1) 形式

(a) TP1/Client/W の場合

■ _s 付き関数

```
#include <dcvtrn.h>
DCLONG dc_trn_info_s(DCCLT_ID cltid, char *flags)
```

■ _s 無し関数

```
#include <dcvtrn.h>
DCLONG dc_trn_info(char *flags)
```

(b) TP1/Client/P の場合

■ _s 付き関数

```
#include <dcvtrn.h>
DCLONG dc_trn_info_s(DCCLT_ID cltid, char CLTFAR *flags)
```

■ _s 無し関数

```
#include <dcvtrn.h>
DCLONG dc_trn_info(char CLTFAR *flags)
```

(2) 機能

dc_trn_info_s 関数を実行した CUP が、現在トランザクションとして稼働しているかどうかを報告します。

(3) UAP で値を設定する引数

- cltid
dc_clt_cltin_s 関数で受け取ったクライアント ID を指定します。
- flags
NULL を指定します。

(4) リターン値

リターン値	数値 (10進数)	意味
—	1	dc_tm_info_s 関数を実行した CUP のプロセスは、トランザクションの範囲内にあります。
—	0	dc_tm_info_s 関数を実行した CUP のプロセスは、トランザクションの範囲外にあります。
DCCLTER_INVALID_CLTID	-2544	cltid に指定したクライアント ID は、dc_clt_cltin_s 関数で受け取ったクライアント ID と異なっています。

(凡例)

—：該当しません。

4.6 TCP/IP 通信機能

4.6.1 dc_clt_send_s – メッセージの送信

(1) 形式

(a) TP1/Client/W の場合

■ _s 付き関数

```
#include <dcvclt.h>
DCLONG dc_clt_send_s(DCCLT_ID cltid, char *buff,
                    DCLONG sendleng, char *hostname,
                    unsigned short portnum, DCLONG flags)
```

■ _s 無し関数

```
#include <dcvclt.h>
int dc_clt_send(char *buff, DCLONG sendleng, char *hostname,
               unsigned short portnum, DCLONG flags)
```

(b) TP1/Client/P の場合

■ _s 付き関数

```
#include <dcvclt.h>
DCLONG dc_clt_send_s(DCCLT_ID cltid, char CLTFAR *buff,
                    DCLONG sendleng, char CLTFAR *hostname,
                    unsigned short portnum, DCLONG flags)
```

■ _s 無し関数

```
#include <dcvclt.h>
int dc_clt_send(char CLTFAR *buff, DCLONG sendleng,
               char CLTFAR *hostname,
               unsigned short portnum, DCLONG flags)
```

(2) 機能

MHP へメッセージを送信します。

dc_clt_send_s 関数を実行する場合、flags に DCCLT_ONEWAY_SND、または DCCLT_SNDRCV を指定した dc_rpc_open_s 関数を、あらかじめ実行しておく必要があります。

(3) UAP で値を設定する引数

- **cltid**
dc_clt_cltin_s 関数で受け取ったクライアント ID を指定します。
- **buff**
送信するメッセージが格納されている領域を指定します。sendleng で指定する長さ以上の領域を指定してください。
- **sendleng**
送信するメッセージの長さを指定します。
- **hostname**
コネクションが確立されていない場合、接続するノードのホスト名を指定します。
NULL を指定すると、dc_rpc_open_s 関数を実行したときに取得したクライアント環境定義 DCSNDHOST の内容を参照します。
ホスト名として指定できる長さは、63 文字*までです。
ホスト名として、10 進ドット記法の IP アドレスを指定することもできます。
注*
クライアント環境定義 DCCLTOPTION に 00000008 を指定した場合、ホスト名として指定できる長さは 255 文字までとなります。
- **portnum**
コネクションが確立されていない場合、コネクションを確立して接続するノードのポート番号を指定します。
0 を指定すると、dc_rpc_open_s 関数を実行したときに取得したクライアント環境定義 DCSNDPORT の内容を参照します。
- **flags**
メッセージを送信後に、コネクションを解放するかどうかを指定します。
DCNOFLAGS：メッセージを送信後、コネクションを解放しません。
DCCLT_SND_CLOSE：メッセージを送信後、コネクションを解放します。
DCNOFLAGS を指定した場合、dc_rpc_close_s 関数を実行するまでコネクションを解放しません。
ただし、障害時を除きます。

(4) リターン値

リターン値	数値 (10 進数)	意味
DC_OK	0	正常終了しました。
DCCLTER_INVALID_ARGS	-2501	引数に指定した値が誤っています。
DCCLTER_PROTO	-2502	dc_rpc_open_s 関数が実行されていません。

リターン値	数値 (10進数)	意味
DCCLTER_PROTO	-2502	または、dc_rpc_open_s 関数は実行されていますが、flags に DCCLT_ONEWAY_SND、または DCCLT_SNDRCV を指定していません。
DCCLTER_NO_BUFS	-2504	メモリ不足が発生しました。
DCCLTER_NET_DOWN	-2506	ネットワーク障害が発生しました。
DCCLTER_TIMED_OUT	-2507	コネクション確立要求時にタイムアウトになりました。
DCCLTER_SYSERR	-2518	システムエラーが発生しました。
DCCLTER_RESOURCE	-2538	資源不足が発生しました。
DCCLTER_WRONG_HOST	-2539	ホスト名が誤っています。または、hostname、および DCSNDHOST の両方にホスト名が指定されていません。
DCCLTER_CONNREFUSED	-2541	相手システムに対するコネクションの確立要求が拒絶されました。
DCCLTER_INVALID_CLTID	-2544	cltid に指定したクライアント ID は、dc_clt_cltin_s 関数で受け取ったクライアント ID と異なります。
DCCLTER_PORT_IN_USE	-2547	OS が自動的に割り当てるポート番号が不足しています。

(5) 注意事項

メッセージ送信時に相手システムからコネクションが解放された場合、送信するメッセージ長によっては、コネクションが解放されたことを検知できないことがあります。この場合は、この関数の次以降に発行する関数で検知することがあります。CUP を作成するときは、このことを考慮してください。

4.6.2 dc_clt_receive_s – メッセージの受信

(1) 形式

(a) TP1/Client/W の場合

■ _s 付き関数

```
#include <dcvclt.h>
DCLONG dc_clt_receive_s(DCCLT_ID cltid, char *buff,
                        DCLONG recvleng, DCLONG timeout,
                        DCLONG flags)
```

■ `_s` 無し関数

```
#include <dcvclt.h>
int dc_clt_receive(char *buff, DCLONG recvleng, DCLONG timeout,
                  DCLONG flags)
```

(b) TP1/Client/P の場合

■ `_s` 付き関数

```
#include <dcvclt.h>
DCLONG dc_clt_receive_s(DCCLT_ID cltid, char CLTFAR *buff,
                       DCLONG recvleng, DCLONG timeout,
                       DCLONG flags)
```

■ `_s` 無し関数

```
#include <dcvclt.h>
int dc_clt_receive(char CLTFAR *buff, DCLONG recvleng,
                  DCLONG timeout,
                  DCLONG flags)
```

(2) 機能

MHP が送信したメッセージを受信します。

`dc_clt_receive_s` 関数を実行する場合、`flags` に `DCCLT_ONERWAY_RCV`、または `DCCLT_SNDRCV` を指定した `dc_rpc_open_s` 関数を、あらかじめ実行しておく必要があります。

(3) UAP で値を設定する引数

- `cltid`
`dc_cltin_s` 関数で受け取ったクライアント ID を指定します。
- `buff`
受信したメッセージを格納する領域を指定します。`recvleng` で指定する長さ以上の領域を指定してください。
- `recvleng`
受信するメッセージの長さを指定します。
- `timeout`
メッセージ受信時の最大待ち時間（秒）を指定します。`-1` から `65535` の整数を指定してください。
`-1` を指定した場合は、メッセージを受信するまで無制限に待ちます。
`0` を指定した場合は、メッセージの受信を待ちません。受信するメッセージがなかった場合は、`DCCLTER_TIMED_OUT` でエラーリターンします。

1 から 65535 を指定した場合は、指定した秒数だけメッセージの受信を待ちます。指定した秒数を過ぎてメッセージを受信できない場合は、DCCLTER_TIMED_OUT でエラーリターンします。

- flags

メッセージを受信後に、接続を解放するかどうかを指定します。

DCNOFLAGS：メッセージを受信後、接続を解放しません。

DCCLT_RCV_CLOSE：メッセージを受信後、接続を解放します。

DCNOFLAGS を指定した場合、dc_rpc_close_s 関数を実行するまで接続を解放しません。ただし、障害時を除きます。

(4) 値が返される引数

- buff

受信したメッセージが返されます。

(5) リターン値

リターン値	数値 (10 進数)	意味
DC_OK	0	正常終了しました。
DCCLTER_INVALID_ARGS	-2501	引数に指定した値が誤っています。
DCCLTER_PROTO	-2502	次のどちらかの要因が考えられます。 <ul style="list-style-type: none"> • dc_rpc_open_s 関数が実行されていません。 • dc_rpc_open_s 関数は実行されていますが、flags に DCCLT_ONEWAY_RCV, または DCCLT_SNDRCV を指定していません。
DCCLTER_NO_BUFS	-2504	メモリ不足が発生しました。
DCCLTER_NET_DOWN	-2506	ネットワーク障害が発生しました。
DCCLTER_TIMED_OUT	-2507	メッセージの受信時にタイムアウトになりました。
DCCLTER_SYSERR	-2518	システムエラーが発生しました。
DCCLTER_RESOURCE	-2538	資源不足が発生しました。
DCCLTER_CONNFREE	-2542	相手システムから接続が解放されました。
DCCLTER_INVALID_CLTID	-2544	cltid に指定したクライアント ID は、dc_clt_cltin_s 関数で受け取ったクライアント ID と異なっています。

(6) 注意事項

- dc_clt_receive_s 関数は、次に示す場合だけ、CUP に制御を戻します。
 - MHP から recvleng で指定した長さ分のメッセージを受信した場合
指定した長さより短いメッセージを受信しても CUP に制御は戻しません

- MHP からのメッセージ受信時に、時間切れ（タイムアウト）が発生した場合
- MHP から接続が解放された場合
指定した長さより短いメッセージを受信しても CUP に制御は戻しません
- ネットワーク障害が発生した場合
- dc_clt_receive_s 関数の発行時に、MHP から接続が解放された場合、DCCLTER_CONNFREE でエラーリターンします。

4.6.3 dc_clt_receive2_s – メッセージの受信（障害時メッセージ受信）

(1) 形式

(a) TP1/Client/W の場合

■ _s 付き関数

```
#include <dcvclt.h>
DCLONG dc_clt_receive2_s(DCCLT_ID cltid, char *buff,
                        DCLONG *recvleng, DCLONG timeout,
                        DCLONG flags)
```

■ _s 無し関数

```
#include <dcvclt.h>
DCLONG dc_clt_receive2(char *buff, DCLONG *recvleng,
                       DCLONG timeout, DCLONG flags)
```

(b) TP1/Client/P の場合

■ _s 付き関数

```
#include <dcvclt.h>
DCLONG dc_clt_receive2_s(DCCLT_ID cltid, char CLTFAR *buff,
                        DCLONG CLTFAR *recvleng, DCLONG timeout,
                        DCLONG flags)
```

■ _s 無し関数

```
#include <dcvclt.h>
DCLONG dc_clt_receive2(char CLTFAR *buff,
                       DCLONG CLTFAR *recvleng,
                       DCLONG timeout, DCLONG flags)
```

(2) 機能

MHP が送信したメッセージを受信します。

dc_clt_receive2_s 関数を実行する場合、flags に DCCLT_ONERWAY_RCV、または DCCLT_SNDRCV を指定した dc_rpc_open_s 関数を、あらかじめ実行しておく必要があります。

(3) UAP で値を設定する引数

- **cltid**
dc_clt_cltin_s 関数で受け取ったクライアント ID を指定します。
- **buff**
受信したメッセージを格納する領域を指定します。recvleng で指定する長さ以上の領域を指定してください。
- **recvleng**
受信するメッセージの長さを指定します。
- **timeout**
メッセージ受信時の最大待ち時間（秒）を指定します。-1 から 65535 の整数を指定してください。
-1 を指定した場合は、メッセージを受信するまで無制限に待ちます。
0 を指定した場合は、メッセージの受信を待ちません。受信するメッセージがなかった場合は、DCCLTER_TIMED_OUT でエラーリターンします。
1 から 65535 を指定した場合は、指定した秒数だけメッセージの受信を待ちます。指定した秒数を過ぎてもメッセージを受信できない場合は、DCCLTER_TIMED_OUT でエラーリターンします。
- **flags**
メッセージを受信後に、コネクションを解放するかどうかを指定します。
DCNOFLAGS：メッセージ受信後、コネクションを解放しません。
DCCLT_RCV_CLOSE：メッセージ受信後、コネクションを解放します。
DCNOFLAGS を指定した場合、dc_rpc_close_s 関数を実行するまでコネクションを解放しません。ただし、障害時を除きます。

(4) 値が返される引数

- **buff**
受信したメッセージが返されます。
- **recvleng**
受信したメッセージの長さが返されます。

(5) リターン値

リターン値	数値 (10 進数)	意味
DC_OK	0	正常に終了しました。

リターン値	数値 (10進数)	意味
DCCLTER_INVALID_ARGS	-2501	引数に指定した値が誤っています。
DCCLTER_PROTO	-2502	次のどちらかの要因が考えられます。 <ul style="list-style-type: none"> • dc_rpc_open_s 関数が実行されていません。 • dc_rpc_open_s 関数は実行されていますが、flags に DCCLT_ONEWAY_RCV, または DCCLT_SNDRCV を指定していません。
DCCLTER_NO_BUFS	-2504	メモリ不足が発生しました。
DCCLTER_NET_DOWN	-2506	ネットワーク障害が発生しました。
DCCLTER_TIMED_OUT	-2507	メッセージの受信時にタイムアウトになりました。
DCCLTER_SYSERR	-2518	システムエラーが発生しました。
DCCLTER_RESOURCE	-2538	資源不足が発生しました。
DCCLTER_CONNFREE	-2542	相手システムからコネクションが解放されました。
DCCLTER_INVALID_CLTID	-2544	cltid に指定したクライアント ID は、dc_clt_cltin_s 関数で受け取ったクライアント ID と異なっています。

(6) 注意事項

- dc_clt_receive2_s 関数は、次に示す場合だけ、CUP に制御を戻します。
 - MHP から recvleng で指定した長さ分のメッセージを受信した場合
指定した長さより短いメッセージを受信しても CUP に制御は戻しません
 - MHP からのメッセージ受信時に、時間切れ（タイムアウト）が発生した場合
 - MHP からコネクションが解放された場合
 - ネットワーク障害が発生した場合
- dc_clt_receive2_s 関数の発行時に、MHP からコネクションが解放された場合、DCCLTER_CONNFREE でエラーリターンします。

4.6.4 dc_clt_assem_send_s – 組み立てメッセージの送信

(1) 形式

(a) _s 付き関数

```
#include <dcvclt.h>
DCLONG dc_clt_assem_send_s(DCCLT_ID cltid, char CLTFAR *buff, DCLONG sendleng,
char CLTFAR *hostname, unsigned short portnum, DCLONG timeout, DCLONG flags)
```

(b) `_s` 無し関数

```
#include <dcvclt.h>
DCLONG dc_clt_assem_send(char CLTFAR *buff, DCLONG sendleng, char CLTFAR *hostname,
unsigned short portnum, DCLONG timeout, DCLONG flags)
```

(2) 機能

メッセージの組み立て機能を使用して、メッセージを送信します。この場合、4バイトのメッセージ情報を送信したあと、引数 `buff` に指定されたメッセージを送信します。相手システムとの接続が確立されていない場合は、引数 `hostname` と `portnum` の指定値を基に接続を確立し、メッセージを送信します。

また、クライアント環境定義 `DCCLTDELIVERYCHECK` に `Y` を指定した場合は、メッセージの送達確認機能を使用して、メッセージを送信します。この場合、11バイトのメッセージ情報を送信したあと、引数 `buff` に指定されたメッセージを送信します。TP1/Client は、11バイトのメッセージ情報を受信したあと、CUP に制御を戻します。

この関数を実行する場合、引数 `flags` に `DCCLT_ONEWAY_SND` または `DCCLT_SNDRCV` を指定した、`dc_rpc_open_s` 関数をあらかじめ実行してください。

(3) UAP で値を設定する引数

- `cltid`
`dc_clt_cltin_s` 関数で受け取ったクライアント ID を指定します。
- `buff`
送信するメッセージを格納する領域を指定します。 `sendleng` で指定する長さ以上の領域を指定してください。 .
- `sendleng`
送信するメッセージの長さを指定します。
- `hostname`
接続が確立されていない場合、接続するノードのホスト名を指定します。
NULL を指定すると、`dc_rpc_open_s` 関数を実行したときに取得したクライアント環境定義 `DCSNDHOST` の内容を参照します。
ホスト名として指定できる長さは、63 文字*までです。
ホスト名として、10 進ドット記法の IP アドレスを指定することもできます。
注※
クライアント環境定義 `DCCLTOPTION` に `00000008` を指定した場合、ホスト名として指定できる長さは 255 文字までとなります。
- `portnum`

コネクションが確立されていない場合、コネクションを確立して接続するノードのポート番号を指定します。

0 を指定すると、dc_rpc_open_s 関数を実行したときに取得したクライアント環境定義 DCSNDPORT の内容を参照します。

• timeout

メッセージの送達確認機能を使用するときには有効な引数です。応答専用データ受信時の最大待ち時間(秒)を指定します。-1 から 65535 の整数を指定してください。

-1 を指定した場合

応答専用データを受信するまで無制限に待ちます。

0 を指定した場合

応答専用データの受信を待ちません。受信するメッセージがなかった場合は、DCCLTER_TIMED_OUT でエラーリターンします。

1 から 65535 を指定した場合

指定した秒数だけメッセージの受信を待ちます。指定した秒数を過ぎてもメッセージを受信できない場合は、DCCLTER_TIMED_OUT でエラーリターンします。

応答専用データが分割されて届いた場合は、11 バイトの応答専用データが届くまで受信処理を繰り返します。受信処理が発生すると、この引数に指定した最大待ち時間が毎回適用されます。この引数に指定した値を、クライアントの最大応答待ち時間として適用したい場合は、クライアント環境定義 DCCLTOPTION に、00000002 オプションを指定してください。

• flags

メッセージを送信したあと、コネクションを解放するかどうかを指定します。

DCNOFLAGS

メッセージを送信したあと、dc_rpc_close_s 関数を実行するまでコネクションを解放しません。ただし、障害時を除きます。

DCCLT_SND_CLOSE

メッセージを送信したあと、コネクションを解放します。メッセージの送達確認機能を使用する場合は、メッセージ情報を受信したあと、コネクションを解放します。

(4) リターン値

リターン値	数値 (10 進数)	意味
DC_OK	0	正常終了しました。
DCCLTER_INVALID_ARGS	-2501	引数に指定した値が誤っています。
DCCLTER_PROTO	-2502	次のどちらかの要因が考えられます。 <ul style="list-style-type: none">dc_rpc_open_s 関数が実行されていません。

リターン値	数値 (10進数)	意味
DCCLTER_PROTO	-2502	• dc_rpc_open_s 関数は実行されていますが、引数 flags に DCCLT_ONEWAY_SND, または DCCLT_SNDRCV を指定していません。
DCCLTER_NO_BUFS	-2504	メモリ不足が発生しました。
DCCLTER_NET_DOWN	-2506	ネットワーク障害が発生しました。コネクションは解放されます。
DCCLTER_TIMED_OUT	-2507	コネクション確立要求時にタイムアウトになりました。または、メッセージの送達確認機能の使用時に、応答専用データの受信でタイムアウトになりました。コネクションは解放されます。
DCCLTER_SYSERR	-2518	システムエラーが発生しました。ネットワーク障害の場合、コネクションは解放されます。
DCCLTER_RESOURCE	-2538	資源不足が発生しました。
DCCLTER_WRONG_HOST	-2539	ホスト名が誤っています。または、hostname および DCSNDHOST の両方にホスト名が指定されていません。
DCCLTER_CONNREFUSED	-2541	相手システムに対するコネクションの確立要求が拒絶されました。
DCCLTER_CONNFREE	-2542	メッセージの送達確認機能の使用時に、相手システムからコネクションが解放されました。
DCCLTER_INVALID_CLTID	-2544	引数 cltid に指定したクライアント ID が、dc_clt_cltin_s 関数で受け取ったクライアント ID と異なっています。
DCCLTER_PORT_IN_USE	-2547	OS が自動的に割り当てるポート番号が不足しています。
DCCLTER_INVALID_MESSAGE	-2548	メッセージの送達確認機能の使用時に、不正なメッセージを受信しました。コネクションは解放されます。
DCCLTER_COLLISION_MESSAGE	-2584	メッセージの送達確認機能の使用時に、メッセージが衝突しました。コネクションは解放されます。

(5) 注意事項

- メッセージ送信時に相手システムからコネクションが解放された場合、送信するメッセージ長によっては、コネクションの解放を検知できないことがあります。次に、使用する機能別に注意事項を示します。

メッセージの組み立て機能を使用する場合

メッセージ送信時にこの関数でコネクションの解放を検知できないときは、この関数の次以降に発行する関数で検知することがあります。CUP を作成するときは、このことを考慮してください。

メッセージの送達確認機能を使用する場合

メッセージ送信時にコネクションの解放を検知できないときは、応答専用データ受信時に検知します。

- メッセージの組み立て機能および送達確認機能を使用する場合は、短いパケットで送受信するため、送信処理に時間が掛かることがあります。送信処理に時間が掛かる場合は、クライアント環境定義 DCCLTTCPNODELAY に Y を指定してください。

4.6.5 dc_clt_assem_receive_s – 組み立てメッセージの受信

(1) 形式

(a) _s 付き関数

```
#include <dcvclt.h>
DCLONG dc_clt_assem_receive_s(DCCLT_ID cltid, char CLTFAR *buff, DCLONG CLTFAR *recvleng,
DCLONG timeout, DCLONG flags)
```

(b) _s 無し関数

```
#include <dcvclt.h>
DCLONG dc_clt_assem_receive(char CLTFAR *buff, DCLONG CLTFAR *recvleng,
DCLONG timeout, DCLONG flags)
```

(2) 機能

メッセージの組み立て機能を使用して、メッセージを受信します。この場合、4バイトのメッセージ情報を受信したあと、メッセージ情報に設定されたメッセージ長分のメッセージを受信し、引数 buff に格納します。ただし、メッセージ情報は、引数 buff には格納しません。引数 recvleng には、受信したメッセージ長を格納します。引数 recvleng に格納するメッセージ長には、メッセージ情報の長さは含みません。

また、クライアント環境定義 DCCLTDELIVERYCHECK に Y を指定した場合は、メッセージの送達確認機能を使用して、メッセージを送受信します。この場合、11バイトのメッセージ情報を受信したあと、メッセージ情報に設定されたメッセージ長分のメッセージを受信し、引数 buff に格納します。ただし、メッセージ情報は、引数 buff には格納しません。引数 recvleng には、受信したメッセージ長を格納します。引数 recvleng に格納するメッセージ長には、メッセージ情報の長さは含みません。受信したメッセージ情報に「応答要求」が設定されていた場合は、11バイトのメッセージ情報を送信したあと、CUP に制御を戻します。

この関数を実行する場合、引数 flags に DCCLT_ONERWAY_RCV, または DCCLT_SNDRCV を指定した dc_rpc_open_s 関数を、あらかじめ実行しておく必要があります。

(3) UAP で値を設定する引数

- cltid
dc_clt_cltin_s 関数で受け取ったクライアント ID を指定します。
- buff

受信したメッセージを格納する領域を指定します。通信相手が送信するメッセージ長以上の領域を指定してください。

- **recvleng**

受信したメッセージを格納する領域長（引数 buff の長さ）を指定します。

- **timeout**

メッセージ受信時の最大待ち時間（秒）を指定します。-1 から 65535 の整数を指定してください。

-1 を指定した場合

メッセージを受信するまで無制限に待ちます。

0 を指定した場合

メッセージの受信を待ちません。受信するメッセージがなかった場合は、DCCLTER_TIMED_OUT でエラーリターンします。

1 から 65535 を指定した場合

指定した秒数だけメッセージの受信を待ちます。指定した秒数を過ぎてもメッセージを受信できない場合は、DCCLTER_TIMED_OUT でエラーリターンします。

メッセージが分割されて届いた場合、メッセージ長分のメッセージが届くまで受信処理を繰り返します。受信処理が発生すると、この引数に指定した値が毎回適用されます。この引数に指定した値をクライアントの最大応答待ち時間として適用したい場合は、クライアント環境定義 DCCLTOPTION に、00000002 オプションを指定してください。

- **flags**

メッセージを受信したあと、コネクションを解放するかどうかを指定します。

DCNOFLAGS

メッセージを受信したあと、dc_rpc_close_s 関数を発行するまで、コネクションを解放しません。ただし、障害時を除きます。

DCCLT_RCV_CLOSE

メッセージを受信したあと、コネクションを解放します。メッセージの送達確認機能を使用していて、受信したメッセージ情報に「応答要求」が設定されていた場合は、メッセージ情報を送信後、コネクションを解放します。

(4) 値が返される引数

- **buff**

受信したメッセージが返されます。メッセージ情報は含みません。タイムアウトが発生すると、タイムアウトまでに受信できたメッセージが格納されます。

- **recvleng**

受信したメッセージの長さが返されます。メッセージ情報の長さは含みません。

タイムアウトが発生すると、タイムアウトまでに受信できたメッセージの長さが格納されます。

(5) リターン値

リターン値	数値 (10進数)	意味
DC_OK	0	正常に終了しました。
DCCLTER_INVALID_ARGS	-2501	引数に指定した値が誤っています。
DCCLTER_PROTO	-2502	次のどちらかの要因が考えられます。 <ul style="list-style-type: none">• dc_rpc_open_s 関数が実行されていません。• dc_rpc_open_s 関数は実行されていますが、引数 flags に DCCLT_ONEWAY_RCV, または DCCLT_SNDRCV を指定していません。
DCCLTER_NO_BUFS	-2504	メモリ不足が発生しました。
DCCLTER_NET_DOWN	-2506	ネットワーク障害が発生しました。コネクションは解放されます。
DCCLTER_TIMED_OUT	-2507	メッセージの受信時にタイムアウトになりました。コネクションは解放されます。
DCCLTER_SYSERR	-2518	システムエラーが発生しました。ネットワーク障害の場合、コネクションは解放されます。
DCCLTER_RESOURCE	-2538	資源不足が発生しました。
DCCLTER_CONNFREE	-2542	相手システムからコネクションが解放されました。
DCCLTER_INVALID_CLTID	-2544	引数 cltid に指定したクライアント ID は、dc_clt_cltin_s 関数で受け取ったクライアント ID と異なります。
DCCLTER_INF_TOO_BIG	-2546	CUP で用意した領域が小さく、相手システムからのメッセージを受信することができません。コネクションは解放されます。
DCCLTER_INVALID_MESSAGE	-2548	不正なメッセージを受信しました。コネクションは解放されます。

(6) 注意事項

- この関数は、次に示す場合にだけ CUP に制御を戻します。
 - メッセージ情報に設定されたメッセージ長分のメッセージを受信した場合
 - ネットワーク障害が発生した場合
 - メッセージ受信時に、タイムアウトが発生した場合
 - 相手システムからコネクションが解放された場合
 - メッセージを格納する領域 (引数 buff) が小さく、通信相手が送信するメッセージを格納できない場合
 - 不正なメッセージを受信した場合

- メッセージの組み立て機能およびメッセージの送達確認機能を使用する場合は、短いパケットで送受信するため、送信処理に時間が掛かることがあります。送信処理に時間が掛かる場合は、クライアント環境定義 DCCLTTCPNODELAY に Y を指定してください。

4.7 サーバからの一方通知受信機能

4.7.1 dc_clt_accept_notification_s – 一方通知メッセージの受信

(1) 形式

(a) TP1/Client/W の場合

■ _s 付き関数

```
#include <dcvclt.h>
DCLONG dc_clt_accept_notification_s(
    HWND hWnd, char *defpath,
    char *inf,
    DCLONG *inf_len,
    unsigned short port, DCLONG timeout,
    char *hostname,
    char *nodeid, DCLONG flags)
```

■ _s 無し関数

```
#include <dcvclt.h>
DCLONG dc_clt_accept_notification(
    char *inf, DCLONG *inf_len,
    unsigned short port, DCLONG timeout,
    char *hostname, char *nodeid,
    DCLONG flags)
```

(b) TP1/Client/P の場合

■ _s 付き関数

```
#include <dcvclt.h>
DCLONG dc_clt_accept_notification_s(
    HWND hWnd, char CLTFAR *defpath,
    char CLTFAR *inf,
    DCLONG CLTFAR *inf_len,
    unsigned short port, DCLONG timeout,
    char CLTFAR *hostname,
    char CLTFAR *nodeid, DCLONG flags)
```

■ _s 無し関数

```
#include <dcvclt.h>
DCLONG dc_clt_accept_notification(
    char CLTFAR *inf,
    DCLONG CLTFAR *inf_len,
    unsigned short port, DCLONG timeout,
```

```
char CLTFAR *hostname,  
char CLTFAR *nodeid, DCLONG flags)
```

(2) 機能

サーバ側の関数 (dc_rpc_cltsend) によって通知されるメッセージを、引数 timeout で指定した値まで待ち続けます。受信した時点で CUP に制御を戻し、リターンコード、通知メッセージ、通知元サーバのホスト名、通知元サーバのノード識別子を返します。この関数の発行前に dc_clt_cltin_s 関数および dc_rpc_open_s 関数を発行しておく必要はありません。

(3) UAP で値を設定する引数

- hWnd

NULL を指定します。

- defpath

クライアント環境定義ファイルへのパス名を指定します。パス名には完全パス、またはカレントドライブ・ディレクトリからの相対パスが指定できます。パス名を指定した場合のファイルの読み込み順序を次に示します。

- TP1/Client/P の場合

クライアント環境定義ファイルの読み込み順序は次のとおりです。

- 1.Windows ディレクトリの BETRAN.INI ファイル
- 2.引数 defpath に指定したクライアント環境定義ファイル

定義は、クライアント環境定義ファイルおよび BETRAN.INI ファイルのどちらのファイルに指定しても有効です。

両方のファイルに同じ定義を異なる値で指定した場合は、クライアント環境定義ファイルに指定した値が有効となります。

クライアント環境定義ファイルおよび BETRAN.INI ファイルのどちらにも指定がない場合は、デフォルト値で動作します。

- TP1/Client/W の場合

環境変数に指定されている定義は、すべて無効となります。引数 defpath に指定したクライアント環境定義ファイルに指定されていない定義はデフォルト値で動作します。

また、引数 defpath の先頭に NULL を指定することでパス名を省略できます。省略時の動作を次に示します。

- TP1/Client/P の場合

Windows ディレクトリの BETRAN.INI ファイルをクライアント環境定義ファイルとして動作します。BETRAN.INI ファイルがない場合、または定義ファイルの内容が不正な場合はデフォルト値で動作します。

- TP1/Client/W の場合

環境変数の指定で動作します。環境変数が指定されていない場合は、デフォルト値で動作します。

引数 defpath に指定したクライアント環境定義ファイルがない場合、または定義ファイルの内容が不正な場合の動作を次に示します。

- TP1/Client/P の場合
Windows ディレクトリの BETRAN.INI ファイルをクライアント環境定義ファイルとして動作します。BETRAN.INI ファイルがない場合、または定義ファイルの内容が不正な場合は、デフォルト値で動作します。
- TP1/Client/W の場合
デフォルト値で動作します。環境変数の指定は無効となります。
- inf
サーバからの通知メッセージを格納する領域を指定します。
- inf_len
サーバからの通知メッセージを格納する領域の長さ（引数 inf の長さ）を指定します。0 から DCRPC_MAX_MESSAGE_SIZE* の範囲で指定します。
注※
クライアント環境定義 DCCLTRPCMAXMSGSIZE に 2 以上を指定した場合、DCRPC_MAX_MESSAGE_SIZE の値（1 メガバイト）ではなく、クライアント環境定義 DCCLTRPCMAXMSGSIZE に指定した値になります。
- port
クライアントのポート番号を指定します。5001 から 65535 の範囲で指定します。なお、同一マシン内で、複数のプロセス、または複数のスレッドを同時に実行する場合は、それぞれ異なるポート番号を指定してください。
- timeout
タイムアウト値（秒）を指定します。0 から 65535 の範囲で指定します。
無限に待ち続ける場合は、0 を指定します。
- hostname
通知したサーバのホスト名を格納する、64 バイト*以上の領域を指定します。NULL が指定された場合は、ホスト名を格納しません。
注※
クライアント環境定義 DCCLTOPTION に 00000008 を指定した場合、64 バイトではなく、256 バイトになります。
- nodeid
通知したサーバのノード識別子を格納する、8 バイトの領域を指定します。
- flags
DCNOFLAGS を指定します。

(4) 値が返される引数

- `inf`
サーバからの通知メッセージが返されます。
- `inf_len`
サーバからの通知メッセージ長が返されます。
- `hostname`
通知したサーバのホスト名が返されます。ホスト名への変換に失敗した場合、10進ドット記法のIPアドレスが返されます。NULLを指定した場合は、返されません。
- `nodeid`
通知したサーバのノード識別子が返されます。ノード識別子のフォーマットは次のとおりです。

ノード識別子 (4バイト)	NULL文字 (4バイト)
---------------	---------------

(5) リターン値

リターン値	数値 (10進数)	意味
DC_OK	0	正常終了しました。
DCCLTER_INVALID_ARGS	-2501	引数に指定した値が誤っています。
DCCLTER_FATAL	-2503	初期化に失敗しました。または、クライアント環境定義の指定に誤りがあります。
DCCLTER_NO_BUFS	-2504	必要なバッファが確保できませんでした。
DCCLTER_NET_DOWN	-2506	ネットワーク障害が発生しました。
DCCLTER_TIMED_OUT	-2507	メッセージの受信時にタイムアウトになりました。
DCCLTER_SYSERR	-2518	システムエラーが発生しました。
DCCLTER_VERSION	-2535	バージョン不一致が発生しました。
DCCLTER_INF_TOO_BIG	-2546	受信したメッセージが、CUPで用意した領域に収まりません。収まり切らないメッセージは切り捨てました。引数 <code>hostname</code> および <code>nodeid</code> には値を設定済みです。
DCCLTER_PORT_IN_USE	-2547	指定したポート番号は使用されています。
DCCLTER_INVALID_MESSAGE	-2548	不正なメッセージを受信しました。
DCCLTER_ACCEPT_CANCELED	-2549	一方通知受信待ち状態が <code>dc_clt_cancel_notification_s</code> 関数によって解除されました。引数 <code>inf</code> , <code>inf_len</code> , および <code>hostname</code> には値を設定済みです。

(6) 注意事項

- 引数 hostname には 64 バイト^{*}以上、引数 nodeid には 8 バイト以上の領域を指定してください。領域がそれより小さい場合、TP1/Client 内部の処理で領域破壊を起こすおそれがあります。

注※

クライアント環境定義 DCCLTOPTION に 00000008 を指定した場合、64 バイトではなく、256 バイトになります。

- 同一マシン内で、複数のプロセス、または複数のスレッドを同時に実行する場合、引数 port にはそれぞれ異なるポート番号を指定してください。また、引数 port に指定できるポート番号でも、OS またはほかのプログラムが使用するポート番号は指定しないでください。指定した場合、応答データを正しく受信できないことがあります。なお、OS が使用するポート番号は、OS ごとに異なります。OS が使用するポート番号については、OS のマニュアルなどを参照してください。
- TP1/Client では、dc_clt_accept_notification_s 関数の呼び出しごとにクライアント環境定義を定義できます。dc_clt_accept_notification_s 関数の呼び出しごとにクライアント環境定義を定義するには、dc_clt_accept_notification_s 関数の呼び出しごとに、異なるファイルをクライアント環境定義ファイルとして作成して、そのファイル名を dc_clt_accept_notification_s 関数の引数 defpath に指定してください。

4.7.2 dc_clt_cancel_notification_s – 一方通知待ち状態のキャンセル

(1) 形式

(a) TP1/Client/W の場合

■ _s 付き関数

```
#include <dcvclt.h>
DCLONG dc_clt_cancel_notification_s(
    HWND hWnd, char *defpath,
    char *inf, DCLONG inf_len,
    unsigned short port,
    char *hostname, DCLONG flags)
```

■ _s 無し関数

```
#include <dcvclt.h>
DCLONG dc_clt_cancel_notification(
    char *inf, DCLONG inf_len,
    unsigned short port, char *hostname,
    DCLONG flags)
```

(b) TP1/Client/P の場合

■ _s 付き関数

```
#include <dcvclt.h>
DCLONG dc_clt_cancel_notification_s(
    HWND hWnd, char CLTFAR *defpath,
    char CLTFAR *inf, DCLONG inf_len,
    unsigned short port,
    char CLTFAR *hostname, DCLONG flags)
```

■ _s 無し関数

```
#include <dcvclt.h>
DCLONG dc_clt_cancel_notification(
    char CLTFAR *inf, DCLONG inf_len,
    unsigned short port, char CLTFAR *hostname,
    DCLONG flags)
```

(2) 機能

dc_clt_accept_notification_s 関数、または dc_clt_chained_accept_notification_s 関数を発行し、サーバからの一方通知待ち状態になっている CUP の、待ち状態をキャンセルします。解除するときに、引数 inf に指定したメッセージを一方通知受信待ち状態の CUP に通知できます。

(3) UAP で値を設定する引数

- hWnd

NULL を指定します。

- defpath

クライアント環境定義ファイルへのパス名を指定します。パス名には完全パス、またはカレントドライブ・ディレクトリからの相対パスが指定できます。パス名を指定した場合のファイルの読み込み順序を次に示します。

- TP1/Client/P の場合

クライアント環境定義ファイルの読み込み順序は次のとおりです。

- 1.Windows ディレクトリの BETRAN.INI ファイル
- 2.引数 defpath に指定したクライアント環境定義ファイル

定義は、クライアント環境定義ファイルおよび BETRAN.INI ファイルのどちらのファイルに指定しても有効です。

両方のファイルに同じ定義を異なる値で指定した場合は、クライアント環境定義ファイルに指定した値が有効となります。

クライアント環境定義ファイルおよび BETRAN.INI ファイルのどちらにも指定がない場合は、デフォルト値で動作します。

- TP1/Client/W の場合

環境変数に指定されている定義は、すべて無効となります。引数 defpath に指定したクライアント環境定義ファイルに指定されていない定義はデフォルト値で動作します。

また、引数 defpath の先頭に NULL を指定することでパス名を省略できます。省略時の動作を次に示します。

- TP1/Client/P の場合

Windows ディレクトリの BETRAN.INI ファイルをクライアント環境定義ファイルとして動作します。BETRAN.INI ファイルがない場合、または定義ファイルの内容が不正な場合はデフォルト値で動作します。

- TP1/Client/W の場合

環境変数の指定で動作します。環境変数が指定されていない場合は、デフォルト値で動作します。

引数 defpath に指定したクライアント環境定義ファイルがない場合、または定義ファイルの内容が不正な場合の動作を次に示します。

- TP1/Client/P の場合

Windows ディレクトリの BETRAN.INI ファイルをクライアント環境定義ファイルとして動作します。BETRAN.INI ファイルがない場合、または定義ファイルの内容が不正な場合は、デフォルト値で動作します。

- TP1/Client/W の場合

デフォルト値で動作します。環境変数の指定は無効となります。

- inf

CUP に通知するメッセージを指定します。

- inf_len

メッセージ長 (inf の長さ) を指定します。0 から DCRPC_MAX_MESSAGE_SIZE^{*} の範囲で指定します。

0 を指定した場合は CUP にメッセージを通知しません。

注※

クライアント環境定義 DCCLTRPCMAXMSGSIZE に 2 以上を指定した場合、DCRPC_MAX_MESSAGE_SIZE の値 (1 メガバイト) ではなく、クライアント環境定義 DCCLTRPCMAXMSGSIZE に指定した値になります。

- port

一方通知受信要求時に指定したポート番号を指定します。5001 から 65535 の範囲で指定します。

- hostname

一方通知受信待ち状態の CUP が存在するホスト名を指定します。ホスト名として指定できる長さは、63 文字^{*}までです。

ホスト名として、10 進ドット記法の IP アドレスを指定することもできます。

注※

クライアント環境定義 DCCLTOPTION に 00000008 を指定した場合、ホスト名として指定できる長さは 255 文字までとなります。

- flags

DCNOFLAGS を指定します。

(4) リターン値

リターン値	数値 (10 進数)	意味
DC_OK	0	正常終了しました。
DCCLTER_INVALID_ARGS	-2501	引数に指定した値が誤っています。
DCCLTER_FATAL	-2503	初期化に失敗しました。または、クライアント環境定義の指定に誤りがあります。
DCCLTER_NO_BUFS	-2504	必要なバッファが確保できませんでした。または、リソース不足が発生しました。
DCCLTER_NET_DOWN	-2506	ネットワーク障害が発生しました。
DCCLTER_SERVICE_NOT_UP	-2514	CUP が一方通知受信待ち状態ではありません。
DCCLTER_SYSERR	-2518	システムエラーが発生しました。
DCCLTER_WRONG_HOST	-2539	ホスト名が不正です。
DCCLTER_PORT_IN_USE	-2547	OS が自動的に割り当てるポート番号が不足しています。

(5) 注意事項

TP1/Client では、dc_clt_cancel_notification_s 関数の呼び出しごとにクライアント環境定義を定義できます。dc_clt_cancel_notification_s 関数の呼び出しごとにクライアント環境定義を定義するには、dc_clt_cancel_notification_s 関数の呼び出しごとに、異なるファイルをクライアント環境定義ファイルとして作成して、そのファイル名を dc_clt_cancel_notification_s 関数の引数 defpath に指定してください。

4.7.3 dc_clt_open_notification_s – 一方通知受信の開始

(1) 形式

(a) TP1/Client/W の場合

■ _s 付き関数

```
#include <dcvclt.h>
DCLONG dc_clt_open_notification_s(HWND hWnd,
                                   DCCLT_ID *ntfid,
                                   char *defpath,
                                   unsigned short port,
                                   DCLONG flags)
```

■ _s 無し関数

```
#include <dcvclt.h>
DCLONG dc_clt_open_notification(unsigned short port,
                                DCLONG flags)
```

(b) TP1/Client/P の場合

■ _s 付き関数

```
#include <dcvclt.h>
DCLONG dc_clt_open_notification_s(HWND hWnd,
                                   DCCLT_ID CLTFAR *ntfid,
                                   char CLTFAR *defpath,
                                   unsigned short port,
                                   DCLONG flags)
```

■ _s 無し関数

```
#include <dcvclt.h>
DCLONG dc_clt_open_notification(unsigned short port,
                                DCLONG flags)
```

(2) 機能

サーバからの一方通知受信機能を使用するための環境を作成します。

この関数は、dc_clt_close_notification_s 関数と対で発行します。

(3) UAP で値を設定する引数

- hWnd
NULL を指定します。

- ntfid

一方通知受信 ID を受け取る領域へのポインタを指定します。

- defpath

クライアント環境定義ファイルへのパス名を指定します。パス名には完全パス、またはカレントドライブ・ディレクトリからの相対パスが指定できます。パス名を指定した場合のファイルの読み込み順序を次に示します。

- TP1/Client/P の場合

クライアント環境定義ファイルの読み込み順序は次のとおりです。

1.Windows ディレクトリの BETRAN.INI ファイル

2.引数 defpath に指定したクライアント環境定義ファイル

定義は、クライアント環境定義ファイルおよび BETRAN.INI ファイルのどちらのファイルに指定しても有効です。

両方のファイルに同じ定義を異なる値で指定した場合は、クライアント環境定義ファイルに指定した値が有効となります。

クライアント環境定義ファイルおよび BETRAN.INI ファイルのどちらにも指定がない場合は、デフォルト値で動作します。

- TP1/Client/W の場合

環境変数に指定されている定義は、すべて無効となります。引数 defpath に指定したクライアント環境定義ファイルに指定されていない定義はデフォルト値で動作します。

また、引数 defpath の先頭に NULL を指定することでパス名を省略できます。省略時の動作を次に示します。

- TP1/Client/P の場合

Windows ディレクトリの BETRAN.INI ファイルをクライアント環境定義ファイルとして動作します。BETRAN.INI ファイルがない場合、または定義ファイルの内容が不正な場合はデフォルト値で動作します。

- TP1/Client/W の場合

環境変数の指定で動作します。環境変数が指定されていない場合は、デフォルト値で動作します。

引数 defpath に指定したクライアント環境定義ファイルがない場合、または定義ファイルの内容が不正な場合の動作を次に示します。

- TP1/Client/P の場合

Windows ディレクトリの BETRAN.INI ファイルをクライアント環境定義ファイルとして動作します。BETRAN.INI ファイルがない場合、または定義ファイルの内容が不正な場合は、デフォルト値で動作します。

- TP1/Client/W の場合

デフォルト値で動作します。環境変数の指定は無効となります。

- port

クライアントのポート番号を指定します。5001 から 65535 の範囲で指定します。なお、同一マシン内で、複数のプロセス、または複数のスレッドを同時に実行する場合は、それぞれ異なるポート番号を指定してください。

- flags

DCNOFLAGS を指定します。

(4) 値が返される引数

- ntfid

一方通知受信 ID が返されます。

(5) リターン値

リターン値	数値 (10 進数)	意味
DC_OK	0	正常終了しました。
DCCLTER_INVALID_ARGS	-2501	引数に指定した値が誤っています。
DCCLTER_PROTO	-2502	dc_clt_open_notification 関数がすでに実行されています。dc_clt_open_notification_s 関数実行時は、このリターン値は返りません。
DCCLTER_FATAL	-2503	初期化に失敗しました。または、クライアント環境定義の指定に誤りがあります。
DCCLTER_NO_BUFS	-2504	必要なバッファが確保できませんでした。
DCCLTER_PORT_IN_USE	-2547	指定したポート番号は使用されています。

(6) 注意事項

- この関数が正常終了した場合は必ず dc_clt_close_notification_s 関数を発行してください。dc_clt_close_notification_s 関数を発行しなかった場合、リソースが残ることがあります。
- 同一マシン内で、複数のプロセス、または複数のスレッドを同時に実行する場合、引数 port にはそれぞれ異なるポート番号を指定してください。また、引数 port に指定できるポート番号でも、OS またはほかのプログラムが使用するポート番号は指定しないでください。指定した場合、応答データを正しく受信できないことがあります。なお、OS が使用するポート番号は、OS ごとに異なります。OS が使用するポート番号については、OS のマニュアルなどを参照してください。
- TP1/Client では、dc_clt_open_notification_s 関数の呼び出しごとにクライアント環境定義を定義できます。dc_clt_open_notification_s 関数の呼び出しごとにクライアント環境定義を定義するには、dc_clt_open_notification_s 関数の呼び出しごとに、異なるファイルをクライアント環境定義ファイルとして作成して、そのファイル名を dc_clt_open_notification_s 関数の引数 defpath に指定してください。

4.7.4 dc_clt_close_notification_s – 一方通知受信の終了

(1) 形式

(a) _s 付き関数の場合

```
#include <dcvclt.h>
DCLONG dc_clt_close_notification_s(DCCLT_ID ntfid, DCLONG flags)
```

(b) _s 無し関数の場合

```
#include <dcvclt.h>
DCLONG dc_clt_close_notification(DCLONG flags)
```

(2) 機能

サーバからの一方通知受信機能を使用するための環境を削除します。

この関数は、dc_clt_open_notification_s 関数と対で発行します。

(3) UAP で値を設定する引数

- ntfid
dc_clt_open_notification_s 関数で受け取った一方通知受信 ID を指定します。
- flags
DCNOFLAGS を指定します。

(4) リターン値

リターン値	数値 (10 進数)	意味
DC_OK	0	正常終了しました。
DCCLTER_INVALID_ARGS	-2501	引数に指定した値が誤っています。
DCCLTER_NO_BUFS	-2504	必要なバッファが確保できませんでした。
DCCLTER_INVALID_NTFFID	-2544	ntfid に指定した一方通知受信 ID は、 dc_clt_open_notification_s 関数で受け取った一方通知受信 ID と異なっています。

4.7.5 dc_clt_chained_accept_notification_s – 一方通知受信

(1) 形式

(a) TP1/Client/W の場合

■ _s 付き関数

```
#include <dcvclt.h>
DCLONG dc_clt_chained_accept_notification_s
    (DCCLT_ID ntfid, char *inf,
     DCLONG *inf_len,
     DCLONG timeout,
     char *hostname,
     char *nodeid,
     DCLONG flags)
```

■ _s 無し関数

```
#include <dcvclt.h>
DCLONG dc_clt_chained_accept_notification
    (char *inf, DCLONG *inf_len,
     DCLONG timeout, char *hostname,
     char *nodeid, DCLONG flags)
```

(b) TP1/Client/P の場合

■ _s 付き関数

```
#include <dcvclt.h>
DCLONG dc_clt_chained_accept_notification_s
    (DCCLT_ID ntfid, char CLTFAR *inf,
     DCLONG CLTFAR *inf_len,
     DCLONG timeout,
     char CLTFAR *hostname,
     char CLTFAR *nodeid,
     DCLONG flags)
```

■ _s 無し関数

```
#include <dcvclt.h>
DCLONG dc_clt_chained_accept_notification
    (char CLTFAR *inf, DCLONG CLTFAR *inf_len,
     DCLONG timeout, char CLTFAR *hostname,
     char CLTFAR *nodeid, DCLONG flags)
```

(2) 機能

サーバ側の関数 dc_rpc_cltsend によって通知されるメッセージを、引数 timeout で指定した値まで待ち続けます。受信した時点で CUP に制御を戻し、リターン値、通知メッセージ、通知メッセージ長、通知

元サーバのホスト名、通知元サーバのノード識別子を返します。この関数を発行する前に dc_clt_open_notification_s 関数を発行しておく必要があります。

(3) UAP で値を設定する引数

- ntfid
dc_clt_open_notification_s 関数で受け取った一方通知受信 ID を指定します。
- inf
サーバからの通知メッセージを格納する領域を指定します。
- inf_len
サーバからの通知メッセージを格納する領域の長さ（引数 inf の長さ）を指定します。0 から DCRPC_MAX_MESSAGE_SIZE* の範囲で指定します。
注※
クライアント環境定義 DCCLTRPCMAXMSGSIZE に 2 以上を指定した場合、DCRPC_MAX_MESSAGE_SIZE の値（1 メガバイト）ではなく、クライアント環境定義 DCCLTRPCMAXMSGSIZE に指定した値になります。
- timeout
タイムアウト値（秒）を指定します。0 から 65535 の範囲で指定します。
無限に待ち続ける場合は、0 を指定します。
- hostname
通知したサーバのホスト名を格納する、64 バイト*以上の領域を指定します。NULL が指定された場合は、ホスト名を格納しません。
注※
クライアント環境定義 DCCLTOPTION に 00000008 を指定した場合、64 バイトではなく、256 バイトになります。
- nodeid
通知したサーバのノード識別子を格納する、8 バイトの領域を指定します。
- flags
DCNOFLAGS を指定します。

(4) 値が返される引数

- inf
サーバからの通知メッセージが返されます。
- inf_len
サーバからの通知メッセージ長が返されます。
- hostname

通知したサーバのホスト名が返されます。

ホスト名への変換に失敗した場合、10進ドット記法のIPアドレスが返されます。NULLを指定した場合は、返されません。

- **nodeid**

通知したサーバのノード識別子が返されます。ノード識別子のフォーマットは次のとおりです。

ノード識別子 (4バイト)	NULL文字 (4バイト)
---------------	---------------

(5) リターン値

リターン値	数値 (10進数)	意味
DC_OK	0	正常終了しました。
DCCLTER_INVALID_ARGS	-2501	引数に指定した値が誤っています。
DCCLTER_PROTO	-2502	dc_clt_open_notification_s 関数が実行されていません。
DCCLTER_NO_BUFS	-2504	必要なバッファが確保できませんでした。
DCCLTER_NET_DOWN	-2506	ネットワーク障害が発生しました。
DCCLTER_TIMED_OUT	-2507	メッセージの受信時にタイムアウトになりました。
DCCLTER_SYSERR	-2518	システムエラーが発生しました。
DCCLTER_VERSION	-2535	バージョン不一致が発生しました。
DCCLTER_INVALID_NTFID	-2544	ntfid に指定した一方通知受信 ID は、 dc_clt_open_notification_s 関数で受け取った一方通知受信 ID と異なっています。
DCCLTER_INF_TOO_BIG	-2546	受信したメッセージが、CUP で用意した領域に収まりません。収まらないメッセージは切り捨てました。引数 hostname, および nodeid には値を設定済みです。
DCCLTER_INVALID_MESSAGE	-2548	不正なメッセージを受信しました。
DCCLTER_ACCEPT_CANCELED	-2549	一方通知受信待ち状態が dc_clt_cancel_notification_s 関数によって解除されました。引数 inf, inf_len, および hostname には値を設定済みです。

(6) 注意事項

引数 hostname には 64 バイト※以上、引数 nodeid には 8 バイト以上の領域を指定してください。領域がこれより小さい場合、TP1/Client 内部の処理で領域破壊を起こすおそれがあります。

注※

クライアント環境定義 DCCLTOPTION に 00000008 を指定した場合、64 バイトではなく、256 バイトになります。

4.8 XATMI インタフェース機能

4.8.1 tmalloc – 型付きバッファの割り当て

(1) 形式

(a) TP1/Client/W の場合

```
#include <dcvxatmi.h>
char *tmalloc(char *type, char *subtype, DCLONG size)
```

(b) TP1/Client/P の場合

```
#include <dcvxatmi.h>
char CLTFAR *tmalloc(char CLTFAR *type, char CLTFAR *subtype, DCLONG size)
```

(2) 機能

型付きのバッファを割り当てます。

バッファの型によっては、使用する前に初期化が必要なものがあります。tmalloc 関数は、バッファが割り当てられてからリターンするまでの間に、バッファを初期化します。関数を呼び出せる状態になってから、バッファは tmalloc 関数を呼び出した側にリターンされます。

初期化の方法は、TP1/Client および TP1/Server Base のコミュニケーションリソースマネージャ (CRM) で定義します。定義されていない場合、tmalloc 関数はバッファを初期化しません。

初期化が正常に完了した場合、tmalloc 関数は、long ワードに位置合わせした適切な形式のバッファへのポインタをリターンします。エラー時には、NULL をリターンし、リターン値としてエラー情報が返されます。初期化が失敗すると、割り当てられたバッファは解放され、NULL をリターンします。

(3) UAP で値を設定する引数

- **type**
バッファの型 (タイプ) を指定します。
X_OCTET を指定してください。
- **subtype**
バッファのサブタイプを指定します。
NULL を指定してください。
- **size**
割り当てるバッファのサイズを指定します。

(4) リターン値

正常に完了した場合、tpalloc 関数は、long ワードに位置合わせした適切な形式のバッファへのポインタをリターンします。エラー時には、NULL をリターンし、リターン値としてエラー情報を示す次のどれか一つの値を、tpermo に設定します。

リターン値	意味
TPEINVAL	引数に誤りがあります。
TPENOENT	引数に指定された値は、システムで定義されていません。
TPEPROTO	tpalloc 関数を呼び出すときの状態が適切ではありません。
TPESYSTEM	コミュニケーションリソースマネージャでエラーが発生しました。
TPEOS	オペレーティングシステムでエラーが発生しました。

(5) 注意事項

- tpalloc 関数は、C 言語の malloc 関数、realloc 関数、free 関数と一緒に使用できません。
例：tpalloc 関数で割り当てたバッファを free 関数で解放する など
上記の関数を一緒に使用した場合、システムの動作は保証できません。
- tpalloc 関数がリターンしたバッファは、0 で初期化されています。
- バッファの領域はグローバルヒープから取得します。
- リターン値として TPESYSTEM が返されたとき、TP1/Client で発生した障害の場合は、エラー情報がエラーログに出力されます。
- リターン値として TPEOS が返されたとき、メモリ不足が原因の場合があります。TP1/Client で発生した障害の場合、エラー情報がエラーログに出力されます。

4.8.2 tpfree – 型付きバッファの解放

(1) 形式

(a) TP1/Client/W の場合

```
#include <dcvxatmi.h>
void tpfree(char *ptr)
```

(b) TP1/Client/P の場合

```
#include <dcvxatmi.h>
void tpfree(char CLTFAR *ptr)
```

(2) 機能

tpalloc で割り当てた型付きのバッファを解放します。

tpfree 関数は、呼び出し元にはリターン値を返しません。関数は void 型で指定します。

引数 ptr に NULL が指定された場合は、解放処理はしません。ptr に指定された値が型付きバッファへのポインタでない場合、または tpfree 関数ですでに解放されていた場合は、tpfree 関数の処理結果は保証されません。

情報を要求するためのバッファ型、またはデータと結び付いているバッファ型は、バッファの解放と同時にそれらの付加情報も削除されます。tpfree 関数はバッファが解放される前に、付加情報の結び付きを削除します。付加情報の削除方法は、TP1/Client および TP1/Server Base のコミュニケーションリソースマネージャ (CRM) で定義します。

いったん tpfree 関数がリターンしたあとに、ptr に指定されていた引数を、新たな引数として XATMI インタフェースの関数に渡したり、参照したりすることはできません。

(3) UAP で値を設定する引数

- ptr

tpalloc 関数で割り当てたバッファへのポインタを指定します。

(4) 注意事項

- tpfree 関数は、C 言語の malloc 関数、realloc 関数、free 関数と一緒に使用できません。

例：malloc 関数で割り当てたバッファを tpfree 関数で解放する など

上記の関数と一緒に使用した場合、システムの動作は保証できません。

4.8.3 tpconnect – 会話型サービスとの接続の確立

(1) 形式

(a) TP1/Client/W の場合

```
#include <dcvxatmi.h>
DCLONG tpconnect(char *svc, char *data, DCLONG len,
                 DCLONG flags)
```

(b) TP1/Client/P の場合

```
#include <dcvxatmi.h>
DCLONG tpconnect(char CLTFAR *svc, char CLTFAR *data, DCLONG len,
                 DCLONG flags)
```

(2) 機能

TP1/Client と会話型サービスとの間に、コネクションを確立します。確立するコネクションは、半二重型です。関数が正常に処理されると、コネクションを指定するための記述子がリターンされます。

コネクションの確立処理時に、tpconnect 関数を発行する側は、受信する側のサービス関数に、指定した情報を渡すことができます。発行する側が情報を渡す場合は、data に tmalloc 関数で割り当てたバッファへのポインタを指定し、len に送信データの長さを指定します。

会話型サービスをするときに、データを受信するための関数を呼び出さないで、情報を受け取れます。

(3) UAP で値を設定する引数

- **svc**
要求するサービスのサービス名を指定します。
- **data**
送信データが格納されている型付きバッファへのポインタを指定します。
- **len**
送信データの長さを指定します。送信データの最大長は 500×1024 バイトです。
長さを指定する必要がない場合は、0 を指定してください。長さの指定が必要なバッファの場合、0 は指定してはいけません。
- **flags**
次に示す値を指定します。

TPNOTRAN

関数を発行する側がトランザクションモードで、かつ TPNOTRAN を指定している場合、起動されたサービスは、発行する側のトランザクションには属しません。

関数を発行する側がトランザクションモードで、かつサービスがトランザクション処理できないサーバに属している場合は、TPNOTRAN を必ず指定します。

関数を発行する側がトランザクションモードの場合は、TPNOTRAN が指定されていても、トランザクションタイムアウトが発生します。

TPNOTRAN で起動されたサービスが失敗しても、関数を発行する側のトランザクションには影響しません。

TPSENDONLY

関数を発行する側がデータを送信し、関数で起動されたサービスがデータの受信だけを行えるように、コネクションを最初に確立します。関数で起動されたサービスは、最初にコネクションの制御権を取得します。

TPRECVONLY

関数を発行する側はデータを受信し、関数で起動されたサービスがデータの送信だけを行えるように、コネクションを最初に確立します。関数で起動されたサービスは最初にコネクションの制御権を取得します。

TPSENDONLY または TPRECVONLY は、どちらかを必ず指定してください。

TPNOBLOCK

ブロッキング状態（送信されたメッセージで内部バッファが満杯になったなど）の場合、接続は確立されず、データは送信されません。

TPNOBLOCK が指定されていないで、ブロッキング状態のときは、関数を発行する側は状況が収まるか、タイムアウト（トランザクション、またはブロッキングタイムアウト）が発生するまでブロッキングします。

TPNOTIME

関数を発行する側を無期限にブロックして、ブロッキングタイムアウトが発生しないようにします。

TPNOTIME を指定しても、トランザクションタイムアウトは発生します。

TPSIGRSTRT

シグナルが実行中のシステムコールを中断した場合に、中断したシステムコールを再び呼びます。

(4) リターン値

正常に完了した場合、tpconnect 関数は、確立した接続を指定するための記述子をリターンします。エラー時には、-1 をリターンし、リターン値としてエラー情報を示す次のどれか一つの値を、tpermo に設定します。

リターン値	意味
TPEINVAL	引数に誤りがあります。
TPENOENT	引数に指定された値は、システムで定義されていないため、接続を確立できません。
TPEITYPE	引数に指定されている値は、指定したサービスでは使用できません。
TPELIMIT	未解決の接続が最大数に達したので、呼び出し側の要求は送信されません。
TPETRAN	指定したサービスがトランザクション処理ができないサーバに属しているのに、TPNOTRAN が指定されていません。
TPETIME	タイムアウトが発生しました。 <ul style="list-style-type: none">発行側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ました。プロセスを異常終了します。 トランザクションはロールバックされます。ロールバックが完了するまでは、どの接続でもメッセージ送受信が行われても、TPETIME が返されます。発行側がトランザクションモード以外の場合 TPNOBLOCK も TPNOTIME も指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した。
TPEBLOCK	TPNOBLOCK を指定した tpconnect 関数を呼び出したときに、ブロッキング状態になりました。
TPGOTSIG	シグナルは受信されましたが、TPSIGRSTRT が指定されていません。
TPEPROTO	tpconnect 関数を呼び出すときの状態が適切ではありません。
TPESYSTEM	コミュニケーションリソースマネージャでエラーが発生しました。

リターン値	意味
TPEOS	オペレーティングシステムでエラーが発生しました。

(5) 注意事項

- OpenTP1 では、ブロッキング状態のため通信ができない場合、ネットワークダウンが原因で通信ができない場合と同様に、TPESYSTEM が返されます。
- OpenTP1 では、サービスで使用できない情報を指定した場合、TPESYSTEM が返されます。関数を呼び出した側がトランザクションモードの場合は、トランザクションはロールバックされます。
- OpenTP1 では、X/Open で特に指定されていない場合、トランザクションをロールバックする必要があるエラーは TPESYSTEM です。ただし、TPESYSTEM が返されても、ロールバックされない場合があります。
- OpenTP1 のセキュリティ機能を使用している場合で、サービス要求が認証されなかったときは、TPEPROTO が返されます。UAP トレースの詳細エラーコードで原因を確認してください。
- TP1/NET/OSI-TP-Extended を使った OSI TP 通信をする場合は、会話型サービスは使用できません。OSI TP 通信で会話型サービスを使用した場合、システムの動作は保証しません。
- TP1/Client では、常設コネクション確立機能を使用して常設コネクションが確立されていないかぎり、tpconnect 関数をトランザクション内から発行できません。
- TP1/Client では、トランザクションタイムアウトが発生した場合、CUP 実行プロセスが異常終了し、それ以前に確立されていたすべてのコネクションは切断されます。TPETIME が返されるのは、ブロッキングタイムアウトの場合だけです。
- TPESYSTEM が返されたとき、TP1/Client で発生した障害の場合、エラー情報がエラーログに出力されます。
- リターン値として TPEOS が返されたとき、メモリ不足が原因の場合があります。TP1/Client で発生した障害の場合、エラー情報がエラーログに出力されます。

4.8.4 tpdiscn – 会話型サービスとのコネクションの切断

(1) 形式 (TP1/Client/W, または TP1/Client/P の場合)

```
#include <dcvxatmi.h>
int tpdiscn(DCLONG cd)
```

(2) 機能

会話型サービスとの間のコネクションを切断して、会話型サービスにイベント (TPEV_DISCONIMM) を通知します。

tpdiscon 関数が発行された場合、コネクションはすぐに切断されます。送信先に届いていないデータは捨てられます。会話型サービスが、発行する側のトランザクションに属している場合も、tpdiscon 関数を使用できます。この場合、トランザクションはロールバックします。

tpdiscon 関数は、会話型サービスを開始した側（オリジネータ）からだけ発行できます。会話型サービス内では、tpdiscon 関数は発行できません。また、tpdiscon 関数が発行するとき、発行する側は、コネクションの制御権を持っている必要はありません。

TP1/Client の会話型サービスは、どちらか一方のシステムが通信を完了した場合、コネクションの切断を通知する tpdiscon 関数を呼び出します。

(3) UAP で値を設定する引数

- cd
コネクションの切断を通知する会話型サービスの記述子を指定します。

(4) リターン値

エラー時には、tpdiscon 関数は-1 をリターンし、リターン値としてエラー情報を示す次のどれか一つの値を tperno に設定します。

リターン値	意味
TPEBADDESC	引数に誤りがあります。または、呼び出された会話型サービスの記述子になっています。
TPETIME	タイムアウトが発生しました。指定された記述子は無効になります。
TPEPROTO	tpdiscon 関数を呼び出したときの状態が適切ではありません。
TPESYSTEM	コミュニケーションリソースマネージャでエラーが発生しました。
TPEOS	オペレーティングシステムでエラーが発生しました。

(5) 注意事項

- OpenTP1 では、TPETIME は返されません。
- TP1/NET/OSI-TP-Extended を使用した OSI TP 通信をする場合は、会話型サービスは使用できません。OSI TP 通信で会話型サービスを使用した場合、システムの動作は保証しません。
- TPESYSTEM が返されたとき、TP1/Client で発生した障害の場合、エラー情報がエラーログに出力されます。
- リターン値として TPEOS が返されたとき、メモリ不足が原因の場合があります。TP1/Client で発生した障害の場合、エラー情報がエラーログに出力されます。

4.8.5 tpsend – 会話型サービスへのメッセージの送信

(1) 形式

(a) TP1/Client/W の場合

```
#include <dcvxatmi.h>
int tpsend(DCLONG cd, char *data, DCLONG len,
           DCLONG flags,
           DCLONG *revent)
```

(b) TP1/Client/P の場合

```
#include <dcvxatmi.h>
int tpsend(DCLONG cd, char CLTFAR *data, DCLONG len,
           DCLONG flags,
           DCLONG CLTFAR *revent)
```

(2) 機能

会話型サービスヘデータを送信します。

tpsend 関数は、コネクションの制御権を持つ側から発行できます。

(3) UAP で値を設定する引数

- **cd**
データを送信するときのコネクションを指定します。tpconnect 関数のリターン値で示された記述子を指定してください。
cd にイベントがある場合は、tpsend 関数は呼び出し側のデータを送信しないで、処理を失敗として終了します。
- **data**
送信するデータを格納する型付きバッファへのポインタを指定します。
NULL を指定した場合は、エラーになります。
ここで指定する値は、会話型サービスで定義されている値と一致させてください。
- **len**
送信するデータの長さを指定します。送信データの最大長は 500×1024 バイトです。
データの長さを指定する必要があるバッファへのポインタのアドレスを指定するときは、len に 0 を指定してください。長さを指定する必要がある場合、len に 0 を指定してはいけません。
- **flags**
次に示す値を指定します。

TPRECVONLY

tpsend 関数を発行する側のデータが送信されたあと、発行する側は、コネクションの制御を放棄します。発行する側は、それ以降 tpsend 関数を発行できません。コネクションのほかの端点であるデータ受信側が、tpsend 関数から送信されてきたデータを受信したとき、コネクションの制御を示すイベント (TPEV_SENDOONLY) を受信します。データ受信側は、それ以降 tprecv 関数を発行できません。

TPNOBLOCK

ブロッキング状態の場合 (送信されたメッセージで内部バッファが満杯の場合など)、データやイベントは送信されません。TPNOBLOCK が指定されていないで、ブロッキング状態のときは、tpsend 関数を発行する側は、再び通信ができる状態になるか、タイムアウト (トランザクション、またはブロッキングタイムアウト) が発生するまで、ブロッキングします。

TPNOTIME

tpsend 関数を発行する側を無制限にブロッキングします。ブロッキングタイムアウトは発生しません。ただし、トランザクションタイムアウトは発生します。

TPSIGRSTRT

シグナルが実行中のシステムコールを中断した場合、中断したシステムコールを再び呼び出します。

• revent

イベントを示す型付きバッファへのポインタを指定します。

tpsend 関数でリターンされるイベントを次に示します。

TPEV_DISCONIMM

開始した側が tpdiscn 関数を使用してコネクションをすぐに切断しました。または、通信エラー (例: サーバ、マシン、ネットワークがダウンした など) によってコネクションが切断されました。TPEV_DISCONIMM は、tpdiscn 関数発行によるコネクション切断の場合、開始された側に通知されます。また、通信エラーによるコネクション切断の場合は、開始した側と開始された側の両方に通知されます。

TPEV_SVCERR

開始された側にコネクションの制御権がない状態で tpreturn 関数を呼び出しました。TPEV_SVCERR は、開始した側に通知されます。

TPEV_SVCFAIL

開始された側にコネクションの制御権がない状態で tpreturn 関数を呼び出しました。TPEV_SVCFAIL は、開始した側に通知されます。

さらに、tpreturn 関数は TPFAIL と data なしで呼び出されました。rval には TPFAIL が指定され、data には NULL が指定されています。

これらのイベントは、コネクションがすぐに切断されたことを示します。そのため、送信中のデータは失われます。このコネクションに使用されている記述子は、無効になります。二つのプログラムが同じトランザクションにある場合は、そのトランザクションはロールバックされます。

(4) リターン値

エラー時には、tpsend 関数は-1 をリターンし、リターン値としてエラー情報を示す次のどれか一つの値を、tperno に設定します。

リターン値	意味
TPEINVAL	引数に誤りがあります。
TPEBADDESC	cd で、誤った記述子を指定しています。
TPETIME	タイムアウトが発生しました。 <ul style="list-style-type: none">発行側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ました。トランザクションはロールバックされます。 トランザクションタイムアウトが発生した場合は、トランザクションをロールバックするまでの間、新規のデータ送信、および未決着の応答に対して、TPETIME が返されます。トランザクションモード以外の場合 TPNOBLOCK および TPNOTIME がどちらも指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した。 上記のどちらの場合も、*data に指定した値は変更されません。
TPEEVENT	イベントが発生しました。リターン値は revent に返されます。
TPEBLOCK	TPNOBLOCK を指定した tprecv 関数を呼び出したときに、ブロッキング状態になりました。
TPGOTSIG	シグナルは受信されましたが、TPSIGRSTRT を指定していません。
TPEPROTO	tpsend 関数発行時の状態が適切ではありません。
TPESYSTEM	コミュニケーションリソースマネージャでエラーが発生しました。
TPEOS	オペレーティングシステムでエラーが発生しました。

(5) 注意事項

- OpenTP1 では、TPNOBLOCK は無効となります。また、TPEBLOCK は返されません。OpenTP1 では、ブロッキング状態のため通信ができない場合、ネットワークダウンによって通信ができない場合と同様に、TPESYSTEM を返します。
- OpenTP1 では、TPNOTIME は無効となります。
- TPSIGRSTRT は無効となります。TPSIGRSTRT が指定されているかどうかは、動作に影響しません。シグナル受信時には、処理を中断してシステムコールを再び呼び出します。TPGOTSIG は返されません。
- OpenTP1 では、トランザクションタイムアウトが発生した場合、該当するプロセスは異常終了します。TPETIME が返されるのは、ブロッキングタイムアウトが発生した場合だけです。
- OpenTP1 では、ロールバックする必要があるエラーは、X/Open で指定されている場合以外は、TPESYSTEM です。ただし、TPESYSTEM が返されても、ロールバックしない場合があります。

- OpenTP1 では、サービスを行っている相手が tpdiscn 関数、tpreturn 関数を呼び出している場合でも、tpsend 関数を呼び出すプロセスでイベントを受信していない場合は、tpsend 関数でイベントを通知できません。
- TP1/NET/OSI-TP-Extended を使用した OSI TP 通信をする場合は、会話型サービスは使用できません。OSI TP 通信で会話型サービスを使用した場合、システムの動作は保証しません。
- TP1/Client では、トランザクションタイムアウトが発生した場合、CUP 実行プロセスが異常終了し、それ以前に確立されていたすべてのコネクションは切断されます。TPETIME が返されるのは、ブロッキングタイムアウトが発生した場合だけです。
- TPESYSTEM が返されたとき、TP1/Client で発生した障害の場合は、エラー情報がエラーログに出力されます。
- リターン値として TPEOS が返されたとき、メモリ不足が原因の場合があります。TP1/Client で発生した障害の場合、エラー情報がエラーログに出力されます。

4.8.6 tprecv - 会話型サービスからのメッセージの受信

(1) 形式

(a) TP1/Client/W の場合

```
#include <dcvxatmi.h>
int tprecv(DCLONG cd, char *CLTFAR *data,
           DCLONG CLTFAR *len,
           DCLONG flags, DCLONG CLTFAR *revent)
```

(b) TP1/Client/P の場合

```
#include <dcvxatmi.h>
int tprecv(DCLONG cd, char CLTFAR *CLTFAR *data,
           DCLONG CLTFAR *len,
           DCLONG flags, DCLONG CLTFAR *revent)
```

(2) 機能

会話型サービスから送信されたデータを受信します。

tprecv 関数は、コネクションの制御権を持たない側から発行できます。

tprecv 関数が、revent に TPEV_SVCSUCC か、TPEV_SVCFAIL を指定してリターンした場合、アプリケーションが tpreturn 関数の引数として渡した値は、グローバル変数 tpurcode として参照できます。

(3) UAP で値を設定する引数

- **cd**

データを受信するときのコネクションを指定します。tpconnect 関数のリターン値で示された記述子を指定してください。

- **data**

受信したデータを格納する型付きバッファへのポインタを指定します。

NULL を指定した場合は、エラーになります。

- **len**

受信したデータの長さを指定します。受信データの最大長は 500×1024 バイトです。

指定した値が、tprecv 関数発行前でのバッファの合計の長さより大きい場合、新しい指定値が len に指定されます。受信データがない場合は、0 が指定されます。

- **flags**

次に示す値を指定します。

TPNOCHANGE

data で指定されているバッファの型は変換されません。

受信したデータのバッファの型と、data で指定されたバッファの型とを必ず一致させます。

TPNOCHANGE を指定していない場合、data の値は、受信したデータのバッファの型に変換されます。

TPNOBLOCK

tprecv 関数はデータの到着を待ちません。

データがすでに受信できる状態の場合は、tprecv 関数はデータを受信してリターンします。

TPNOBLOCK を指定しないで、データが受信できない状態の場合は、関数を発行する側はデータが到着するまでブロッキングします。

TPNOTIME

関数を発行する側を無期限にブロッキングします。

ブロッキングタイムアウトは発生しません。ただし、トランザクションタイムアウトは発生します。

TPSIGRSTRT

シグナルが実行中のシステムコールを中断した場合、中断したシステムコールを再び呼び出します。

- **revent**

イベントを格納する型付きバッファへのポインタを指定します。

cd にイベントがある場合は、revent にイベントの種類が返されます。data に指定した値はイベントの TPEV_SVCSUCC, TPEV_SVCFAIL, TPEV_SENDOONLY と一緒に受信できます。

tprecv 関数では次に示すイベントを指定できます。

TPEV_DISCONIMM

会話型サービスを開始した側が、tpdiscon 関数を呼び出してコネクションをすぐに切断しました。または、通信エラーによってコネクションが切断されました。

tpdiscon 関数の発行でコネクションが切断した場合、サービスを開始された側に TPEV_DISCONIMM が返されます。

通信エラーによるコネクション切断で、サーバ、マシン、およびネットワークがダウンした場合は、サービスを開始した側と開始された側の両方に TPEV_DISCONIMM が返されます。

コネクション切断時は TPEV_DISCONIMM はすぐに通知され、転送中のデータは捨てられます。トランザクションはロールバックされます。このとき、コネクションに使われている記述子は無効になります。

TPEV_SENDOONLY

コネクションのほかの端点にあるシステムは、コネクションの制御権を放棄しています。

TPEV_SENDOONLY を受信した側は、データを送信できますが、制御権を放棄するまではデータを受信できません。

TPEV_SVCERR

サービスを開始された側が、tpreturn 関数を発行しています。tpreturn 関数は、処理中に次のようなエラーが発生しました。

- ・ tpreturn 関数に間違った引数が渡されていた
- ・ サービスがコネクションをオープンしている間に、tpreturn 関数が発行された

TPEV_SVCERR が返されたときは、アプリケーションで定義されたデータやリターン値は動作に影響しません。コネクションは切断され、cd の値は無効になります。TPEV_SVCERR がデータを受信する側のトランザクション内で発生した場合は、トランザクションはロールバックされます。

TPEV_SVCFAIL

サービスのほかの端点にある、開始された側のサービスが、アプリケーションで指定されて、完了しないで終了 (tpreturn 関数のリターン値が TPFail) しました。TPEV_SVCFAIL は、開始した側に返されます。

tpreturn 関数発行時、開始された側のサービスがこのコネクションを制御する場合は、サービスは開始した側へ型付きバッファを渡すことができます。

tprecv 関数終了時、サーバはコネクションを切断します。その場合、cd の値は無効になります。TPEV_SVCFAIL がデータを受信する側のトランザクション内で発生した場合は、トランザクションはロールバックします。

TPEV_SVCSUCC

サービスのほかの端点にある、開始された側のサービスが、アプリケーションで指定されて、完了して終了 (tpreturn 関数のリターン値が TPSUCCESS) しました。TPEV_SVCSUCC は、開始した側に返されます。

tprecv 関数終了時、サーバはコネクションを切断します。その場合、cd の値は無効になります。TPEV_SVCSUCC がデータを受信する側のトランザクション内で発生した場合は、トランザクションはコミットまたはロールバックします。このときサーバ側は、処理が発生したトランザクションをコミットまたはロールバックできます。どちらを実行するかは、トランザクションモードによって異なります。

(4) リターン値

エラー時には、tprecv 関数は-1 をリターンし、リターン値としてエラー情報を示す次のどれか一つの値を、tperno に設定します。

リターン値	意味
TPEINVAL	引数に誤りがあります。
TPEBADDESC	cd で、誤った記述子を指定しています。
TPEOTYPE	到着したバッファの型を、tprecv 関数を呼び出した側で認識していません。または、flags に TPNOCHANGE を指定しているのに、data に指定したバッファの型が送信されてきたバッファの型と一致していません。 上記のどちらの場合も data および len の値は変更されません。tprecv 関数を呼び出した側のトランザクションとしてサービスが実行されている場合は、到着したバッファを捨てるまで、トランザクションはロールバックします。 上記のエラー発生時、指定されたイベントは捨てられ、サービスの処理結果は未決着になります。発行側では、すぐにサービスを終了してください。
TPETIME	タイムアウトが発生しました。 <ul style="list-style-type: none">発行側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ました。トランザクションはロールバックされます。 トランザクションタイムアウトが発生した場合は、トランザクションをロールバックするまでの間、新規のデータ送信、および未決着の応答に対して、TPETIME が返されます。トランザクションモード以外の場合 TPNOBLOCK および TPNOTIME がどちらも指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した。 上記のどちらの場合も、data に指定した値は変更されません。
TPEEVENT	イベントが発生しました。リターン値は revent に返されます。
TPEBLOCK	TPNOBLOCK を指定した tprecv 関数を呼び出したときに、ブロッキング状態になりました。
TPGOTSIG	シグナルは受信されましたが、TPSIGRSTRT を指定していません。
TPEPROTO	tpdiscon 関数を呼び出したときの状態が適切ではありません。
TPESYSTEM	コミュニケーションリソースマネージャでエラーが発生しました。
TPEOS	オペレーティングシステムでエラーが発生しました。

(5) 注意事項

- シグナル受信時には、処理を中断してシステムコールを再び呼び出します。TPGOTSIG はリターンされません。TPSIGRSTRT が指定されているかどうかは、システムコールの呼び出しには影響しません。
- OpenTP1 では、トランザクションタイムアウトが発生した場合には、該当するプロセスは異常終了します。TPETIME は、ブロッキングタイムアウトが発生した場合だけ返されます。

- OpenTP1 では、X/Open で指定されていない場合、トランザクションをロールバックする必要があるエラーは TPESYSTEM です。ただし、TPESYSTEM が返されても、ロールバックされない場合があります。
- TP1/NET/OSI-TP-Extended を使用した OSI TP 通信をする場合は、会話型サービスは使用できません。OSI TP 通信で会話型サービスを使った場合、システムの動作は保証しません。
- TP1/Client では、トランザクションタイムアウトが発生した場合、CUP 実行プロセスは異常終了します。それ以前に確立されていたすべてのコネクションは切断されます。また、TPETIME がリターンされるのは、ブロッキングタイムアウトの場合だけです。
- TP1/Client に対して tpconnect 関数を呼び出すことはできません。cd には tpconnect 関数がリターンした記述子だけ指定できます。
- TPESYSTEM が返されたとき、TP1/Client で発生した障害の場合、エラー情報がエラーログに出力されます。
- リターン値として TPEOS が返されたとき、メモリ不足が原因の場合があります。TP1/Client で発生した障害の場合、エラー情報がエラーログに出力されます。

4.9 文字コード変換機能（コードマッピングテーブルを使用しない場合）

文字コード変換機能で提供する関数は、`_s` 無し関数しかありませんが、マルチスレッド環境でも正しく動作します。

文字コード変換機能は、TP1/Client/P でだけ使用できる機能です。

4.9.1 `dc_clt_code_convert` – 文字コード変換

(1) 形式

```
#include <dcvclt.h>
DCLONG dc_clt_code_convert(
    DCLONG request, char CLTFAR *source,
    DCULONG CLTFAR *source_len,
    char CLTFAR *dest,
    DCULONG CLTFAR *dest_len, DCLONG flags)
```

(2) 機能

- JIS コード、またはシフト JIS コードで構成される文字列を、EBCDIC コード、EBCDIK コード、または KEIS コードで構成される文字列に変換します。
- EBCDIC コード、EBCDIK コード、または KEIS コードで構成される文字列を、JIS コード、またはシフト JIS コードで構成される文字列に変換します。

(3) UAP で値を設定する引数

- `request`

変換の方法を次の要求コードで指定します。

`DCCLT_JISSJIS_TO_EBCKEIS`

JIS コード、またはシフト JIS コードで構成される文字列を、EBCDIC コード、EBCDIK コード、または KEIS コードで構成される文字列に変換します。

`DCCLT_EBCKEIS_TO_JISSJIS`

EBCDIC コード、EBCDIK コード、または KEIS コードで構成される文字列を、JIS コード、またはシフト JIS コードで構成される文字列に変換します。

- `source`

変換する文字列を指定します。

- `source_len`

変換する文字列長を指定します。1 から `DCRPC_MAX_MESSAGE_SIZE` までの範囲の長さが指定できます。

- **dest**

変換後の文字列を受け取る領域を指定します。

- **dest_len**

変換後の文字列を受け取る領域長を指定します。1 から DCRPC_MAX_MESSAGE_SIZE までの範囲の長さが指定できます。

- **flags**

変換時の条件を DCNOFLAGS, または次の形式 (指定値の論理和) で指定します。

2, 3, 4, 5 または 6 が先頭になる場合, | (ストローク) は省略してください。

[1] [| 2] [| 3] [| 4] [| 5] [| 6]

1 : {DCCLT_CNV_EBCDIC | DCCLT_CNV_EBCDIK}

2 : {DCCLT_CNV_SPCHAN | DCCLT_CNV_SPCZEN}

3 : {DCCLT_CNV_KEIS78 | DCCLT_CNV_KEIS83}

4 : {DCCLT_CNV_INVSPC | DCCLT_CNV_INVERR}

5 : {DCCLT_CNV_TAB | DCCLT_CNV_NOTAB}

6 : {DCCLT_CNV_CNTL | DCCLT_CNV_NOCNTL}

指定値の説明

DCNOFLAGS

次に示す假定値を使用します。

- EBCDIK コードを使用します。
- 全角スペースを全角スペースのままにします。
- '83 版 KEIS コードを使用します。
- 無効コードがあった場合, エラーにします。
- タブコードを半角コードとして認識しません。直前, または直後のデータが全角コードの場合でもシフトコードは付けられません。
- 制御コードを半角コードとして認識しません。直前, または直後のデータが全角コードの場合でもシフトコードは付けられません。

DCCLT_CNV_EBCDIC

EBCDIC コードを使用します。

DCCLT_CNV_EBCDIK

EBCDIK コードを使用します。

DCCLT_CNV_SPCHAN

全角スペースを半角スペース 2 個に変換します。この指定は, 引数 request に DCCLT_JISSJIS_TO_EBCKEIS を指定した場合だけ有効です。

DCCLT_CNV_SPCZEN

全角スペースを全角スペースのままにします。

DCCLT_CNV_KEIS78

'78 版 KEIS コードを使用します。

DCCLT_CNV_KEIS83

'83 版 KEIS コードを使用します。

DCCLT_CNV_INVSPC

無効コードがあった場合、スペースに変換します。

DCCLT_CNV_INVERR

無効コードがあった場合、エラーにします。

DCCLT_CNV_TAB

タブコードを半角コードとして認識します。直前、または直後のデータが全角コードの場合はシフトコードが付けられます。

DCCLT_CNV_NOTAB

タブコードを半角コードとして認識しません。直前、または直後のデータが全角コードの場合でもシフトコードは付けられません。

DCCLT_CNV_CNTRL

制御コードを半角コードとして認識します。直前、または直後のデータが全角コードの場合はシフトコードが付けられます。

DCCLT_CNV_NOCNTRL

制御コードを半角コードとして認識しません。直前、または直後のデータが全角コードの場合でもシフトコードは付けられません。

(4) 値が返される引数

- dest
変換後の文字列が返されます。
- dest_len
変換後の文字列長が返されます。

(5) リターン値

リターン値	数値 (10 進数)	意味
DC_OK	0	正常に終了しました。
DCCLTER_INVALID_ARGS	-2501	引数に指定した値が誤っています。
DCCLTER_NO_BUFS	-2504	メモリ不足が発生しました。 このリターン値は、変換する文字列に全角文字 (2 バイト) が含まれているのに、全角文字の 1 バイト目までの文字列長が指定されている場合にも返されます。

リターン値	数値 (10進数)	意味
DCCLTER_INVALID_CODE	-2550	文字列中に無効コードがあります。
DCCLTER_OVERFLOW	-2551	変換後の文字列の長さが CUP で用意した領域を超えています。

(6) 注意事項

- 引数 request に DCCLT_EBCKEIS_TO_JISSJIS, 引数 flags に DCCLT_CNV_TAB, または DCCLT_CNV_CNTL を指定した場合, タブコードまたは制御コードを半角コードとして意識させたデータを, あらかじめ用意しておく必要があります。
- コード変換の仕様の詳細については, 「[付録 A コード変換の仕様](#)」を参照してください。

4.10 文字コード変換機能（コードマッピングテーブルを使用する場合）

文字コード変換機能で提供する関数は、`_s` 無し関数しかありませんが、マルチスレッド環境でも正しく動作します。

文字コード変換機能は、TP1/Client/P でだけ使用できる機能です。

4.10.1 `dc_clt_codeconv_open` – 文字コード変換の開始

(1) 形式

(a) TP1/Client/P の場合

```
#include <dcvclt.h>
DCLONG dc_clt_codeconv_open(char CLTFAR *defpath,
                             DCULONG CLTFAR *cnthdl, DCLONG flags)
```

(b) TP1/Client/P(64)の場合

```
#include <dcvclt.h>
DCLONG dc_clt_codeconv_open(char CLTFAR *defpath,
                             DCCLT_ID CLTFAR *cnthdl, DCLONG flags)
```

(2) 機能

文字コード変換を開始し、使用するコードマッピングテーブルをメモリに確保します。

(3) UAP で値を設定する引数

- `defpath`
NULL を指定します。
- `cnthdl`
文字コード変換で使用される制御テーブルのハンドルを受け取る領域へのポインタを指定します。
- `flags`
変換方法を指定します。
`DCNOFLAGS`
コードマッピングテーブルを使用しないで、換算による変換を行います。
`DCCLT_CNV_CommuniNet`
CommuniNet との連携による変換を行います。

(4) 値が返される引数

- cnthdl

メモリ上に確保された文字コード変換制御テーブルのハンドルが返されます。

(5) リターン値

リターン値	数値 (10進数)	意味
DC_OK	0	正常に終了しました。
DCCLTER_INVALID_ARGS	-2501	引数に指定した値が誤っています。
DCCLTER_NO_BUFS	-2504	メモリ不足が発生しました。
DCCLTER_NOFILE	-2557	コードマッピングテーブルがありません。
DCCLTER_NOT_SUPPORTED	-2558	現在使用中のコードマッピングテーブルは未サポートです。 このリターン値は、CommuniNet インストール後、 CommuniNet コードマッピングユーティリティでコードマッピングテーブルの保存を一度も行っていない場合にも返されます。
DCCLTER_FILE_IO	-2559	コードマッピングテーブルの I/O エラーが起きました。

(6) 注意事項

- この機能を使用する場合は、CommuniNet のコードマッピングテーブルが必要です。CommuniNet のコードマッピングユーティリティで、コードマッピングテーブルを作成してから、この機能を使用してください。
- CommuniNet インストール後、一度も CommuniNet コードマッピングユーティリティで保存されていないコードマッピングテーブルは、使用できません。この機能を使用する前に、CommuniNet コードマッピングユーティリティでコードマッピングテーブルを保存してください。
- CommuniNet のコードマッピングテーブルのファイル名は、必ず CMAPEX.TBL とし、この機能を使用する前に Windows のディレクトリ下に格納してください。
- この機能を使用している途中で、CommuniNet のコードマッピングユーティリティによって、コードマッピングテーブルの内容を変更しても、文字コード変換機能の処理には反映されません。
- この機能では、エラーログ、および UAP トレースの情報は取得しません。
- 文字コード変換の開始 (dc_clt_codeconv_open 関数) は、1 回だけ発行して、文字コード変換の実行 (dc_clt_codeconv_exec 関数) をしてください。メモリ不足となるおそれがあるため、文字コード変換の開始は複数回発行しないでください。複数回発行した場合は、発行した回数分、文字コード変換の終了 (dc_clt_codeconv_close 関数) を発行してください。

4.10.2 dc_clt_codeconv_close – 文字コード変換の終了

(1) 形式

(a) TP1/Client/P の場合

```
#include <dcvclt.h>
DCLONG dc_clt_codeconv_close(DCULONG cnthdl, DCLONG flags)
```

(b) TP1/Client/P(64)の場合

```
#include <dcvclt.h>
DCLONG dc_clt_codeconv_close(DCCLT_ID cnthdl, DCLONG flags)
```

(2) 機能

文字コード変換を終了し、コードマッピングテーブルが確保されたメモリ上の領域を解放します。

(3) UAP で値を設定する引数

- cnthdl
dc_clt_codeconv_open 関数で取得した文字コード変換で使用される制御テーブルのハンドルを指定します。
- flags
DCNOFLAGS を指定します。

(4) リターン値

リターン値	数値 (10進数)	意味
DC_OK	0	正常に終了しました。
DCCLTER_INVALID_ARGS	-2501	引数に指定した値が誤っています。
DCCLTER_NO_BUFS	-2504	メモリ不足が発生しました。

(5) 注意事項

- この機能を使用する場合は、CommuniNet のコードマッピングテーブルが必要です。CommuniNet のコードマッピングユーティリティで、コードマッピングテーブルを作成してから、この機能を使用してください。

- CommuniNet インストール後、一度も CommuniNet コードマッピングユーティリティで保存されていないコードマッピングテーブルは、使用できません。この機能を使用する前に、CommuniNet コードマッピングユーティリティでコードマッピングテーブルを保存してください。
- CommuniNet のコードマッピングテーブルのファイル名は、必ず CMAPEX.TBL とし、この機能を使用する前に Windows のディレクトリ下に格納してください。
- この機能を使用している途中で、CommuniNet のコードマッピングユーティリティによって、コードマッピングテーブルの内容を変更しても、文字コード変換機能の処理には反映されません。
- この機能では、エラーログ、および UAP トレースの情報は取得しません。
- 文字コード変換の開始 (dc_clt_codeconv_open 関数) は、1 回だけ発行して、文字コード変換の実行 (dc_clt_codeconv_exec 関数) をしてください。メモリ不足となるおそれがあるため、文字コード変換の開始は複数回発行しないでください。複数回発行した場合は、発行した回数分、文字コード変換の終了 (dc_clt_codeconv_close 関数) を発行してください。

4.10.3 dc_clt_codeconv_exec – 文字コード変換の実行

(1) 形式

(a) TP1/Client/P の場合

```
#include <dcvclt.h>
DCLONG dc_clt_codeconv_exec(DCLONG request, char CLTFAR *source,
                           DCULONG CLTFAR *source_len,
                           char CLTFAR *dest,
                           DCULONG CLTFAR *dest_len,
                           DCULONG cnthdl, DCLONG flags)
```

(b) TP1/Client/P(64)の場合

```
#include <dcvclt.h>
DCLONG dc_clt_codeconv_exec(DCLONG request, char CLTFAR *source,
                           DCULONG CLTFAR *source_len,
                           char CLTFAR *dest,
                           DCULONG CLTFAR *dest_len,
                           DCCLT_ID cnthdl, DCLONG flags)
```

(2) 機能

次に示す文字コード変換を実行します。

JIS コードまたはシフト JIS コードで構成される文字列を、EBCDIC コード、EBCDIK コード、または KEIS コードで構成される文字列に変換します。

また、EBCDIC コード EBCDIK コードまたは KEIS コードで構成される文字列を、JIS コードまたはシフト JIS コードで構成される文字列に変換します。

(3) UAP で値を設定する引数

- **request**

変換の方法を次の要求コードで指定します。

DCCLT_JISSJIS_TO_EBCKEIS

JIS コード、またはシフト JIS コードで構成される文字列を EBCDIC コード、EBCDIK コード、または KEIS コードで構成される文字列に変換します。

DCCLT_EBCKEIS_TO_JISSJIS

EBCDIC コード、EBCDIK コード、または KEIS コードで構成される文字列を、JIS コード、またはシフト JIS コードで構成される文字列に変換します。

- **source**

変換する文字列を指定します。

- **source_len**

変換する文字列の長さを指定します。

1 から DCRPC_MAX_MESSAGE_SIZE までの範囲の長さが指定できます。

- **dest**

変換後の文字列を受け取る領域を指定します。

- **dest_len**

変換後の文字列を受け取る領域の長さを指定します。

1 から DCRPC_MAX_MESSAGE_SIZE までの範囲の長さが指定できます。

- **cnthdl**

dc_clt_codeconv_open() で取得したコード変換で使用する制御テーブルのハンドルを指定します。

- **flags**

変換時の条件を DCNOFLAGS、または次の形式（指定値の論理和）で指定します。

2, 3, 4, 5, または 6 が先頭になる場合、|（ストローク）は省略してください。

[1] [| 2] [| 3] [| 4] [| 5] [| 6]

1: {DCCLT_CNV_EBCDIC | DCCLT_CNV_EBCDIK}

2: {DCCLT_CNV_SPCHAN | DCCLT_CNV_SPCZEN}

3: {DCCLT_CNV_KEIS78 | DCCLT_CNV_KEIS83}

4: {DCCLT_CNV_INVSPC | DCCLT_CNV_INVERR}

5: {DCCLT_CNV_TAB | DCCLT_CNV_NOTAB}

6: {DCCLT_CNV_CNTRL | DCCLT_CNV_NOCNTRL}

指定値の説明

DCNOFLAGS

次に示す假定値を使用します。

- ・ EBCDIK コードを使用します。
- ・ 全角スペースを全角スペースのままにします。
- ・ '83 版 KEIS コードを使用します。
- ・ 無効コードがあった場合、エラーにします。
- ・ タブコードを半角コードとして認識しません。直前、または直後のデータが全角コードの場合でもシフトコードは付けられません。
- ・ 制御コードを半角コードとして認識しません。直前、または直後のデータが全角コードの場合でもシフトコードは付けられません。

DCCLT_CNV_EBCDIC

EBCDIC コードを使用します。

DCCLT_CNV_EBCDIK

EBCDIK コードを使用します。

DCCLT_CNV_SPCHAN

全角スペースを半角スペース 2 個に変換します。

この指定は、引数 request に DCCLT_JISSJIS_TO_EBCKEIS を指定した場合だけ有効です。

DCCLT_CNV_SPCZEN

全角スペースを全角スペースのままにします。

DCCLT_CNV_KEIS78

'78 版 KEIS コードを使用します。

DCCLT_CNV_KEIS83

'83 版 KEIS コードを使用します。

DCCLT_CNV_INVSPC

無効コードがあった場合、スペースに変換します。

DCCLT_CNV_INVERR

無効コードがあった場合、エラーにします。

DCCLT_CNV_TAB

タブコードを半角コードとして認識します。直前、または直後のデータが全角コードの場合はシフトコードが付けられます。

DCCLT_CNV_NOTAB

タブコードを半角コードとして認識しません。直前、または直後のデータが全角コードの場合でもシフトコードは付けられません。

DCCLT_CNV_CNTRL

制御コードを半角コードとして認識します。直前、または直後のデータが全角コードの場合はシフトコードが付けられます。

DCCLT_CNV_NOCNTL

制御コードを半角コードとして認識しません。直前、または直後のデータが全角コードの場合でもシフトコードは付けられません。

(4) 値が返される引数

- dest
変換後の文字列が返されます。
- dest_len
変換後の文字列の長さが返されます。

(5) リターン値

リターン値	数値 (10進数)	意味
DC_OK	0	正常に終了しました。
DCCLTER_INVALID_ARGS	-2501	引数に指定した値が誤っています。
DCCLTER_NO_BUFS	-2504	メモリ不足が発生しました。 このリターン値は、制御テーブルのハンドルの値が不正である場合、および変換する文字列に全角文字（2バイト）が含まれているのに、全角文字の1バイト目までの文字列長が指定されている場合にも返されます。
DCCLTER_INVALID_CODE	-2550	文字列中に無効コードがあります。
DCCLTER_OVERFLOW	-2551	変換後の文字列の長さがCUPで用意した領域を超えています。

(6) 注意事項

- この機能を使用する場合は、CommuniNetのコードマッピングテーブルが必要です。CommuniNetのコードマッピングユーティリティで、コードマッピングテーブルを作成してから、この機能を使用してください。
- CommuniNetインストール後、一度もCommuniNetコードマッピングユーティリティで保存されていないコードマッピングテーブルは、使用できません。この機能を使用する前に、CommuniNetコードマッピングユーティリティでコードマッピングテーブルを保存してください。
- CommuniNetのコードマッピングテーブルのファイル名は、必ずCMAPEX.TBLとし、この機能を使用する前にWindowsのディレクトリ下に格納してください。
- この機能を使用している途中で、CommuniNetのコードマッピングユーティリティによって、コードマッピングテーブルの内容を変更しても、文字コード変換機能の処理には反映されません。
- この機能では、エラーログ、およびUAPトレースの情報は取得しません。

- 文字コード変換の開始 (dc_clt_codeconv_open 関数) は、1 回だけ発行して、文字コード変換の実行 (dc_clt_codeconv_exec 関数) をしてください。メモリ不足となるおそれがあるため、文字コード変換の開始は複数回発行しないでください。複数回発行した場合は、発行した回数分、文字コード変換の終了 (dc_clt_codeconv_close 関数) を発行してください。
- 引数 request に DCCLT_EBCKEIS_TO_JISSJIS, 引数 flags に DCCLT_CNV_TAB, または DCCLT_CNV_CNTL を指定した場合、タブコードまたは制御コードを半角コードとして意識させたデータを、あらかじめ用意しておく必要があります。
- コード変換の仕様の詳細については、「[付録 A コード変換の仕様](#)」を参照してください。

5

ユーザアプリケーションプログラムの作成（COBOL 言語編）

ユーザアプリケーションプログラムの COBOL 言語での作成方法、および翻訳と結合について説明します。

この章では、各要求文を、マルチスレッド環境の要求文（CBLDCCLS(' ')など）を用いて説明します。シングルスレッド環境の要求文を使う場合は、シングルスレッド環境の要求文（CBLDCCLT(' ')など）に置き換えて読んでください。

5.1 COBOL-UAP 作成用プログラムの機能

OpenTP1 の機能を使用するには、TP1/Client のライブラリにある COBOL-UAP 作成用プログラムを使用します。この COBOL-UAP 作成用プログラムの機能について説明します。

CUP は、COBOL2002 の様式でコーディングしてください。

CUP は OpenTP1 のサービス利用プログラム (SUP) と同様にスタブを使用しません。そのため、ユーザデータのコード (コード体系、およびバイトオーダ) はユーザプログラムで変換してください。

5.1.1 各 UAP と機能の対応

TP1/Client の機能と COBOL-UAP 作成用プログラムとの対応を表 5-1 に示します。

DLL を呼び出すときの要求文 (CBLDCCLS(' ')など)、および文字コード変換機能用の要求文は、マルチスレッド環境で使用できます。そのため、文字コード変換機能の要求文以外で DLL を呼び出すときの要求文をサポートしている要求文は、DLL を呼び出すときの要求文を使用することをお勧めします。

ただし、ご使用になられる TP1/Client 製品のプログラムプロダクトによっては、DLL を呼び出すときの要求文をサポートしていない場合がありますので、「リリースノート」でご確認ください。

各 UAP の詳細については「6. TP1/Client で使用できる要求文 (COBOL 言語編)」を参照してください。

表 5-1 TP1/Client の機能と COBOL-UAP 作成用プログラムとの対応

機能		CALL 文で呼び出す COBOL-UAP 作成用プログラム
ユーザ認証機能	クライアントユーザの認証要求	CBLDCCLS(' CLTIN ')
		CBLDCCLT(' CLTIN ')
		CBLDCCLS(' EXCLTIN ')*1
		CBLDCCLT(' EXCLTIN ')*1
	クライアントユーザの認証解除	CBLDCCLS(' CLTOUT ')
		CBLDCCLT(' CLTOUT ')
リモートプロシジャコール	UAP の開始	CBLDCRPS(' OPEN ')
		CBLDCRPC(' OPEN ')
	UAP の終了	CBLDCRPS(' CLOSE ')
		CBLDCRPC(' CLOSE ')
	遠隔サービスの要求	CBLDCRPS(' CALL ')
		CBLDCRPC(' CALL ')

機能		CALL 文で呼び出す COBOL-UAP 作成用プログラム
リモートプロシジャコール	サービス応答待ち時間の更新	CBLDCRPS(' SETWATCH')
		CBLDCRPC(' SETWATCH')
	サービス応答待ち時間の参照	CBLDCRPS(' GETWATCH')
		CBLDCRPC(' GETWATCH')
常設コネクション	常設コネクションの確立	CBLDCCLS(' CONNECT ')
		CBLDCCLT(' CONNECT ')
	常設コネクションの解放	CBLDCCLS(' DISCNCT ')
		CBLDCCLT(' DISCNCT ')
	常設コネクション確立要求先の指定	CBLDCCLS(' STRAPHST')※2
		CBLDCCLT(' STRAPHST')※2
	常設コネクション確立要求先の取得	CBLDCCLS(' GTRAPHST')※2
		CBLDCCLT(' GTRAPHST')※2
	端末識別情報の設定	CBLDCCLS(' STCONINF')
		CBLDCCLT(' STCONINF')
トランザクション制御	トランザクションの開始	CBLDCTRS(' BEGIN ')
		CBLDCTRN(' BEGIN ')
	連鎖モードのコミット	CBLDCTRS(' C-COMMIT')
		CBLDCTRN(' C-COMMIT')
	連鎖モードのロールバック	CBLDCTRS(' C-ROLL ')
		CBLDCTRN(' C-ROLL ')
	非連鎖モードのコミット	CBLDCTRS(' U-COMMIT')
		CBLDCTRN(' U-COMMIT')
	非連鎖モードのロールバック	CBLDCTRS(' U-ROLL ')
		CBLDCTRN(' U-ROLL ')
	現在のトランザクションに関する情報の報告	CBLDCTRS(' INFO ')
		CBLDCTRN(' INFO ')
	現在のトランザクションに関する識別子の取得	CBLDCCLS(' GETTRNID')
		CBLDCCLT(' GETTRNID')
TCP/IP 通信機能	メッセージの送信	CBLDCCLS(' SEND ')
		CBLDCCLT(' SEND ')

機能		CALL 文で呼び出す COBOL-UAP 作成用プログラム
TCP/IP 通信機能	メッセージの送信	CBLDCCLS(' EXSEND ')*1
		CBLDCCLT(' EXSEND ')*1
	メッセージの受信	CBLDCCLS(' RECEIVE ')
		CBLDCCLT(' RECEIVE ')
	メッセージの受信 (障害時メッセージ受信)	CBLDCCLS(' RECEIVE2')
		CBLDCCLT(' RECEIVE2')
組み立てメッセージの送信	CBLDCCLS(' ASMSEND ')	
組み立てメッセージの受信	CBLDCCLS(' ASMRECV ')	
サーバからの一方通知受信機能	一方通知メッセージの受信	CBLDCCLS(' NOTIFY ')
		CBLDCCLT(' NOTIFY ')
		CBLDCCLS(' EXNACPT ')*1
		CBLDCCLT(' EXNACPT ')*1
	一方通知待ち状態のキャンセル	CBLDCCLS(' CANCEL ')
		CBLDCCLT(' CANCEL ')
		CBLDCCLS(' EXNCANCL')*1
		CBLDCCLT(' EXNCANCL')*1
	一方通知受信の開始	CBLDCCLS(' O-NOTIFY')
		CBLDCCLT(' O-NOTIFY')
	一方通知受信の終了	CBLDCCLS(' C-NOTIFY')
		CBLDCCLT(' C-NOTIFY')
	一方通知受信	CBLDCCLS(' A-NOTIFY')
		CBLDCCLT(' A-NOTIFY')
CBLDCCLS(' EXNCACPT')*1		
CBLDCCLT(' EXNCACPT')*1		
文字コード変換機能 (コードマッピングテーブルを使用しない場合)	文字コード変換	CBLDCUTL(' CODECNV ')
文字コード変換機能 (コードマッピングテーブルを使用する場合)	文字コード変換の開始	CBLDCUTL(' CNVOPN ')
	文字コード変換の終了	CBLDCUTL(' CNVCLS ')
	文字コード変換の実行	CBLDCUTL(' CNVEXEC ')

注※1

クライアント環境定義 DCCLTOPTION に 00000008 を指定した場合に使用してください。

注※2

クライアント環境定義 DCCLTOPTION に 00000008 を指定した場合、データ領域を大きくする必要がありますので、注意してください。

5.1.2 COBOL-UAP 作成用プログラムの記述形式

CUP を COBOL 言語で作成するときは、TP1/Client のライブラリにある関数に対応した COBOL-UAP 作成用プログラムを CALL 文で呼び出します。

「6. TP1/Client で使用できる要求文 (COBOL 言語編)」では、COBOL-UAP 作成用プログラムを次の形式で説明します。

形式

ライブラリにある関数と対応する COBOL-UAP 作成用プログラムを CALL 文で呼び出す形式と、領域の指定方法を示します。

データ名に値を指定するときは、ここで示す PICTURE 句のデータ形式に従ってください。指定する値が決まっている場合は VALUE 句で記述してあります。

一意名で示すファイル名とデータ名には、断りがないかぎり、固有の名称を任意に付けてください。データ名として指定する文字の長さなどは、使用する COBOL の仕様に従ってください。

機能

COBOL-UAP 作成用プログラムの機能について説明します。以降、COBOL-UAP 作成用プログラムを、次に示す形式で表記します。

```
CBLDCXXX('YYYYYYYY')
```

CBLDCXXX : COBOL-UAP作成用プログラムのプログラム名を示します。
YYYYYYYY : 要求コードを示します。

UAP で値を設定するデータ領域

DATA DIVISION に指定するデータのうち、COBOL-UAP 作成用プログラムの呼び出し時にデータ領域に値を指定しておくデータ名です。各データ名の説明に従って、値を設定してください。

値が返されるデータ領域

CALL 文を実行したあとに、OpenTP1、サーバ UAP、TP1/Client などから値が返されるデータ名です。

ステータスコード

CALL 文を実行したときに返される値を、表形式で説明します。ステータスコードによって、COBOL-UAP 作成用プログラムが正常に実行されたかがわかります。エラーが発生したときは、エラーの内容を示します。

COBOL 言語のステータスコードは 5 けたの数字列で、USING 句で指定する最初の一意名に含まれます。CALL 文の USING 句で指定する一意名とステータスコードの関係を次に示します。

CALL ' 呼び出すプログラム名' USING 一意名1 一意名2 ...

	8文字	5文字	
一意名1	要求コード	ステータスコード	CALL文で呼び出す プログラムで使う領域
一意名2	CALL文で呼び出す プログラムで使う領域		

注意事項

COBOL-UAP 作成用プログラムを使うときの注意を記述します。

(1) 指定する項目の説明で使う記号

指定する項目の説明で使う記号の一覧を示します。

記号	説明
[]	この記号で囲まれている項目は、省略しても良いことを示します。 (例) [:ポート番号] :ポート番号は省略できることを示します。
...	記述が省略されていることを示します。この記号の直前に示された項目を繰り返し複数個指定できます。 (例) ホスト名 [:ポート番号] [,ホスト名 [:ポート番号] ,...] ホスト名 [:ポート番号] を続けて指定できることを示します。
△	空白を示します。
~	この記号の前に示された項目が、~に続く < >, (())の規則に従うことを示します。
<文字列>	任意の文字の配列を示します。
<符号なし整数>	数字 (0~9) を示します。
(())	指定値の指定範囲を示します。

5.2 ユーザアプリケーションプログラムの翻訳と結合

翻訳と結合の方法は、OS 環境によって異なります。

5.2.1 UNIX 環境の場合の翻訳と結合

(1) 翻訳

COBOL 言語の CUP のオブジェクトファイルを作成するには、ソースプログラムを COBOL コンパイラで翻訳します。

翻訳方法の詳細については、マニュアル「OpenTP1 プログラム作成リファレンス COBOL 言語編」を参照してください。

COBOL2002 を使用してソースプログラムを翻訳するときのコマンドの入力例を次に示します。

〈例〉

COBOL 言語で作成した UAP のソースプログラム

- cupmain.cbl (主プログラム)
- cupfnc1.cbl (副プログラム 1)
- cupfnc2.cbl (副プログラム 2)

この場合、それぞれのソースプログラムは次のように翻訳します。

```
ccbl2002 -Compile,NoLink -Main,System cupmain.cbl
ccbl2002 -Compile,NoLink cupfnc1.cbl
ccbl2002 -Compile,NoLink cupfnc2.cbl
```

注

翻訳環境のロケールが UTF-8 の場合、ccbl2002 のオプションに -UniObjGen も追加してください。

上記 ccbl2002 コマンドを実行すると、次のオブジェクトファイルが作成されます。

- cupmain.o (主プログラムのオブジェクトファイル)
- cupfnc1.o (副プログラム 1 のオブジェクトファイル)
- cupfnc2.o (副プログラム 2 のオブジェクトファイル)

(2) 結合

CUP の実行形式ファイルは、次に示すファイルを結合させて作成します。

- CUP のオブジェクトファイル (主プログラムと副プログラム)
- TP1/Client/W のライブラリ

- COBOL2002 のライブラリ

COBOL2002 を使用して上記のファイルを結合するときのコマンドの入力例を次に示します。

〈例〉

COBOL CUP 実行形式ファイル「example」を作成する場合

- 主プログラムのオブジェクトファイル名…cupmain.o
- 副プログラムのオブジェクトファイル名…cupfnc1.o, cupfnc2.o

この場合、次のようにファイルを結合させます。

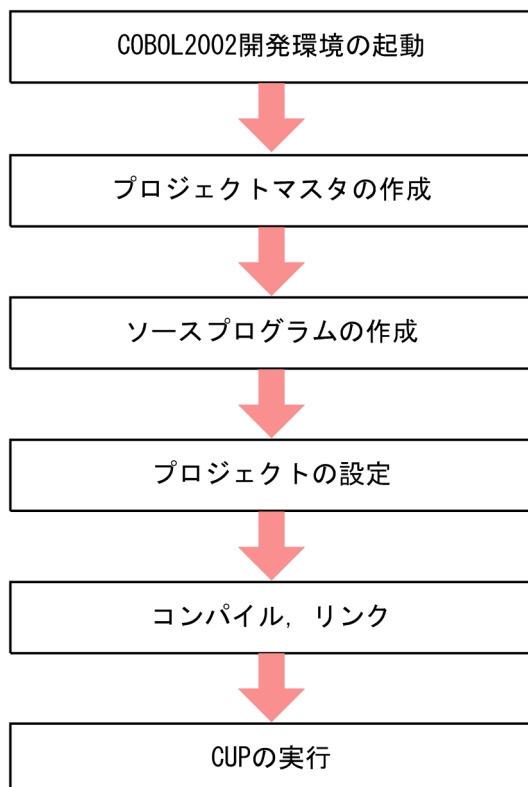
```
ccbl2002 cupmain.o cupfnc1.o cupfnc2.o  
-L/usr/lib -lclt -OutputFile example
```

5.2.2 Windows 環境の場合の翻訳と結合

(1) 手順

COBOL CUP 作成時の手順の例を次の図に示します。

図 5-1 COBOL CUP 新規作成時の手順例



(2) 翻訳, 結合, 実行

(a) COBOL2002 開発環境の起動

COBOL2002 の「開発マネージャ for COBOL2002」を起動します。

(b) プロジェクトマスタの作成

「開発マネージャ for COBOL2002」ウィンドウの「プロジェクトマスタ」メニューから、「新規作成」を選択します。ウィンドウの指示に従って、プロジェクトマスタ名などを入力し、プロジェクトマスタを作成します。

(c) ソースプログラムの作成

次に示す方法でソースプログラムを作成します。

- プロジェクトマスタの作成が完了すると、「開発マネージャ for COBOL2002」上にツリービューウィンドウが表示されます。ソースファイルノードの各ソースファイルを選択すると、エディタのウィンドウが表示されるので、ソースプログラムを作成してください。
なお、Windows 上で COBOL エディタ for COBOL2002 を単独で起動して、ソースプログラムを作成することもできます。
- 32bit 版で文字コード変換機能を使用する場合は、stdcall 呼び出し指示ファイルを作成し、プログラム名として CBLDCUTL を記述してください。stdcall 呼び出し指示ファイル名は、ソースファイル名.cbw とし、ソースファイルと同じフォルダに格納してください。

(d) プロジェクトの設定

次に示す方法でプロジェクトを設定します。

- 「プロジェクト」メニューから、「プロジェクトの設定」を選択し、プロジェクト設定ダイアログボックスを開きます。「リンク」タブから「ライブラリの指定」を選択し、TP1/Client/P のライブラリ (CLTW32.LIB, CLTWS32.LIB または CLTW64.LIB) ※を指定します。
- 32bit 版で文字コード変換機能を使用する場合は、「リンク」タブから「-StdCall:stdcall 呼び出し指示ファイルを有効にする」を選択してください。開発マネージャを使用しないときは、ccbl2002 コマンドのコンパイラオプションに -StdCall を指定してください。

注※

文字コード変換機能を使用する場合は、CLTCNV32.LIB または CLTCNV64.LIB も必要です。

その他のプロジェクト設定は、COBOL2002 のマニュアルを参照してください。

(e) コンパイル, リンク

「開発マネージャ for COBOL2002」ウィンドウの「ビルド」メニューから、「ビルド」を選択します。

自動的に、翻訳とリンケージが行われます。

(f) CUP の実行

「開発マネージャ for COBOL2002」ウィンドウの「ビルド」メニューから、「実行」を選択します。CUPの作成が終わったあとは、「実行支援 for COBOL2002」を経由して実行することもできます。

5.3 COBOL 言語用テンプレート

UAP を COBOL 言語で作成するときに、データ部 (DATA DIVISION) のコーディングの負荷を軽くするため、COBOL 言語用テンプレートを提供します。COBOL 言語用テンプレートは次のディレクトリの下にあります。

- TP1/Client/P
ユーザが指定するディレクトリの下 include ディレクトリの下 TP1COBOL
または前回インストール時に指定されたディレクトリの下 include ディレクトリの下 TP1COBOL
- TP1/Client/W
/usr/include/TP1COBOL

5.3.1 COBOL 言語用テンプレートのファイル

COBOL 言語用テンプレートとして使用できるファイルを次に示します。

- DCCLT.cbl, DCCLS.cbl
ユーザ認証機能, TCP/IP 通信機能, サーバからの一方通知受信機能
- DCRPC.cbl, DCRPS.cbl
リモートプロシジャコール
- DCTRN.cbl, DCTRS.cbl
トランザクション制御
- DCUTL.cbl
コード変換 (TP1/Client/P だけ)

5.3.2 COBOL 言語用テンプレートの使用方法

COBOL 言語用テンプレートを使用するときは、次に示す値をコーディングする UAP の処理に合わせたものに修正する必要があります。

- データ領域のサイズ (一部)
- 各データ領域へ代入する値

データ領域に設定する値については、「[6. TP1/Client で使用できる要求文 \(COBOL 言語編\)](#)」の各機能の記述を参照してください。

COBOL 言語用テンプレートの使用方法は、次に示す 2 通りがあります。

(1) テキストエディタの呼び出し機能を使用する方法

次の手順でテンプレートを使用します。

1. インストール先ディレクトリから、使用する機能のテンプレートを選択します。
2. テキストエディタの呼び出し機能を使用して、DATA DIVISION 部分を切り取り、UAP のソースプログラムにはり付けます。
3. はり付けた部分を、コーディングの処理に合ったデータ領域に修正します。

(2) COBOL 言語の COPY 文を使用する方法

次の手順でテンプレートを使用します。

1. インストール先ディレクトリから、使用する機能のテンプレートを選択します。
2. UAP のソースプログラムから、テンプレートのファイル名で COPY 宣言します。
3. テンプレートのファイルを、COPY 文で参照できるディレクトリに置きます。
この方法は、使用する COBOL 言語の処理系に従ってください（ファイルのコピー、または環境変数の設定など）。
4. テンプレートのファイルを、コーディングの処理に合ったデータ領域に修正します。

5.3.3 COBOL 言語用テンプレートについての使用上の注意

- UAP の処理に合わせて変更するデータ領域では、PICTURE 句の長さを (n) と宣言しています。この部分を修正してから使用してください。修正しないままコンパイルすると、エラーになります。
- UAP の処理に合わせて変更するテンプレートを使用する場合は、元のディレクトリからコピーして使用することをお勧めします。

5.4 ユーザアプリケーションプログラムの作成例

ここでは、UAP を作成する場合の CUP や SPP などのコーディングについて、例を使って説明します。

5.4.1 CUP と SPP の作成

「3.3.1 CUP と SPP の作成」に示す CUP と SPP の構成例の、CUP を COBOL 言語で作成した場合のコーディング例を次に示します。

```
000010      *
000020      *****
000030      *   CUPサンプルプログラム                               *
000040      *****
000050      *
000060      IDENTIFICATION DIVISION.
000070      PROGRAM-ID. CUP01.
000080      *
000090      *****
000100      *   データ領域の設定                                   *
000110      *****
000120      *
000130      DATA DIVISION.
000140      WORKING-STORAGE SECTION.
000150      01  DCCLS-CLTIN-ARG.
000160          02  DCCLS-CLTIN-REQUEST      PIC  X(8) VALUE 'CLTIN  '.
000170          02  DCCLS-CLTIN-STATUS-CODE PIC  X(5).
000180          02  FILLER                   PIC  X(3).
000190          02  DCCLS-CLTIN-FLAGS        PIC  S9(9) COMP VALUE ZERO.
000200          02  DCCLS-CLTIN-T-HOST       PIC  X(64).
000210          02  DCCLS-CLTIN-LOGNAME      PIC  X(16).
000220          02  DCCLS-CLTIN-PASSWD       PIC  X(16).
000230          02  DCCLS-CLTIN-S-HOST       PIC  X(64).
000240          02  DCCLS-CLTIN-HWND         PIC  9(4) COMP.
000250          02  FILLER                   PIC  X(2).
000260          02  DCCLS-CLTIN-CLTID        PIC  9(18) COMP.
000270          02  DCCLS-CLTIN-DEFPATH      PIC  X(256).
000280      *
000290      01  DCCLS-CLTOUT-ARG.
000300          02  DCCLS-CLTOUT-REQUEST     PIC  X(8) VALUE 'CLTOUT  '.
000310          02  DCCLS-CLTOUT-STATUS-CODE PIC  X(5).
000320          02  FILLER                   PIC  X(3).
000330          02  DCCLS-CLTOUT-FLAGS       PIC  S9(9) COMP VALUE ZERO.
000340          02  FILLER                   PIC  X(4).
000350          02  DCCLS-CLTOUT-CLTID       PIC  9(18) COMP.
000360      *
000370      01  DCRPS-OPEN-ARG1.
000380          02  DCRPS-OPEN-REQUEST       PIC  X(8) VALUE 'OPEN  '.
000390          02  DCRPS-OPEN-STATUS-CODE   PIC  X(5).
000400          02  FILLER                   PIC  X(3).
000410          02  DCRPS-OPEN-FLAGS        PIC  S9(9) COMP VALUE ZERO.
000420          02  FILLER                   PIC  X(4).
000430          02  DCRPS-OPEN-CLTID        PIC  9(18) COMP.
000440      *
```

```

000450 01 DCRPS-OPEN-ARG2.
000460 02 FILLER PIC X(1).
000470 *
000480 01 DCRPS-OPEN-ARG3.
000490 02 FILLER PIC X(1).
000500 *
000510 01 DCRPS-CALL-ARG1.
000520 02 DCRPS-CALL-REQUEST PIC X(8) VALUE 'CALL '.
000530 02 DCRPS-CALL-STATUS-CODE PIC X(5).
000540 02 FILLER PIC X(3).
000550 02 DCRPS-CALL-FLAGS PIC S9(9) COMP VALUE ZERO.
000560 02 DCRPS-CALL-DESCRIPTOR PIC S9(9) COMP.
000570 02 DCRPS-CALL-SVGROUP PIC X(32).
000580 02 DCRPS-CALL-SVNAME PIC X(32).
000590 02 DCRPS-CALL-CLTID PIC 9(18) COMP.
000600 *
000610 01 DCRPS-CALL-ARG2.
000620 02 DCRPS-CALL-INDATALEN PIC S9(9) COMP.
000630 02 DCRPS-CALL-INDATA PIC X(512).
000640 *
000650 01 DCRPS-CALL-ARG3.
000660 02 DCRPS-CALL-OUTDATALEN PIC S9(9) COMP.
000670 02 DCRPS-CALL-OUTDATA PIC X(512).
000680 *
000690 01 DCRPS-CLOSE-ARG1.
000700 02 DCRPS-CLOSE-REQUEST PIC X(8) VALUE 'CLOSE '.
000710 02 DCRPS-CLOSE-STATUS-CODE PIC X(5).
000720 02 FILLER PIC X(3).
000730 02 DCRPS-CLOSE-FLAGS PIC S9(9) COMP VALUE ZERO.
000740 02 FILLER PIC X(4).
000750 02 DCRPS-CLOSE-CLTID PIC 9(18) COMP.
000760 *
000770 01 DCRPS-CLOSE-ARG2.
000780 02 FILLER PIC X(1).
000790 *
000800 01 DCRPS-CLOSE-ARG3.
000810 02 FILLER PIC X(1).
000820 *
000830 77 FOREVER-FLAG PIC 9 COMP VALUE ZERO.
000840 77 INDATA PIC X(512) VALUE SPACE.
000850 *
000860 *****
000870 * CUPの開始 *
000880 *****
000890 PROCEDURE DIVISION.
000900 MAIN SECTION.
000910 PROG-START.
000920 *
000930 *****
000940 * クライアントユーザの認証要求 *
000950 *****
000960 MOVE 'CLTIN ' TO DCCLS-CLTIN-REQUEST IN DCCLS-CLTIN-ARG.
000970 MOVE ZERO TO DCCLS-CLTIN-FLAGS IN DCCLS-CLTIN-ARG.
000980 MOVE SPACE TO DCCLS-CLTIN-T-HOST IN DCCLS-CLTIN-ARG.
000990 MOVE 'user01' TO DCCLS-CLTIN-LOGNAME IN DCCLS-CLTIN-ARG.
001000 MOVE 'puser01' TO DCCLS-CLTIN-PASSWD IN DCCLS-CLTIN-ARG.
001010 MOVE ZERO TO DCCLS-CLTIN-HWND IN DCCLS-CLTIN-ARG.
001020 MOVE SPACE TO DCCLS-CLTIN-DEFPATH IN DCCLS-CLTIN-ARG.

```



```

001030 *
001040 * *****
001050 CALL 'CBLDCCLS' USING DCCLS-CLTIN-ARG.
001060 * *****
001070 IF DCCLS-CLTIN-STATUS-CODE
001080             IN DCCLS-CLTIN-ARG NOT = '00000'
001090 THEN
001100     DISPLAY 'CUP01: CBLDCCLS(CLTIN)に失敗しました。CODE='
001110             DCCLS-CLTIN-STATUS-CODE IN DCCLS-CLTIN-ARG
001120     GO TO PROG-EXIT
001130 END-IF.
001140 *
001150 *****
001160 * RPC-OPEN(RPC環境の初期設定) *
001170 *****
001180     MOVE 'OPEN'          TO
001190             DCRPS-OPEN-REQUEST IN DCRPS-OPEN-ARG1.
001200     MOVE ZERO           TO
001210             DCRPS-OPEN-FLAGS   IN DCRPS-OPEN-ARG1.
001220     MOVE DCCLS-CLTIN-CLTID IN DCCLS-CLTIN-ARG TO
001230             DCRPS-OPEN-CLTID  IN DCRPS-OPEN-ARG1.
001240 *
001250 * *****
001260 CALL 'CBLDCRPS' USING DCRPS-OPEN-ARG1
001270             DCRPS-OPEN-ARG2 DCRPS-OPEN-ARG3.
001280 * *****
001290 IF DCRPS-OPEN-STATUS-CODE IN DCRPS-OPEN-ARG1
001300             NOT = '00000'
001310 THEN
001320     DISPLAY 'CUP01: CBLDCRPS(OPEN)に失敗しました。CODE='
001330             DCRPS-OPEN-STATUS-CODE IN DCRPS-OPEN-ARG1
001340     GO TO PROG-END
001350 END-IF.
001360 *
001370 PERFORM UNTIL FOREVER-FLAG NOT = ZERO
001380     DISPLAY '***** 伝言板メニュー *****'
001390     DISPLAY '伝言の取り出し .... [1]'
001400             '伝言の書き込み .... [2]'
001410     DISPLAY '終了 ..... [9]'
001420     DISPLAY '番号を入力して下さい。=>'
001430     ACCEPT INDATA
001440     EVALUATE INDATA
001450     WHEN '1'
001460 *
001470 * *****
001480 * * RPC-CALL(RPCの実行) *
001490 * *****
001500     MOVE 'CALL'          TO
001510             DCRPS-CALL-REQUEST IN DCRPS-CALL-ARG1
001520     MOVE ZERO           TO
001530             DCRPS-CALL-FLAGS   IN DCRPS-CALL-ARG1
001540     MOVE 'spp01'        TO
001550             DCRPS-CALL-SVGROUP IN DCRPS-CALL-ARG1
001560     MOVE 'get'          TO
001570             DCRPS-CALL-SVNAME  IN DCRPS-CALL-ARG1
001580     MOVE DCCLS-CLTIN-CLTID IN DCCLS-CLTIN-ARG TO
001590             DCRPS-CALL-CLTID  IN DCRPS-CALL-ARG1
001600     MOVE 'cup01'        TO

```

```

001610         DCRPS-CALL-INDATA   IN DCRPS-CALL-ARG2
001620     MOVE 512           TO
001630         DCRPS-CALL-INDATALEN IN DCRPS-CALL-ARG2
001640     MOVE SPACE        TO
001650         DCRPS-CALL-OUTDATA   IN DCRPS-CALL-ARG3
001660     MOVE 512           TO
001670         DCRPS-CALL-OUTDATALEN IN DCRPS-CALL-ARG3
001680 *
001690 * *****
001700     CALL 'CBLDCRPS' USING DCRPS-CALL-ARG1
001710                             DCRPS-CALL-ARG2 DCRPS-CALL-ARG3
001720 * *****
001730     IF DCRPS-CALL-STATUS-CODE IN DCRPS-CALL-ARG1
001740                             NOT = '00000'
001750     THEN
001760         DISPLAY 'CUP01: CBLDCRPS(CALL)に失敗しました。CODE='
001770                             DCRPS-CALL-STATUS-CODE IN DCRPS-CALL-ARG1
001780         GO TO PROG-END
001790     END-IF
001800     DISPLAY '伝言板の内容: ' DCRPS-CALL-OUTDATA
001810                             IN DCRPS-CALL-ARG3
001820     WHEN '2'
001830     DISPLAY '伝言を入力して下さい =>'
001840     ACCEPT INDATA
001850     IF INDATA = SPACE
001860     THEN
001870         MOVE '伝言はありません。' TO INDATA
001880     END-IF
001890 *
001900 * *****
001910 * * RPC-CALL(RPCの実行) *
001920 * *****
001930     MOVE 'CALL ' TO
001940         DCRPS-CALL-REQUEST   IN DCRPS-CALL-ARG1
001950     MOVE ZERO           TO
001960         DCRPS-CALL-FLAGS     IN DCRPS-CALL-ARG1
001970     MOVE ' spp01'      TO
001980         DCRPS-CALL-SVGROUP   IN DCRPS-CALL-ARG1
001990     MOVE 'put'         TO
002000         DCRPS-CALL-SVNAME    IN DCRPS-CALL-ARG1
002010     MOVE DCCLS-CLTIN-CLTID IN DCCLS-CLTIN-ARG TO
002020         DCRPS-CALL-CLTID    IN DCRPS-CALL-ARG1
002030     MOVE INDATA        TO
002040         DCRPS-CALL-INDATA    IN DCRPS-CALL-ARG2
002050     MOVE 512           TO
002060         DCRPS-CALL-INDATALEN IN DCRPS-CALL-ARG2
002070     MOVE SPACE        TO
002080         DCRPS-CALL-OUTDATA   IN DCRPS-CALL-ARG3
002090     MOVE 512           TO
002100         DCRPS-CALL-OUTDATALEN IN DCRPS-CALL-ARG3
002110
002120 *
002130 * *****
002140     CALL 'CBLDCRPS' USING DCRPS-CALL-ARG1
002150                             DCRPS-CALL-ARG2 DCRPS-CALL-ARG3
002160 * *****
002170     IF DCRPS-CALL-STATUS-CODE IN DCRPS-CALL-ARG1
002180                             NOT = '00000'

```

```

002190      THEN
002200          DISPLAY 'CUP01: CBLDCRPS(CALL)に失敗しました。CODE='
002210              DCRPS-CALL-STATUS-CODE IN DCRPS-CALL-ARG1
002220      GO TO PROG-END
002230      END-IF
002240      DISPLAY DCRPS-CALL-OUTDATA IN DCRPS-CALL-ARG3
002250      WHEN '9'
002260          GO TO PROG-END
002270      WHEN OTHER
002280          CONTINUE
002290      END-EVALUATE
002300      END-PERFORM.
002310      PROG-END.
002320      *
002330      *****
002340      * RPC-CLOSE(RPC環境の解除) *
002350      *****
002360          MOVE 'CLOSE' TO
002370              DCRPS-CLOSE-REQUEST IN DCRPS-CLOSE-ARG1.
002380          MOVE ZERO TO
002390              DCRPS-CLOSE-FLAGS IN DCRPS-CLOSE-ARG1.
002400          MOVE DCCLS-CLTIN-CLTID IN DCCLS-CLTIN-ARG TO
002410              DCRPS-CLOSE-CLTID IN DCRPS-CLOSE-ARG1.
002420      *
002430      * *****
002440          CALL 'CBLDCRPS' USING DCRPS-CLOSE-ARG1.
002450      * *****
002460      PROG-EXIT.
002470          MOVE 'CLTOUT' TO
002480              DCCLS-CLTOUT-REQUEST IN DCCLS-CLTOUT-ARG.
002490          MOVE ZERO TO
002500              DCCLS-CLTOUT-FLAGS IN DCCLS-CLTOUT-ARG.
002510          MOVE DCCLS-CLTIN-CLTID IN DCCLS-CLTIN-ARG TO
002520              DCCLS-CLTOUT-CLTID IN DCCLS-CLTOUT-ARG.
002530      *
002540      * *****
002550          CALL 'CBLDCCLS' USING DCCLS-CLTOUT-ARG.
002560      * *****
002570          STOP RUN.
002580      *
002590      MAIN-EXIT SECTION.
002600          EXIT.

```

5.4.2 マルチスレッド対応のユーザアプリケーションプログラムの作成

マルチスレッドで動作する UAP を COBOL 言語で作成する方法について、説明します。

(1) 翻訳

COBOL 言語で作成した UAP のソースプログラムの翻訳について、説明します。この UAP のソースプログラムには、スレッド起動プログラム、CUP の主プログラムなどが必要です。

スレッド起動プログラムは C 言語で記述し、C コンパイラで翻訳してオブジェクトファイルを作成します。また、CUP の主プログラムは COBOL 言語で作成し、COBOL コンパイラで翻訳してオブジェクトファイルを作成します。

それぞれのプログラムを翻訳するときのコマンド入力例を次に示します。なお、COBOL コンパイラには、COBOL2002 を使用した場合の例を示します。

〈例〉

C 言語で作成したスレッド起動プログラム

- thdcup_main.c

COBOL 言語で作成した CUP の主プログラム

- sample.cbl

この場合、それぞれのソースプログラムは次のように翻訳します。

```
cc -c -I/usr/include -ansi thdcup_main.c
ccbl2002 -MultiThread -Compile,NoLink sample.cbl
```

注

翻訳環境のロケールが UTF-8 の場合、ccbl2002 のオプションに -UniObjGen も追加してください。上記の cc コマンドおよび ccbl2002 コマンドを実行すると、次のオブジェクトファイルが作成されます。

- thdcup_main.o (スレッド起動プログラムのオブジェクトファイル)
- sample.o (CUP の主プログラムのオブジェクトファイル)

(2) 結合

COBOL 言語で作成した UAP の実行形式ファイルの作成には、ccbl2002 コマンドまたは cc コマンドを使用する方法があります。それぞれの場合の結合方法について、説明します。

(a) ccbl2002 コマンドを使用する場合

ccbl2002 コマンドを使用して UAP の実行形式ファイルを作成する場合、次に示すファイルを結合させて作成します。

- スレッド起動プログラムのオブジェクトファイル
- CUP の主プログラムのオブジェクトファイル
- TP1/Client/W のライブラリ
- COBOL2002 のライブラリ
- POSIX スレッドのライブラリ

ccbl2002 コマンドを使用して、上記のファイルを結合するときのコマンドの入力例を次に示します。

〈例〉

COBOL CUP 実行形式ファイル「example」を作成する場合

- スレッド起動プログラムのオブジェクトファイル名…thdcup_main.o
- CUP の主プログラムのオブジェクトファイル名…sample.o

次のようにファイルを結合させます。

```
ccbl2002 sample.o thdcup_main.o -L/usr/lib -lclt -lpthread -L/opt/HILNGcbl2k/lib -lcbl2kml -lcbl2kmp -OutputFile example
```

(b) cc コマンドを使用する場合

cc コマンドを使用して UAP の実行形式ファイルを作成する場合、次に示すファイルを結合させて作成します。

- スレッド起動プログラムのオブジェクトファイル
- CUP の主プログラムのオブジェクトファイル
- TP1/Client/W のライブラリ
- COBOL2002 のライブラリ
- POSIX スレッドのライブラリ

cc コマンドを使用して、上記のファイルを結合するときのコマンドの入力例を次に示します。

〈例〉

COBOL CUP 実行形式ファイル「example」を作成する場合

- スレッド起動プログラムのオブジェクトファイル名…thdcup_main.o
- CUP の主プログラムのオブジェクトファイル名…sample.o

次のようにファイルを結合させます。

```
cc -o example thdcup_main.o sample.o -L/usr/lib -lclt -lpthread -L/opt/HILNGcbl2k/lib -lcbl2k -lcbl2kml -lcbl2kmp
```

(3) スレッド起動プログラムおよび CUP の主プログラムのコーディング例

COBOL 言語で作成した UAP のスレッド起動プログラム (C 言語)、および CUP の主プログラム (COBOL 言語) のコーディング例を示します。

(a) スレッド起動プログラム (C 言語) のコーディング例

```
000010 #include <stdio.h>
000020 #include <pthread.h>
000030 #include <sys/errno.h>
000040
000050 #define THDMAX 5
000060
```

```

000070 extern void *CUP_THREAD();
000080
000090 main()
000100 {
000110     int        i;
000120     int        rc;
000130     int        *exit_value;
000140     pthread_t  threads[THDMAX];
000150
000160     /*--- スレッドを生成する ---*/
000170     for (i = 1; i < THDMAX; i++) {
000180         fflush(stdout);
000190         rc = pthread_create((pthread_t *)&threads[i],
000200                             NULL,
000210                             CUP_THREAD,
000220                             (void *)&i);
000230         if (rc < 0) {
000240             printf("cup0: pthread_create に失敗しました。CODE=%d\n", errno);
000250         }
000260     }
000270
000280     /*--- スレッドの終了を待ち合わせる ---*/
000290     for (i = 1; i < THDMAX; i++) {
000300         rc = pthread_join(threads[i], (void *)&exit_value);
000310         if (rc < 0) {
000320             printf("cup0: pthread_join に失敗しました。CODE=%d\n", errno);
000330         }
000340     }
000350 }
000360 }
000370

```

(b) CUPの主プログラム (COBOL 言語) のコーディング例

```

000010 *
000020 *****
000030 *   CUPサンプルプログラム                               *
000040 *****
000050 *
000060 IDENTIFICATION DIVISION.
000070 PROGRAM-ID. CUP_THREAD.
000080 *
000090 *****
000100 *   データ領域の設定                                   *
000110 *****
000120 *
000130 DATA DIVISION.
000140 WORKING-STORAGE SECTION.
000150 01 DCCLS-CLTIN-ARG.
000160     02 DCCLS-CLTIN-REQUEST          PIC X(8) VALUE 'CLTIN  '.
000170     02 DCCLS-CLTIN-STATUS-CODE     PIC X(5).
000180     02 FILLER                      PIC X(3).
000190     02 DCCLS-CLTIN-FLAGS           PIC S9(9) COMP VALUE ZERO.
000200     02 DCCLS-CLTIN-T-HOST          PIC X(64).
000210     02 DCCLS-CLTIN-LOGNAME         PIC X(16).
000220     02 DCCLS-CLTIN-PASSWD          PIC X(16).
000230     02 DCCLS-CLTIN-S-HOST          PIC X(64).

```

000240	02	DCCLS-CLTIN-HWND	PIC 9(4) COMP.
000250	02	FILLER	PIC X(2).
000260	02	DCCLS-CLTIN-CLTID	PIC 9(18) COMP.
000270	02	DCCLS-CLTIN-DEFPATH	PIC X(256).
000280	*		
000290	01	DCCLS-CLTOUT-ARG.	
000300	02	DCCLS-CLTOUT-REQUEST	PIC X(8) VALUE 'CLTOUT '.
000310	02	DCCLS-CLTOUT-STATUS-CODE	PIC X(5).
000320	02	FILLER	PIC X(3).
000330	02	DCCLS-CLTOUT-FLAGS	PIC S9(9) COMP VALUE ZERO.
000340	02	FILLER	PIC X(4).
000350	02	DCCLS-CLTOUT-CLTID	PIC 9(18) COMP.
000360	*		
000370	01	DCRPS-OPEN-ARG1.	
000380	02	DCRPS-OPEN-REQUEST	PIC X(8) VALUE 'OPEN '.
000390	02	DCRPS-OPEN-STATUS-CODE	PIC X(5).
000400	02	FILLER	PIC X(3).
000410	02	DCRPS-OPEN-FLAGS	PIC S9(9) COMP VALUE ZERO.
000420	02	FILLER	PIC X(4).
000430	02	DCRPS-OPEN-CLTID	PIC 9(18) COMP.
000440	*		
000450	01	DCRPS-OPEN-ARG2.	
000460	02	FILLER	PIC X(1).
000470	*		
000480	01	DCRPS-OPEN-ARG3.	
000490	02	FILLER	PIC X(1).
000500	*		
000510	01	DCRPS-CALL-ARG1.	
000520	02	DCRPS-CALL-REQUEST	PIC X(8) VALUE 'CALL '.
000530	02	DCRPS-CALL-STATUS-CODE	PIC X(5).
000540	02	FILLER	PIC X(3).
000550	02	DCRPS-CALL-FLAGS	PIC S9(9) COMP VALUE ZERO.
000560	02	DCRPS-CALL-DESCRIPTOR	PIC S9(9) COMP.
000570	02	DCRPS-CALL-SVGROUP	PIC X(32).
000580	02	DCRPS-CALL-SVNAME	PIC X(32).
000590	02	DCRPS-CALL-CLTID	PIC 9(18) COMP.
000600	*		
000610	01	DCRPS-CALL-ARG2.	
000620	02	DCRPS-CALL-INDATALEN	PIC S9(9) COMP.
000630	02	DCRPS-CALL-INDATA	PIC X(512).
000640	*		
000650	01	DCRPS-CALL-ARG3.	
000660	02	DCRPS-CALL-OUTDATALEN	PIC S9(9) COMP.
000670	02	DCRPS-CALL-OUTDATA	PIC X(512).
000680	*		
000690	01	DCRPS-CLOSE-ARG1.	
000700	02	DCRPS-CLOSE-REQUEST	PIC X(8) VALUE 'CLOSE '.
000710	02	DCRPS-CLOSE-STATUS-CODE	PIC X(5).
000720	02	FILLER	PIC X(3).
000730	02	DCRPS-CLOSE-FLAGS	PIC S9(9) COMP VALUE ZERO.
000740	02	FILLER	PIC X(4).
000750	02	DCRPS-CLOSE-CLTID	PIC 9(18) COMP.
000760	*		
000770	01	DCRPS-CLOSE-ARG2.	
000780	02	FILLER	PIC X(1).
000790	*		
000800	01	DCRPS-CLOSE-ARG3.	
000810	02	FILLER	PIC X(1).

```

000820 *
000830 *****
000840 * CUPの開始 *
000850 *****
000860 PROCEDURE DIVISION.
000870 MAIN SECTION.
000880 PROG-START.
000890 *
000900 *****
000910 * クライアントユーザの認証要求 *
000920 *****
000930 MOVE 'CLTIN ' TO DCCLS-CLTIN-REQUEST IN DCCLS-CLTIN-ARG.
000940 MOVE ZERO TO DCCLS-CLTIN-FLAGS IN DCCLS-CLTIN-ARG.
000950 MOVE 'host01:10000' TO DCCLS-CLTIN-T-HOST
000960 IN DCCLS-CLTIN-ARG.
000970 MOVE 'user01' TO DCCLS-CLTIN-LOGNAME IN DCCLS-CLTIN-ARG.
000980 MOVE 'puser01' TO DCCLS-CLTIN-PASSWD IN DCCLS-CLTIN-ARG.
000990 MOVE ZERO TO DCCLS-CLTIN-HWND IN DCCLS-CLTIN-ARG.
001000 MOVE SPACE TO DCCLS-CLTIN-DEFPATH IN DCCLS-CLTIN-ARG.
001010 *
001020 * *****
001030 CALL 'CBLDCCLS' USING DCCLS-CLTIN-ARG.
001040 * *****
001050 IF DCCLS-CLTIN-STATUS-CODE IN DCCLS-CLTIN-ARG NOT = '00000'
001060 THEN
001070 DISPLAY 'CUP01: CBLDCCLS(CLTIN)に失敗しました。CODE='
001080 DCCLS-CLTIN-STATUS-CODE IN DCCLS-CLTIN-ARG
001090 GO TO PROG-EXIT
001100 END-IF.
001110 *
001120 *****
001130 * RPC-OPEN(RPC環境の初期設定) *
001140 *****
001150 MOVE 'OPEN ' TO
001160 DCRPS-OPEN-REQUEST IN DCRPS-OPEN-ARG1.
001170 MOVE ZERO TO DCRPS-OPEN-FLAGS
001180 IN DCRPS-OPEN-ARG1.
001190 MOVE DCCLS-CLTIN-CLTID IN DCCLS-CLTIN-ARG TO
001200 DCRPS-OPEN-CLTID IN DCRPS-OPEN-ARG1.
001210 *
001220 * *****
001230 CALL 'CBLDCRPS' USING DCRPS-OPEN-ARG1 DCRPS-OPEN-ARG2
001240 DCRPS-OPEN-ARG3.
001250 * *****
001260 IF DCRPS-OPEN-STATUS-CODE IN DCRPS-OPEN-ARG1 NOT = '00000'
001270 THEN
001280 DISPLAY 'CUP01: CBLDCRPS(OPEN)に失敗しました。CODE='
001290 DCRPS-OPEN-STATUS-CODE IN DCRPS-OPEN-ARG1
001300 GO TO PROG-END
001310 END-IF.
001320 *
001330 * *****
001340 * * RPC-CALL(RPCの実行) *
001350 * *****
001360 MOVE 'CALL ' TO
001370 DCRPS-CALL-REQUEST IN DCRPS-CALL-ARG1.
001380 MOVE ZERO TO
001390 DCRPS-CALL-FLAGS IN DCRPS-CALL-ARG1.

```



```

001400     MOVE ' spp01'          TO
001410     DCRPS-CALL-SVGROUP IN DCRPS-CALL-ARG1.
001420     MOVE ' svr01'          TO
001430     DCRPS-CALL-SVNAME      IN DCRPS-CALL-ARG1.
001440     MOVE DCCLS-CLTIN-CLTID IN DCCLS-CLTIN-ARG TO
001450     DCRPS-CALL-CLTID      IN DCRPS-CALL-ARG1.
001460     MOVE 'HELLO SPP !!' TO
001470     DCRPS-CALL-INDATA     IN DCRPS-CALL-ARG2.
001480     MOVE 512              TO
001490     DCRPS-CALL-INDATALEN IN DCRPS-CALL-ARG2.
001500     MOVE SPACE            TO
001510     DCRPS-CALL-OUTDATA     IN DCRPS-CALL-ARG3.
001520     MOVE 512              TO
001530     DCRPS-CALL-OUTDATALEN IN DCRPS-CALL-ARG3.
001540     *
001550     * *****
001560     CALL 'CBLDCRPS' USING DCRPS-CALL-ARG1 DCRPS-CALL-ARG2
001570     DCRPS-CALL-ARG3
001580     * *****
001590     IF DCRPS-CALL-STATUS-CODE IN DCRPS-CALL-ARG1 NOT = '00000'
001600     THEN
001610         DISPLAY 'CUP01: CBLDCRPS(CALL)に失敗しました。'
001620         'CODE=' DCRPS-CALL-STATUS-CODE IN DCRPS-CALL-ARG1
001630         GO TO PROG-END
001640     END-IF.
001650     PROG-END.
001660     *
001670     * *****
001680     * RPC-CLOSE(RPC環境の解除) *
001690     * *****
001700     MOVE 'CLOSE' TO DCRPS-CLOSE-REQUEST IN DCRPS-CLOSE-ARG1.
001710     MOVE ZERO TO DCRPS-CLOSE-FLAGS IN DCRPS-CLOSE-ARG1.
001720     MOVE DCCLS-CLTIN-CLTID IN DCCLS-CLTIN-ARG TO
001730     DCRPS-CLOSE-CLTID IN DCRPS-CLOSE-ARG1.
001740     *
001750     * *****
001760     CALL 'CBLDCRPS' USING DCRPS-CLOSE-ARG1 DCRPS-CLOSE-ARG2
001770     DCRPS-CLOSE-ARG3.
001780     * *****
001790     PROG-EXIT.
001800     MOVE 'CLTOUT' TO DCCLS-CLTOUT-REQUEST IN DCCLS-CLTOUT-ARG.
001810     MOVE ZERO TO DCCLS-CLTOUT-FLAGS IN DCCLS-CLTOUT-ARG.
001820     MOVE DCCLS-CLTIN-CLTID IN DCCLS-CLTIN-ARG TO
001830     DCCLS-CLTOUT-CLTID IN DCCLS-CLTOUT-ARG.
001840     *
001850     * *****
001860     CALL 'CBLDCCLS' USING DCCLS-CLTOUT-ARG.
001870     * *****
001880     STOP RUN.
001890     *

```

6

TP1/Client で使用できる要求文 (COBOL 言語編)

TP1/Client で使用できる要求文について説明します。

この章では、各要求文を、マルチスレッド環境の要求文 (CBLDCCLS(' ')など) で説明します。シングルスレッド環境の要求文を使う場合は、シングルスレッド環境の要求文 (CBLDCCLT(' ')など) に置き換えて読んでください。

6.1 要求文を使用するときの注意事項

要求文を使用するときは、シングルスレッド環境の場合も、マルチスレッド環境を呼び出すときに使用する要求文を、ご使用になられることをお勧めします。

マルチスレッド環境の場合は、シングルスレッド環境を呼び出すときに使用する要求文を、使用しないでください。

6.2 ユーザ認証機能

6.2.1 CBLDCCLS('CLTIN ') – クライアントユーザの認証要求

(1) 形式

(a) マルチスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCCLS' USING 一意名1
```

■ DATA DIVISION の指定

```
01 一意名1.  
02 データ名A PIC X(8) VALUE 'CLTIN ' .  
02 データ名B PIC X(5).  
02 FILLER PIC X(3).  
02 データ名C PIC S9(9) COMP VALUE ZERO.  
02 データ名D PIC X(64).  
02 データ名E PIC X(16).  
02 データ名F PIC X(16).  
02 データ名G PIC X(64).  
02 データ名H PIC 9(4) COMP.  
02 FILLER PIC X(2).  
02 データ名I PIC 9(9) COMP-X.  
02 データ名J PIC X(256).
```

(b) シングルスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCCLT' USING 一意名1
```

■ DATA DIVISION の指定

```
01 一意名1.  
02 データ名A PIC X(8) VALUE 'CLTIN ' .  
02 データ名B PIC X(5).  
02 FILLER PIC X(3).  
02 データ名C PIC S9(9) COMP VALUE ZERO.  
02 データ名D PIC X(64).  
02 データ名E PIC X(16).  
02 データ名F PIC X(16).  
02 データ名G PIC X(64).
```

(2) 機能

指定された窓口となる TP1/Server に対して、ログイン名で指定されたクライアントユーザの認証を要求します。

ユーザ認証を抑止する場合でも、CBLDCCLS('CLTIN ')は必ず実行してください。

(3) UAP で値を設定するデータ領域

- データ名 A

クライアントユーザの認証要求を示す要求コードを「VALUE 'CLTIN△△△」と設定します。

- データ名 C

リモート API 機能を使用するためにユーザ認証を抑止する場合は、-2147483648 を設定します。ユーザ認証を抑止しない場合は、0 を設定します。

- データ名 D

認証要求を実行する時に窓口となる TP1/Server のホスト名およびポート番号を設定します。複数の窓口となる TP1/Server を設定できます（区切り文字は','を使用）。また、ホスト名として、10 進ドット記法の IP アドレスを設定することもできます。

形式

ホスト名 [:ポート番号] [,ホスト名 [:ポート番号] ,...]

- ホスト名～<文字列>
- ポート番号～<符号なし整数>((5001～65535))

区切り文字','の後ろ以外は空白文字（スペースまたはタブ）を入れないでください。

ポート番号省略時は、クライアント環境定義 DCNAMPORT の値を仮定します。

データ名 D に TP1/Server を二つ以上指定し、窓口となる TP1/Server の障害を検出した場合、クライアント環境定義 DCHOSTSELECT に N を指定したとき、現在窓口となっている TP1/Server の次の TP1/Server を参照して切り替えを試みます。クライアント環境定義 DCHOSTSELECT に Y を指定したとき、障害を検出した TP1/Server を除いて、窓口となる TP1/Server をランダムに選択して切り替えを試みます。

先頭に空白が指定されていた場合は、クライアント環境定義 DCHOST を参照します。

ただし、データ名 D の先頭に空白が指定されていて、DCHOST が設定されていなかった場合は、ブロードキャストを行い、窓口となる TP1/Server を決定します。

TP1/Client/P でブロードキャストを行う場合、hosts ファイルにブロードキャストアドレスを指定する必要があります（ホスト名は broadcast としてください）。指定がなかった場合、CBLDCCLS('EXCLTIN ')は 02518 でエラーリターンします。

文字列の最後は空白を設定してください。

- データ名 E

クライアントユーザのログイン名を設定します。

文字列の最後は空白文字を設定してください。

- データ名 F

データ名 E で指定したログイン名に対するパスワードを設定します。パスワードが設定されていない場合は、先頭に空白を指定します。

文字列の最後は空白文字を設定してください。

- データ名 H

0 を設定します。

- データ名 J

クライアント環境定義ファイルへのパス名を指定します。パス名には完全パス、またはカレントドライブ・ディレクトリからの相対パスが指定できます。パス名を指定した場合のファイルの読み込み順序を次に示します。

- TP1/Client/P の場合

クライアント環境定義ファイルの読み込み順序は次のとおりです。

- 1.Windows ディレクトリの BETRAN.INI ファイル
- 2.データ名 J に指定したクライアント環境定義ファイル

定義は、クライアント環境定義ファイルおよび BETRAN.INI ファイルのどちらのファイルに指定しても有効です。

両方のファイルに同じ定義を異なる値で指定した場合は、クライアント環境定義ファイルに指定した値が有効となります。

クライアント環境定義ファイルおよび BETRAN.INI ファイルのどちらにも指定がない場合は、デフォルト値で動作します。

- TP1/Client/W の場合

環境変数に指定されている定義は、すべて無効となります。データ名 J に指定したクライアント環境定義ファイルに指定されていない定義はデフォルト値で動作します。

また、データ名 J の先頭に空白を指定することでパス名を省略できます。省略時の動作を次に示します。

- TP1/Client/P の場合

Windows ディレクトリの BETRAN.INI ファイルをクライアント環境定義ファイルとして動作します。BETRAN.INI ファイルがない場合、または定義ファイルの内容が不正な場合はデフォルト値で動作します。

- TP1/Client/W の場合

環境変数の指定で動作します。環境変数が指定されていない場合は、デフォルト値で動作します。

データ名 J に指定したクライアント環境定義ファイルがない場合、または定義ファイルの内容が不正な場合の動作を次に示します。

- TP1/Client/P の場合

Windows ディレクトリの BETRAN.INI ファイルをクライアント環境定義ファイルとして動作します。BETRAN.INI ファイルがない場合、または定義ファイルの内容が不正な場合は、デフォルト値で動作します。

- TP1/Client/W の場合

デフォルト値で動作します。環境変数の指定は無効となります。

(4) 値が返されるデータ領域

- データ名 B

ステータスコードが、5 けたの数字で返されます。

- データ名 G

ユーザ認証したサーバのホスト名（または 10 進ドット記法の IP アドレス）が返されます。ユーザ認証を抑止した場合は、返されません。

格納されたホスト名の最後には、空白が設定されます。

- データ名 I

クライアントユーザの認証が正常に終了すると、クライアント ID が設定されます。設定されたクライアント ID は、CBLDCCLS('CLTOUT') を実行するまで破壊してはなりません。

(5) ステータスコード

ステータスコード	意味
00000	正常終了しました。
02501	データ名に指定した値が間違っています。要求コード（データ名 A）が間違っている場合も含まれます。
02502	すでに CBLDCCLT('CLTIN') が実行されています。CBLDCCLS('CLTIN') 実行時、このステータスコードは返りません。
02503	通信路の初期化に失敗しました。または、クライアント環境定義の指定が誤っています。
02504	必要なバッファが確保できませんでした。
02506	ネットワーク障害が発生しました。
02515	設定したサービスがあるノードの OpenTP1 が起動されていません。
02518	システムエラーが発生しました。
02527	次のどれかの要因で、このステータスコードが戻りました。 <ul style="list-style-type: none">• 指定されたデータ名 E が対象とするホストに登録されていません。• パスワードが一致しません。• ユーザ認証機能をサポートしていない OpenTP1 サーバの可能性があります。システム共通定義 client_uid_check の指定が正しくありません。
02547	指定したポート番号は使用されています。または、OS が自動的に割り当てるポート番号が不足しています。

6.2.2 CBLDCCLS('EXCLTIN ') – クライアントユーザの認証要求 (ホスト名長の拡張時)

(1) 形式

(a) マルチスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCCLS' USING 一意名1 一意名2 一意名3
```

■ DATA DIVISION の指定

```
01 一意名1.  
02 データ名A PIC X(8) VALUE 'EXCLTIN'.  
02 データ名B PIC X(5).  
02 FILLER PIC X(3).  
02 データ名C PIC S9(9) COMP VALUE ZERO.  
02 データ名D PIC X(16).  
02 データ名E PIC X(16).  
02 データ名F PIC X(n).  
01 一意名2.  
02 FILLER PIC X(9) COMP.  
02 データ名G PIC X(n).  
01 一意名3.  
02 データ名H PIC 9(9) COMP-X.  
02 データ名I PIC 9(4) COMP.  
02 FILLER PIC X(2).  
02 データ名J PIC X(n).
```

(b) シングルスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCCLT' USING 一意名1 一意名2
```

■ DATA DIVISION の指定

```
01 一意名1.  
02 データ名A PIC X(8) VALUE 'EXCLTIN'.  
02 データ名B PIC X(5).  
02 FILLER PIC X(3).  
02 データ名C PIC S9(9) COMP VALUE ZERO.  
02 データ名D PIC X(16).  
02 データ名E PIC X(16).  
02 データ名F PIC X(n).  
01 一意名2.  
02 FILLER PIC X(9) COMP.  
02 データ名G PIC X(n).
```


(2) 機能

指定された窓口となる TP1/Server に対して、ログイン名で指定されたクライアントユーザの認証を要求します。

ユーザ認証を抑止する場合でも、CBLDCCLS('EXCLTIN ')は必ず実行してください。

ホスト名長の拡張機能を使用している場合、この関数を使用してください。

(3) UAP で値を設定するデータ領域

• データ名 A

クライアントユーザの認証要求を示す要求コードを「VALUE 'EXCLTIN△」と設定します。

• データ名 C

リモート API 機能を使用するためにユーザ認証を抑止する場合は、-2147483648 を設定します。ユーザ認証を抑止しない場合は、0 を設定します。

• データ名 D

クライアントユーザのログイン名を設定します。

文字列の最後は空白文字を設定してください。

• データ名 E

データ名 D で指定したログイン名に対するパスワードを設定します。パスワードが設定されていない場合は、先頭に空白を指定します。

文字列の最後は空白文字を設定してください。

• データ名 F

認証要求を実行するとき窓口となる TP1/Server のホスト名およびポート番号を設定します。複数の窓口となる TP1/Server を設定できます（区切り文字は','を使用）。また、ホスト名として、10 進ドット記法の IP アドレスを設定することもできます。

形式

ホスト名 [:ポート番号] [,ホスト名 [:ポート番号] ,...]

・ホスト名～<文字列>

・ポート番号～<符号なし整数>((5001～65535))

ホスト名として指定できる長さは、63 文字*までです。ホスト名を複数指定する場合、データ名 F に指定できる長さ（ポート番号なども含む）は、255 文字*までです。文字列の最後は空白文字を設定してください。

区切り文字','の後ろ以外は空白文字（スペースまたはタブ）を入れないでください。ポート番号省略時は、クライアント環境定義 DCNAMPORT の値を仮定します。

データ名 F に TP1/Server を二つ以上指定し、窓口となる TP1/Server の障害を検出した場合、クライアント環境定義 DCHOSTSELECT に N を指定したとき、現在窓口となっている TP1/Server の次の TP1/Server を参照して切り替えを試みます。クライアント環境定義 DCHOSTSELECT に Y を指

定したとき、障害を検出した TP1/Server を除いて、窓口となる TP1/Server をランダムに選択して切り替えを試みます。

先頭に空白が指定されていた場合は、クライアント環境定義 DCHOST を参照します。

ただし、データ名 F の先頭に空白が指定されていて、DCHOST が設定されていなかった場合は、ブロードキャストを行い、窓口となる TP1/Server を決定します。

TP1/Client/P でブロードキャストを行う場合、hosts ファイルにブロードキャストアドレスを指定する必要があります（ホスト名は broadcast としてください）。指定がなかった場合、CBLDCCLS('EXCLTIN ')は 02518 でエラーリターンします。

文字列の最後は空白を設定してください。

注※

クライアント環境定義 DCCLTOPTION に 00000008 を指定した場合、ホスト名として指定できる長さは、255 文字までです。ホスト名を複数指定する場合、データ名 F に指定できる長さ（ポート番号なども含む）は 1023 文字までとなります。

• データ名 G

ユーザ認証したサーバのホスト名を格納する 64 バイト以上※の領域を用意してください。

注※

クライアント環境定義 DCCLTOPTION に 00000008 を指定した場合は、256 バイト以上の領域を用意してください。

• データ名 I

0 を設定します。

• データ名 J

クライアント環境定義ファイルへのパス名を指定します。パス名には完全パス、またはカレントドライブ・ディレクトリからの相対パスが指定できます。パス名を指定した場合のファイルの読み込み順序を次に示します。

• TP1/Client/P の場合

クライアント環境定義ファイルの読み込み順序は次のとおりです。

- 1.Windows ディレクトリの BETRAN.INI ファイル
- 2.データ名 J に指定したクライアント環境定義ファイル

定義は、クライアント環境定義ファイルおよび BETRAN.INI ファイルのどちらのファイルに指定しても有効です。

両方のファイルに同じ定義を異なる値で指定した場合は、クライアント環境定義ファイルに指定した値が有効となります。

クライアント環境定義ファイルおよび BETRAN.INI ファイルのどちらにも指定がない場合は、デフォルト値で動作します。

• TP1/Client/W の場合

環境変数に指定されている定義は、すべて無効となります。データ名 J に指定したクライアント環境定義ファイルに指定されていない定義はデフォルト値で動作します。

また、データ名 J の先頭に空白を指定することでパス名を省略できます。省略時の動作を次に示します。

- TP1/Client/P の場合

Windows ディレクトリの BETRAN.INI ファイルをクライアント環境定義ファイルとして動作します。BETRAN.INI ファイルがない場合、または定義ファイルの内容が不正な場合はデフォルト値で動作します。

- TP1/Client/W の場合

環境変数の指定で動作します。環境変数が指定されていない場合は、デフォルト値で動作します。

データ名 J に指定したクライアント環境定義ファイルがない場合、または定義ファイルの内容が不正な場合の動作を次に示します。

- TP1/Client/P の場合

Windows ディレクトリの BETRAN.INI ファイルをクライアント環境定義ファイルとして動作します。BETRAN.INI ファイルがない場合、または定義ファイルの内容が不正な場合は、デフォルト値で動作します。

- TP1/Client/W の場合

デフォルト値で動作します。環境変数の指定は無効となります。

(4) 値が返されるデータ領域

- データ名 B

ステータスコードが、5 けたの数字で返されます。

- データ名 G

ユーザ認証したサーバのホスト名（または 10 進ドット記法の IP アドレス）が返されます。ユーザ認証を抑制した場合は、返されません。

格納されたホスト名の最後には、空白が設定されます。

- データ名 H

クライアントユーザの認証が正常に終了すると、クライアント ID が設定されます。設定されたクライアント ID は、CBLDCCLS('CLTOUT ')を実行するまで破壊してはなりません。

(5) ステータスコード

ステータスコード	意味
00000	正常終了しました。
02501	データ名に指定した値が間違っています。要求コード（データ名 A）が間違っている場合も含まれます。
02502	すでに CBLDCCLT('EXCLTIN ')が実行されています。 CBLDCCLS('EXCLTIN ')実行時、このステータスコードは返りません。
02503	通信路の初期化に失敗しました。または、クライアント環境定義の指定が誤っています。

ステータスコード	意味
02504	必要なバッファが確保できませんでした。
02506	ネットワーク障害が発生しました。
02515	設定したサービスがあるノードの OpenTP1 が起動されていません。
02518	システムエラーが発生しました。
02527	次のどれかの要因で、このステータスコードが戻りました。 <ul style="list-style-type: none"> 指定されたログイン名（データ名 D）が対象とするホストに登録されていません。 パスワード（データ名 E）が一致しません。 ユーザ認証機能をサポートしていない OpenTP1 サーバの可能性あります。 システム共通定義の client_uid_check の指定が正しいか見直してください。
02547	指定したポート番号は使用されています。または、OS が自動的に割り当てるポート番号が不足しています。

6.2.3 CBLDCCLS('CLTOUT') – クライアントユーザの認証解除

(1) 形式

(a) マルチスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCCLS' USING 一意名1
```

■ DATA DIVISION の指定

```
01 一意名1.
02 データ名A PIC X(8) VALUE 'CLTOUT '.
02 データ名B PIC X(5).
02 FILLER PIC X(3).
02 データ名C PIC S9(9) COMP VALUE ZERO.
02 データ名D PIC 9(9) COMP-X.
```

(b) シングルスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCCLT' USING 一意名1
```

■ DATA DIVISION の指定

```
01 一意名1.
02 データ名A PIC X(8) VALUE 'CLTOUT '.
02 データ名B PIC X(5).
```

```
02 FILLER      PIC X(3).
02 データ名C   PIC S9(9) COMP VALUE ZERO.
```

(2) 機能

クライアントユーザの認証を解除し、以降 OpenTP1 のサービスを受けられないようにします。

CBLDCCLS('CLTOUT ')は、CUP を終了する前に必ず実行してください。CBLDCCLS('CLTOUT ')は、CBLDCCLS('CLTIN ')または CBLDCCLS('EXCLTIN ')と対になるように実行する必要があります。

(3) UAP で値を設定するデータ領域

- データ名 A

クライアントユーザの認証解除を示す要求コードを「VALUE 'CLTOUT△△」と設定します。

- データ名 C

0 を設定します。

- データ名 D

CBLDCCLS('CLTIN '), または CBLDCCLS('EXCLTIN ')で受け取ったクライアント ID を指定します。

(4) OpenTP1 から値が返されるデータ領域

- データ名 B

ステータスコードが、5 けたの数字で返されます。

(5) ステータスコード

ステータスコード	意味
00000	正常終了しました。
02501	要求コード（データ名 A）に指定した値が間違っています。

(6) 注意事項

クライアント ID（データ名 D）に不正な値を指定した場合、CUP が異常終了または予期しない動作をするおそれがあります。

6.3 リモートプロシジャコール

6.3.1 CBLDCRPS('OPEN ') – UAP の開始

(1) 形式

(a) マルチスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCRPS' USING 一意名1 一意名2 一意名3
```

■ DATA DIVISION の指定

```
01 一意名1.  
  02 データ名A PIC X(8) VALUE 'OPEN  ' .  
  02 データ名B PIC X(5).  
  02 FILLER PIC X(3).  
  02 データ名C PIC S9(9) COMP VALUE ZERO.  
  02 データ名D PIC 9(9) COMP-X.  
  
01 一意名2.  
  02 FILLER PIC X(1).  
  
01 一意名3.  
  02 FILLER PIC X(1).
```

(b) シングルスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCRPC' USING 一意名1
```

■ DATA DIVISION の指定

```
01 一意名1.  
  02 データ名A PIC X(8) VALUE 'OPEN  ' .  
  02 データ名B PIC X(5).  
  02 FILLER PIC X(3).  
  02 データ名C PIC S9(9) COMP VALUE ZERO.
```

(2) 機能

OpenTP1 の SPP を呼び出すための環境、または TCP/IP 通信機能を使用するための環境を初期化します。

CBLDCRPS('OPEN ')は、RPC、トランザクション制御、トランザクション制御の各種プログラムを発行する前に実行してください。

(3) UAP で値を設定するデータ領域

- データ名 A

UAP の開始を示す要求コードを「VALUE 'OPEN△△△△」と設定します。

- データ名 C

初期化する環境を指定します。

0：SPP を呼び出すための環境

4：メッセージを一方送信するための環境

8：メッセージを一方受信するための環境

16：メッセージを送受信するための環境

0 を指定した場合は、TCP/IP 通信機能を使用できません。

0 以外を指定した場合でも、RPC の機能は使用できます。

- データ名 D

CBLDCCLS('CLTIN '), または CBLDCCLS('EXCLTIN ') で受け取ったクライアント ID を指定します。

(4) 値が返されるデータ領域

- データ名 B

ステータスコードが、5 けたの数字で返されます。

(5) ステータスコード

ステータスコード	意味
00000	正常終了しました。
02401	データ名に設定した値が間違っています。要求コード（データ名 A）が間違っている場合も含まれます。
02402	CBLDCRPS('OPEN ') はすでに実行されています。
02403	次のどれかの要因で、このステータスコードが戻りました。 <ul style="list-style-type: none">• 初期化に失敗しました。• ユーザ認証がされていません。• クライアント環境定義の指定が誤っています。
02415	設定したサービスが存在するノードの OpenTP1 が実行されていません。
02447	指定したポート番号は使用されています。
02544	データ名 D に指定したクライアント ID は CBLDCCLS('CLTIN '), または CBLDCCLS('EXCLTIN ') で受け取ったクライアント ID と異なります。

(6) 注意事項

次に示す場合は、CBLDCRPS('CLOSE ')を実行した直後に、再びデータ名 C に 8 を指定した CBLDCRPS('OPEN ')は実行できません。この場合、15 秒から 20 秒後に CBLDCRPS('OPEN ')を実行してください。

- データ名 C に 8 を指定した CBLDCRPS('OPEN ')実行したあと、CBLDCCLS('RECEIVE ')を実行してメッセージを受信している場合で、相手システムがコネクションを解放する前に CUP 側で CBLDCRPS('CLOSE ')を実行してコネクションを解放した場合

6.3.2 CBLDCRPS('CLOSE ') – UAP の終了

(1) 形式

(a) マルチスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCRPS' USING 一意名1 一意名2 一意名3
```

■ DATA DIVISION の指定

```
01 一意名1.  
02 データ名A PIC X(8) VALUE 'CLOSE ' .  
02 データ名B PIC X(5).  
02 FILLER PIC X(3).  
02 データ名C PIC S9(9) COMP VALUE ZERO.  
02 データ名D PIC 9(9) COMP-X.  
  
01 一意名2.  
02 FILLER PIC X(1).  
  
01 一意名3.  
02 FILLER PIC X(1).
```

(b) シングルスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCRPC' USING 一意名1
```

■ DATA DIVISION の指定

```
01 一意名1.  
02 データ名A PIC X(8) VALUE 'CLOSE ' .  
02 データ名B PIC X(5).
```



```
02 FILLER PIC X(3).
02 データ名C PIC S9(9) COMP VALUE ZERO.
```

(2) 機能

OpenTP1 の SPP を呼び出すための環境、または TCP/IP 通信機能を使用するための環境を解除します。

CBLDCRPS('CLOSE ')は、CBLDCRPS('OPEN ')と対になるように実行する必要があります。

CBLDCRPS('CLOSE ')を実行したあとに発行できるプログラムを次に示します。

- CBLDCRPS('OPEN ')
- CBLDCCLS('CLTOUT ')

(3) UAP で値を設定するデータ領域

- データ名 A

UAP の終了を示す要求コードを「VALUE 'CLOSE△△△」と設定します。

- データ名 C

0 を設定します。

- データ名 D

CBLDCCLS('CLTIN '), または CBLDCCLS('EXCLTIN ')で受け取ったクライアント ID を指定します。

(4) 値が返されるデータ領域

- データ名 B

ステータスコードが、5 けたの数字で返されます。

(5) ステータスコード

ステータスコード	意味
00000	正常に終了しました。
02401	要求コード (データ名 A) に指定した値が間違っています。

6.3.3 CBLDCRPS('CALL') - 遠隔サービスの要求

(1) 形式

(a) マルチスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCRPS' USING 一意名1 一意名2 一意名3
```

■ DATA DIVISION の指定

```
01 一意名1.  
02 データ名A PIC X(8) VALUE 'CALL' .  
02 データ名B PIC X(5).  
02 FILLER PIC X(3).  
02 データ名C PIC S9(9) COMP VALUE ZERO.  
02 データ名D PIC S9(9) COMP.  
02 データ名E PIC X(32).  
02 データ名F PIC X(32).  
02 データ名G PIC 9(9) COMP-X.  
  
01 一意名2.  
02 データ名H PIC S9(9) COMP.  
02 データ名I PIC X(n).  
  
01 一意名3.  
02 データ名J PIC S9(9) COMP.  
02 データ名K PIC X(n).
```

(b) シングルスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCRPC' USING 一意名1 一意名2 一意名3
```

■ DATA DIVISION の指定

```
01 一意名1.  
02 データ名A PIC X(8) VALUE 'CALL' .  
02 データ名B PIC X(5).  
02 FILLER PIC X(3).  
02 データ名C PIC S9(9) COMP VALUE ZERO.  
02 データ名D PIC S9(9) COMP.  
02 データ名E PIC X(32).  
02 データ名F PIC X(32).  
  
01 一意名2.  
02 データ名H PIC S9(9) COMP.  
02 データ名I PIC X(n).  
  
01 一意名3.
```

(2) 機能

SPP のサービスを要求します。「サービスグループ名+サービス名」に該当するサービスプログラムを呼び、その応答を受け取ります。

サービス要求されたサーバ UAP が存在するノードの OpenTP1 は、稼働していなければなりません。OpenTP1 が稼働していない場合（開始処理中を含む）は、CBLDCRPS('CALL ')は、02406, 02415, 02420 でエラーリターンします。

CBLDCRPS('CALL ')を実行したときに、目的のサービスグループが閉塞されている場合は、02412 でエラーリターンします。

CBLDCRPS('CALL ')を実行したときに、目的のサービスグループが dcsvstop コマンドなどで終了処理中、または終了している場合、02413, 02412, 02410 のうちのどれかでエラーリターンします。どのステータスコードが戻るかは、CBLDCRPS('CALL ')を実行したタイミングで決まります。

ソケット受信型サーバでは、ユーザサービス定義の max_socket_msg と max_socket_msglen の指定でデータの輻輳制御をしています。そのため、サービス要求を受信できない場合があります。このとき、CBLDCRPS('CALL ')は、02456 でエラーリターンします。この値が戻った場合、CUP は適当な時間をおいてから再実行すれば、サービス要求できる場合があります。

通常の通信形態の場合、クライアント環境定義 DCCLTSERVICEGROUPLIST に XDM/DCCM3 論理端末のホスト名およびポート番号を指定し、CBLDCRPS('CALL ')を実行します。

(a) サーバ UAP に渡す値

CUP では、サービスプログラムの応答の領域（データ名 K）を確保しておきます。さらに、CUP では CBLDCRPS('CALL ')に次の値を指定します。

- 入力パラメタ（データ名 I）
- 入力パラメタ長（データ名 H）
- 応答の長さ（データ名 J）

入力パラメタ、入力パラメタ長、応答の長さは、CUP の CBLDCRPS('CALL ')で指定した値がそのままサービスプログラムに渡されます。応答を返さないサービスプログラムのサービスを呼ぶときは、応答の長さを指定しても無視されます。入力パラメタ長と応答の長さの最大値は、C 言語のヘッダファイル dcvrpc.h で規定している DCRPC_MAX_MESSAGE_SIZE[※]です。

注※

クライアント環境定義 DCCLTRPCMAXMSGSIZE に 2 以上を指定した場合、DCRPC_MAX_MESSAGE_SIZE の値（1 メガバイト）ではなく、クライアント環境定義 DCCLTRPCMAXMSGSIZE に指定した値になります。

(b) サーバUAPから戻ってくる値

サービスプログラムの処理の終了後に、次の値が参照できます。

- サービスプログラムの応答（データ名 K）
- サービスプログラムの応答の長さ（データ名 J）

データ名 J は、サービスプログラムから実際に返ってきた応答の長さです。同期応答型 RPC（データ名 C に 0 を設定）の場合、CBLDCRPS('CALL ') がリターンしたあと、データ名 K とデータ名 J を参照できます。非応答型 RPC（データ名 C に 1 を設定）の場合、データ名 K とデータ名 J は参照できません。また、CBLDCRPS('CALL ') がエラーリターンした場合もデータ名 K とデータ名 J は参照できません。

返ってきた応答が CUP で確保した応答の領域（データ名 K）よりも大きい場合は、ステータスコード 02409 でエラーリターンします。

(3) UAP で値を設定するデータ領域

- データ名 A

遠隔サービスの要求を示す要求コードを「VALUE 'CALL△△△△」と設定します。

- データ名 C

RPC の形態を設定します。

0：同期応答型 RPC

1：非応答型 RPC

4：連鎖 RPC

データ名 C に 0、または 4 を指定すると、応答が返されてくるか、または応答待ち時間（クライアント環境定義 DCWATCHTIM の値）切れ（タイムアウト）エラーになるまで、CBLDCRPS('CALL ') は戻りません。ただし、サービス要求先の SPP がアボートした場合は、即時にエラーリターンします。この場合、DCWATCHTIM で指定された応答待ち時間によって、次の二つのステータスコードが返されます。

- DCWATCHTIM に 1～65535 を指定した場合：02407
- DCWATCHTIM に 0（無限に待つ）を指定した場合：02414

このため、サービス要求されるサービスプログラムごと、または 1 回のサービス要求ごとの応答待ち時間は指定できません。

データ名 C に 1 を指定すると、要求したサービスは応答を返さないサービスとみなされます。この場合、CBLDCRPS('CALL ') はサービスの実行終了を待たないで、すぐに戻ります。この指定をした場合は、応答（データ名 K）と応答の長さ（データ名 J）は参照できません。さらに、サービスプログラムが発行されたかどうかは、CUP ではわかりません。

トランザクションの処理からの RPC を、トランザクションとしないサービス要求にできます。RPC の形態を示すパラメータに 32 を指定すると、該当する CBLDCRPS('CALL ') のサービス要求は、トランザクションの処理でないサービス要求になります。

32：同期応答型 RPC

33：非応答型 RPC

36：連鎖 RPC

また、トランザクション外、または常設コネクション確立中でないときに 4 を指定するとステータスコード 02401 でエラーリターンします。

- データ名 E

サービスグループ名を 31 バイト以内のアスキー文字列で設定します。文字列の最後は空白文字を設定してください。

- データ名 F

サービス名を 31 バイト以内のアスキー文字列で設定します。文字列の最後は空白文字を設定してください。

- データ名 G

CBLDCCLS('CLTIN '), または CBLDCCLS('EXCLTIN ') で受け取ったクライアント ID を指定します。

- データ名 H

入力パラメタ長（データ名 I の長さ）を設定します。データ名 H 自身の長さは含みません。1 から DCRPC_MAX_MESSAGE_SIZE* までの値が設定できます。

注※

クライアント環境定義 DCCLTRPCMAXMSGSIZE に 2 以上を指定した場合、DCRPC_MAX_MESSAGE_SIZE の値（1 メガバイト）ではなく、クライアント環境定義 DCCLTRPCMAXMSGSIZE に指定した値になります。

- データ名 I

入力パラメタを設定します。

- データ名 J

応答格納領域長（データ名 K の長さ）を設定します。データ名 J 自身の長さは含みません。1 から DCRPC_MAX_MESSAGE_SIZE* までの値が設定できます。

注※

クライアント環境定義 DCCLTRPCMAXMSGSIZE に 2 以上を指定した場合、DCRPC_MAX_MESSAGE_SIZE の値（1 メガバイト）ではなく、クライアント環境定義 DCCLTRPCMAXMSGSIZE に指定した値になります。

- データ名 K

応答を格納する領域です。データ名 J で設定する長さ以上の領域を用意してください。

(4) 値が返されるデータ領域

- データ名 B

ステータスコードが、5 けたの数字で返されます。

- データ名 D

OpenTP1 で使用する領域です。

• データ名 J

応答の長さ（データ名 K の長さ）が返されます。データ名 C に 1 または 33 を指定したときは、返されません。

• データ名 K

応答が返されます。データ名 C に 1 または 33 を指定したときは、返されません。

(5) ステータスコード

ステータスコード	意味
00000	正常終了しました。
02401	データ名に設定した値が間違っています。要求コード（データ名 A）が間違っている場合も含まれます。
02402	CBLDCRPS('OPEN ')が実行されていません。
02403	次のどれかの要因で、このステータスコードが戻りました。 <ul style="list-style-type: none">• 初期化に失敗しました。• ユーザ認証がされていません。• クライアント環境定義の指定が誤っています。
02404	メモリ不足が発生しました。
02406	ネットワーク障害が発生しました。
02407	CBLDCRPS('CALL ')の処理で時間切れ（タイムアウト）が発生しました。または、サービス要求先の SPP が処理完了前に異常終了しました。
02408	入力パラメタ長が最大値を超えました。
02409	返ってきた応答の長さが、CUP で用意した領域を超えています。
02410	データ名 E に指定したサービスグループ名は定義されていません。
02411	データ名 F に指定したサービス名は定義されていません。
02412	データ名 F に指定したサービスが存在するサービスグループは、閉塞されています。
02413	指定したサービスは終了処理中です。
02414	サービス要求先の SPP が未起動であるか、または処理完了前に異常終了しました。この値はクライアント環境定義 DCWATCHTIM に 0 を指定（応答無限待ち指定）した場合に戻ります。
02415	設定したサービスが存在するノードの OpenTP1 が実行されていません。
02416	指定したサービスでシステムエラーが発生しました。
02417	指定したサービスでメモリ不足が発生しました。
02418	システムエラーが発生しました。
02419	サービスプログラムが OpenTP1 に返した応答長が 1 から

ステータスコード	意味
02419	DCRPC_MAX_MESSAGE_SIZE*までの範囲にありません。
02420	サービス要求されたノードにある OpenTP1 は開始処理中です。
02423	メモリ不足が発生しました。
02424	システムエラーが発生しました。
02425	指定したサービスでシステムエラーが発生しました。
02426	返ってきた応答が、CUP で用意した領域に収まりません。
02427	ノード間負荷バランス機能の環境で、複数の SPP のトランザクション属性が一致していません。このステータスコードは、ノード間負荷バランス機能を使用している SPP にサービスを要求した場合にだけ戻ります。
02442	常設コネクションが解放されました。
02456	サービス要求先のソケット受信型サーバがサービス要求を受信できません。
02466	クライアント環境定義 DCUTOKEY を指定している環境下で、ユーザサービス定義で test_mode=no と指定した SPP に対してサービス要求しました。 または、次の条件が重なった環境下から関数を呼び出しています。 <ul style="list-style-type: none"> クライアント環境定義 DCUTOKEY を指定しています。 CUP 実行プロセスとの常設コネクションが確立中です。 トランザクションの範囲外です。 ユーザサービス定義で test_mode=no と指定した SPP に対してサービス要求しました。
02467	トランザクション処理の連鎖 RPC を使ったあとで、データ名 C に 32 を設定した CBLDCRPS('CALL ') でサービスを要求しています。
02470	サービス要求先の SPP は、セキュリティ機能で保護されています。 CBLDCRPS('CALL ') を呼び出した UAP には、サーバ UAP へのアクセス権限がありません。
02472	同時に起動できるトランザクションブランチの数を越えたため、トランザクションブランチを開始できません。 または、一つのトランザクションブランチから開始できる子トランザクションブランチの最大数を越えたため、トランザクションブランチを開始できません。 または、トランザクション内でドメイン修飾をしたサービス要求で、データ名 C に 32 を合わせて設定していません。
02478	サービス要求先の SPP が処理完了前に異常終了しました。この値はクライアント環境定義 DCEXTENDFUNCTION に 00000001 を指定した場合に戻ります。00000000 を指定しているか指定を省略していると、ステータスコードには 02407、または 02414 が戻ります。
02479	サービス要求先の TP1/Server Base のバージョンが古い (03-03 以降でない) ため、データ圧縮機能は使用できません。このステータスコードは、トランザクションの範囲内でサービス要求した場合に戻ります。
02544	データ名 G に指定したクライアント ID は CBLDCCLS('CLTIN '), または CBLDCCLS('EXCLTIN ') で受け取ったクライアント ID と異なっています。
02547	指定したポート番号は使用されています。または、OS が自動的に割り当てるポート番号が不足しています。

注※

クライアント環境定義 DCCLTRPCMAXMSGSIZE に 2 以上を指定した場合、DCRPC_MAX_MESSAGE_SIZE の値 (1 メガバイト) ではなく、クライアント環境定義 DCCLTRPCMAXMSGSIZE に指定した値になります。

(6) 注意事項

- 入力パラメタ, およびサービスプログラムの応答に同じバッファを指定しないでください。
- データ名 C に 1 を指定した場合, 次のステータスコードは戻りません。

発生しないエラー

02409

02419

発生しても検出できないエラー

02411

02412

02413

02416

02417

02420

- ステータスコード 02407 が戻る場合, 次に示す要因が考えられます。
 - クライアント環境定義で指定した最大応答待ち時間が短い
 - サービス要求先の SPP から発行したサービスプログラムの異常終了
 - サービス要求先の SPP が存在するノードの障害
 - サービス要求先の SPP の処理完了前での異常終了
 - ネットワーク障害

上記の場合, サービス要求先の SPP から開始したトランザクションの処理はコミットされて, データベースが更新されていることがあります。データベースが更新されているかどうかを確認してください。

- CBLDCTRS('BEGIN ')を実行後, CBLDCRPS('CALL ')を実行して, 次に示すステータスコードが戻った場合は, 必要があればロールバック要求のプログラムを発行してください。

02407

02411

02417

02419

02423

02424

02425

02426

6.3.4 CBLDCRPS('SETWATCH') – サービス応答待ち時間の更新

(1) 形式

(a) マルチスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCRPS' USING 一意名1 一意名2 一意名3
```

■ DATA DIVISION の指定

```
01 一意名1.  
  02 データ名A PIC X(8) VALUE 'SETWATCH'.  
  02 データ名B PIC X(5).  
  02 FILLER PIC X(3).  
  02 データ名C PIC S9(9) COMP VALUE ZERO.  
  02 データ名D PIC 9(9) COMP-X.  
  
01 一意名2.  
  02 FILLER PIC X(1).  
  
01 一意名3.  
  02 FILLER PIC X(1).
```

(b) シングルスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCRPC' USING 一意名1
```

■ DATA DIVISION の指定

```
01 一意名1.  
  02 データ名A PIC X(8) VALUE 'SETWATCH'.  
  02 データ名B PIC X(5).  
  02 FILLER PIC X(3).  
  02 データ名C PIC S9(9) COMP VALUE ZERO.
```

(2) 機能

サービス要求の応答待ち時間を変更します。設定した場合、以降の CBLDCRPS('CALL') では、この要求コードで設定したサービス要求の応答待ち時間が使用されます。変更した値は、CBLDCRPS('CLOSE') を実行するまで有効です。ただし、CBLDCRPS('SETWATCH') は、クライアント環境定義 DCWATCHTIM に指定した値を変更しません。

サービス要求の応答待ち時間を CBLDCRPS('SETWATCH') の実行前に戻すときは、CBLDCRPS('GETWATCH') で返された元の値を、再び設定してください。

(3) UAP で値を設定するデータ領域

- データ名 A
サービス応答待ち時間の更新を示す要求コードを「VALUE 'SETWATCH」と設定します。
- データ名 C
変更後のサービス応答待ち時間を設定します。1 から 65535 の範囲で設定します。無制限に待ち続ける場合は、0 を設定します。
- データ名 D
CBLDCCLS('CLTIN '), または CBLDCCLS('EXCLTIN ') で受け取ったクライアント ID を指定します。

(4) 値が返されるデータ領域

- データ名 B
ステータスコードが、5 けたの数字で返されます。

(5) ステータスコード

ステータスコード	意味
00000	正常終了しました。
02401	データ名に設定した値が間違っています。 要求コード (データ名 A) が間違っている場合も含まれます。
02402	CBLDCRPS('OPEN ') が実行されていません。
02404	メモリ不足が発生しました。
02544	データ名 D に指定したクライアント ID は CBLDCCLS('CLTIN '), または CBLDCCLS('EXCLTIN ') で受け取ったクライアント ID と異なります。

6.3.5 CBLDCRPS('GETWATCH') - サービス応答待ち時間の参照

(1) 形式

(a) マルチスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCRPS' USING 一意名1 一意名2 一意名3
```

■ DATA DIVISION の指定

```
01 一意名1.  
  02 データ名A PIC X(8) VALUE 'GETWATCH'.  
  02 データ名B PIC X(5).  
  02 FILLER PIC X(3).  
  02 データ名C PIC S9(9) COMP VALUE ZERO.  
  02 データ名D PIC 9(9) COMP-X.  
  
01 一意名2.  
  02 FILLER PIC X(1).  
  
01 一意名3.  
  02 FILLER PIC X(1).
```

(b) シングルスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCRPC' USING 一意名1
```

■ DATA DIVISION の指定

```
01 一意名1.  
  02 データ名A PIC X(8) VALUE 'GETWATCH'.  
  02 データ名B PIC X(5).  
  02 FILLER PIC X(3).  
  02 データ名C PIC S9(9) COMP VALUE ZERO.
```

(2) 機能

現在のサービス要求の応答待ち時間を参照します。CBLDCRPS('SETWATCH')は、CBLDCRPS('SETWATCH')でサービス応答待ち時間を一時的に変更する前に、元の値を退避するために使用します。

CBLDCRPS('SETWATCH')は、CBLDCRPS('SETWATCH')で変更したサービスの応答時間をリターンします。変更していない場合は、クライアント環境定義 DCWATCHTIM の値をリターンします。

得られる値は、OpenTP1 の CBLDCRPS('CALL ')に対して有効です。

(3) UAP で値を設定するデータ領域

- データ名 A
サービス応答待ち時間の参照を示す要求コードを「VALUE 'GETWATCH'」と設定します。
- データ名 C
0 を設定します。
- データ名 D

CBLDCCLS('CLTIN '), または CBLDCCLS('EXCLTIN ') で受け取ったクライアント ID を指定します。

(4) 値が返されるデータ領域

- データ名 B

ステータスコードが、5 けたの数字で返されます。

- データ名 C

現在のサービス応答待ち時間が返されます。0 が返された場合は、無制限に応答を待ち続ける指定であることを示します。

(5) ステータスコード

ステータスコード	意味
00000	正常終了しました。
02401	要求コード (データ名 A) に指定した値が間違っています。
02402	CBLDCRPS('OPEN ') が実行されていません。
02404	メモリ不足が発生しました。
02544	データ名 D に指定したクライアント ID は CBLDCCLS('CLTIN '), または CBLDCCLS('EXCLTIN ') で受け取ったクライアント ID と異なります。

6.4 常設コネクション

6.4.1 CBLDCCLS('CONNECT ') – 常設コネクションの確立

(1) 形式

(a) マルチスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCCLS' USING 一意名1
```

■ DATA DIVISION の指定

```
01 一意名1.  
02 データ名A PIC X(8) VALUE 'CONNECT'.  
02 データ名B PIC X(5).  
02 FILLER PIC X(3).  
02 データ名C PIC S9(9) COMP VALUE ZERO.  
02 データ名D PIC 9(9) COMP-X.
```

(b) シングルスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCCLT' USING 一意名1
```

■ DATA DIVISION の指定

```
01 一意名1.  
02 データ名A PIC X(8) VALUE 'CONNECT'.  
02 データ名B PIC X(5).  
02 FILLER PIC X(3).  
02 データ名C PIC S9(9) COMP VALUE ZERO.
```

(2) 機能

CUP 実行プロセス、rap サーバ、または DCCM3 の論理端末との間に常設コネクションを確立します。

常設コネクションを確立する CUP 実行プロセスが起動されている OpenTP1 ノードは、CBLDCCLS('CLTIN')のデータ名 D に指定した OpenTP1 ノード、クライアント環境定義 DCCLTRAPHOST または DCHOST に指定した OpenTP1 ノードです。

DCCM3 の論理端末との間に常設コネクションを確立する場合、クライアント環境定義に DCCLTDCCMHOST および DCCLTDCCMPORT を定義し、CBLDCCLS('CONNECT ')のデータ名 C に 32 を指定します。

また、リモート API 機能を使用する場合、DCCM3 の論理端末との間に常設コネクションを確立するには、DCCLTRAPHOST に DCCM3 の論理端末のホスト名およびポート番号を指定し、CBLDCCLS('CONNECT ')のデータ名 C に 0 を指定します。

(3) UAP で値を設定するデータ領域

- データ名 A

常設コネクションの確立を示す要求コードを「VALUE 'CONNECT△」と設定します。

- データ名 C

常設コネクションを確立する通信相手を指定します。

0

CUP 実行プロセス、rap サーバまたは DCCM3 の論理端末との間に常設コネクションを確立します。

32

DCCM3 の論理端末との間に常設コネクションを確立します。

- データ名 D

CBLDCCLS('CLTIN '), または CBLDCCLS('EXCLTIN ')で受け取ったクライアント ID を指定します。

(4) 値が返されるデータ領域

- データ名 B

ステータスコードが、5 けたの数字で返されます。

(5) ステータスコード

ステータスコード	意味
00000	正常終了しました。または、すでに常設コネクションが確立されています。
02501	データ名に指定した値が間違っています。要求コード（データ名 A）が間違っている場合も含まれます。
02502	次のどれかの要因が考えられます。 <ul style="list-style-type: none">• トランザクション内で発行されています• CBLDCRPS('OPEN ')が発行されていません。• OpenTP1 に対する確立要求が発行されましたが、すでに DCCM3 との常設コネクションが確立されています。• DCCM3 に対する確立要求が発行されましたが、すでに OpenTP1 との常設コネクションが確立されています。

ステータスコード	意味
02504	必要なバッファが確保できませんでした。
02506	通信障害が発生しました。
02507	常設コネクション確立時に時間切れ（タイムアウト）が発生しました。
02515	次のどれかの要因が考えられます。 <ul style="list-style-type: none"> • OpenTP1 サーバ，または DCCM3 論理端末が起動されていません。 • クライアント拡張サービスが起動されていません。システムサービス構成定義 clt_conf の指定が正しいか見直してください。 • CUP 実行プロセスが起動されていません。クライアントサービス定義 clt_cup_conf の指定が正しいか見直してください。
02518	システムエラーが発生しました。
02539	DCCM3 の論理端末に対する確立要求が発行されましたが，ホスト名が不正です。
02544	データ名 D に指定したクライアント ID は CBLDCCLS('CLTIN ')，または CBLDCCLS('EXCLTIN ') で受け取ったクライアント ID と異なります。
02547	指定したポート番号は使用されています。または，OS が自動的に割り当てるポート番号が不足しています。

(6) 注意事項

- CBLDCCLS('CONNECT ')がエラーリターンした場合，常設コネクションは確立されません。ただし，次のステータスコードでエラーリターンした場合，CUP 実行プロセス側では常設コネクションが確立状態になることがあります。

- 02506
- 02507
- 02518

このとき，CUP 実行プロセスまたは DCCM3 の論理端末は，CUP からの要求を待ち続ける場合があります。要求待ちを避けるため，常設コネクション問い合わせ間隔最大時間（DCCM3 論理端末の場合は端末放置監視時間）に適切な値を設定してください。

- CBLDCCLS('CONNECT ')をトランザクション内で発行することはできません。
- CUP 実行プロセス，rap サーバ，またはクライアント環境定義 DCCLTRAPHOST に指定した DCCM3 の論理端末との間に常設コネクションを確立した場合は，CBLDCCLS('DISCNCT ')を発行するまでクライアント環境定義 DCCLTDCCMHOST に指定した DCCM3 の論理端末との通信はできません。

また，クライアント環境定義 DCCLTDCCMHOST に指定した DCCM3 の論理端末との間に常設コネクションを確立した場合は，CBLDCCLS('DISCNCT ')を発行するまで CUP 実行プロセス，rap サーバ，またはクライアント環境定義 DCCLTRAPHOST に指定した DCCM3 の論理端末との通信はできません。

- DCCM3 の論理端末との間に常設コネクションを確立する場合、データ圧縮機能は使用できません。クライアント環境定義 DCCLTDATACOMP の指定を省略するか、または DCCLTDATACOMP に N を指定する必要があります。

6.4.2 CBLDCCLS('DISCNCT') – 常設コネクションの解放

(1) 形式

(a) マルチスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCCLS' USING 一意名1
```

■ DATA DIVISION の指定

```
01 一意名1.
02 データ名A PIC X(8) VALUE 'DISCNCT'.
02 データ名B PIC X(5).
02 FILLER PIC X(3).
02 データ名C PIC S9(9) COMP VALUE ZERO.
02 データ名D PIC 9(9) COMP-X.
```

(b) シングルスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCCLT' USING 一意名1
```

■ DATA DIVISION の指定

```
01 一意名1.
02 データ名A PIC X(8) VALUE 'DISCNCT'.
02 データ名B PIC X(5).
02 FILLER PIC X(3).
02 データ名C PIC S9(9) COMP VALUE ZERO.
```

(2) 機能

CUP 実行プロセス、rap サーバまたは DCCM3 の論理端末との間の常設コネクションを解放します。

(3) UAP で値を設定するデータ領域

- データ名 A

常設コネクションの解放を示す要求コードを「VALUE 'DISCNCT△」と設定します。

- データ名 C
0 を指定します。
- データ名 D
CBLDCCLS('CLTIN '), または CBLDCCLS('EXCLTIN ') で受け取ったクライアント ID を指定します。

(4) 値が返されるデータ領域

- データ名 B
ステータスコードが、5 けたの数字で返されます。

(5) ステータスコード

ステータスコード	意味
00000	正常終了しました。または、TP1/Client/W の場合、すでに常設コネクションが切断されています。
02501	データ名に指定した値が間違っています。要求コード（データ名 A）が間違っている場合も含まれます。
02502	トランザクション内で発行されているか、CBLDCCLS('OPEN ') が発行されていません。
02504	必要なバッファが確保できませんでした。
02506	通信障害が発生しました。または、TP1/Client/P の場合、すでに常設コネクションが切断されています。
02507	常設コネクション確立時に時間切れ（タイムアウト）が発生しました。
02518	システムエラーが発生しました。
02544	データ名 D に指定したクライアント ID は CBLDCCLS('CLTIN '), または CBLDCCLS('EXCLTIN ') で受け取ったクライアント ID と異なっています。

(6) 注意事項

- CBLDCCLS('DISCNCT ') が次のステータスコードでエラーリターンした場合、常設コネクションは解放されません。
 - 02501
 - 02502
 - 02504（クライアント側で検知した場合）
 - 02544
- CBLDCCLS('DISCNCT ') が次のステータスコードでエラーリターンしたとき、TP1/Client 側で強制的に常設コネクションを解放します。
 - 02504（サーバ側で検知した場合）
 - 02506

- 02507
- 02518

このとき、CUP 実行プロセスまたは DCCM3 の論理端末は TP1/Client で常設コネクションが解放されたことを検知しないで、CUP からの要求を待ち続けることがあります。要求待ちを避けるため、常設コネクション問い合わせ間隔最大時間（DCCM3 論理端末の場合は端末放置監視時間）に適切な値を設定してください。

- CBLDCCLS('DISCNCT ')をトランザクション内で発行した場合、トランザクションはコミットされます。

6.4.3 CBLDCCLS('STRAPHST') – 常設コネクション確立要求先の指定

(1) 形式

(a) マルチスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCCLS' USING 一意名1
```

■ DATA DIVISION の指定

```
01 一意名1.
02 データ名A PIC X(8) VALUE 'STRAPHST'.
02 データ名B PIC X(5).
02 FILLER PIC X(3).
02 データ名C PIC S9(9) COMP VALUE ZERO.
02 データ名D PIC 9(9) COMP-X.
02 データ名E PIC X(n).
```

(b) シングルスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCCLT' USING 一意名1
```

■ DATA DIVISION の指定

```
01 一意名1.
02 データ名A PIC X(8) VALUE 'STRAPHST'.
02 データ名B PIC X(5).
02 FILLER PIC X(3).
02 データ名C PIC S9(9) COMP VALUE ZERO.
02 FILLER PIC 9(9) COMP.
02 データ名E PIC X(n).
```

(2) 機能

常設コネクション確立要求先のホスト名およびポート番号を指定します。この関数を使用した場合、クライアント環境定義 DCCLTRAPHOST に定義したホスト名およびポート番号は無視され、以降の CBLDCCLS('CONNECT ')では、この関数で指定したホスト名およびポート番号が使用されます。

常設コネクション確立要求先のホスト名およびポート番号をこの関数の実行前に戻すときは、CBLDCCLS('GTRAPHST')で返された元の値を、この関数で再設定してください。

(3) UAP で値を設定するデータ領域

- データ名 A

常設コネクション確立要求先の指定を示す要求コードを「VALUE 'STRAPHST'」と設定します。

- データ名 C

0 を設定します。

- データ名 D

CBLDCCLS('CLTIN '), または CBLDCCLS('EXCLTIN ')で受け取ったクライアント ID を指定します。

- データ名 E

常設コネクション確立要求先のホスト名およびポート番号を指定します。また、ホスト名として、10進ドット記法の IP アドレスを指定することもできます。

形式

ホスト名 [:ポート番号] [,ホスト名 [:ポート番号] ,...]

- ホスト名 ~ 〈文字列〉
- ポート番号 ~ 〈符号なし整数〉 ((5001~65535))

ホスト名として指定できる長さは、63 文字*までです。ホスト名を複数指定する場合、データ名 E に指定できる長さ（ポート番号なども含む）は、255 文字*までです。

文字列の終わり以外は空白文字（スペースまたはタブ）を入れないでください。

注※

クライアント環境定義 DCCLTOPTION に 00000008 を指定した場合、ホスト名として指定できる長さは、255 文字までです。ホスト名を複数指定する場合、データ名 E に指定できる長さ（ポート番号なども含む）は 1023 文字までとなります。

(4) 値が返されるデータ領域

- データ名 B

ステータスコードが 5 けたの数字で返されます。

(5) ステータスコード

ステータスコード	意味
00000	正常終了しました。
02501	引数に指定した値が間違っています。要求コード（データ名 A）が間違っている場合も含まれます。
02502	次のどれかの要因が考えられます。 <ul style="list-style-type: none">• トランザクション内で発行されています。• 常設コネクション確立中です。• CBLDCRPS('OPEN ')が発行されていません。
02504	必要なバッファが確保できませんでした。
02544	データ名 D に指定したクライアント ID は CBLDCCLS('CLTIN '), または CBLDCCLS('EXCLTIN ') で受け取ったクライアント ID と異なっています。

(6) 注意事項

- この関数は、クライアント環境定義 DCCLTRAPHOST に指定した値を変更しません。
- データ名 E の先頭に空白文字を指定した場合、クライアント環境定義 DCCLTRAPHOST を定義していない状態になります。この場合、以降に実行する CBLDCCLS('CONNECT ') では、CUP 実行プロセスまたは DCCM3 の論理端末に常設コネクションを確立します。

6.4.4 CBLDCCLS('GTRAPHST') – 常設コネクション確立要求先の取得

(1) 形式

(a) マルチスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCCLS' USING 一意名1
```

■ DATA DIVISION の指定

```
01 一意名1.  
02 データ名A PIC X(8) VALUE 'GTRAPHST'.  
02 データ名B PIC X(5).  
02 FILLER PIC X(3).  
02 データ名C PIC S9(9) COMP VALUE ZERO.  
02 データ名D PIC 9(9) COMP-X.  
02 データ名E PIC X(n).
```

(b) シングルスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCCLT' USING 一意名1
```

■ DATA DIVISION の指定

```
01 一意名1.  
02 データ名A PIC X(8) VALUE 'GTRAPHST'.  
02 データ名B PIC X(5).  
02 FILLER PIC X(3).  
02 データ名C PIC S9(9) COMP VALUE ZERO.  
02 FILLER PIC 9(9) COMP.  
02 データ名E PIC X(n).
```

(2) 機能

常設コネクション確立要求先のホスト名およびポート番号を取得します。この関数は、CBLDCCLS('STRAPHST')で常設コネクション確立要求先を変更する前に、元の値を退避するために使用します。

この関数は、CBLDCCLS('STRAPHST')で変更した常設コネクション確立要求先をデータ名 E に返します。変更していない場合は、クライアント環境定義 DCCLTRAPHOST の値をデータ名 E に返します。

(3) UAP で値を設定するデータ領域

• データ名 A

常設コネクション確立要求先の取得を示す要求コードを「VALUE 'GTRAPHST'」と設定します。

• データ名 C

0 を設定します。

• データ名 D

CBLDCCLS('CLTIN '), または CBLDCCLS('EXCLTIN ')で受け取ったクライアント ID を指定します。

• データ名 E

現在設定されている常設コネクション確立要求先のホスト名およびポート番号を格納する 256 バイト
※以上の領域を用意してください。

注※

クライアント環境定義 DCCLTOPTION に 00000008 を指定した場合、256 バイトではなく、1024 バイトになります。

(4) 値が返されるデータ領域

• データ名 B

ステータスコードが5けたの数字で返されます。

• データ名 E

現在設定されている常設コネクション確立要求先のホスト名およびポート番号が返されます。クライアント環境定義 DCCLTRAPHOST を定義しないで、かつ CBLDCCLS('STRAPHST') で常設コネクション確立要求先を指定していない場合は、データ名 E の先頭にスペースが返されます。

形式

ホスト名 [:ポート番号] [,ホスト名 [:ポート番号] ,...]

- ・ホスト名 ~ 〈文字列〉
- ・ポート番号 ~ 〈符号なし整数〉 ((5001~65535))

(5) ステータスコード

ステータスコード	意味
00000	正常終了しました。
02501	引数に指定した値が間違っています。要求コード（データ名 A）が間違っている場合も含まれます。
02502	CBLDCRPS('OPEN ')が発行されていません。
02504	必要なバッファが確保できませんでした。
02544	データ名 D に指定したクライアント ID は CBLDCCLS('CLTIN '), または CBLDCCLS('EXCLTIN ') で受け取ったクライアント ID と異なっています。

6.4.5 CBLDCCLS('STCONINF') – 端末識別情報の設定

(1) 形式

(a) マルチスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCCLS' USING 一意名1
```

■ DATA DIVISION の指定

```
01 一意名1.  
02 データ名A PIC X(8) VALUE 'STCONINF'.  
02 データ名B PIC X(5).  
02 FILLER PIC X(3).  
02 データ名C PIC S9(9) COMP VALUE ZERO.  
02 データ名D PIC 9(4) COMP.  
02 FILLER PIC X(2).  
02 データ名F PIC 9(9) COMP-X.  
02 データ名G PIC X(n).
```

(b) シングルスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCCLT' USING 一意名1
```

■ DATA DIVISION の指定

```
01 一意名1.  
02 データ名A PIC X(8) VALUE 'STCONINF'.  
02 データ名B PIC X(5).  
02 FILLER PIC X(3).  
02 データ名C PIC S9(9) COMP VALUE ZERO.  
02 データ名D PIC 9(4) COMP.  
02 FILLER PIC X(2).  
02 データ名E PIC 9(9) COMP.  
02 データ名G PIC X(n).
```

(2) 機能

端末識別情報を動的に設定します。

常設コネクションを使用して DCCM3 論理端末と通信する場合、端末識別情報を DCCM3 論理端末に通知することで、DCCM3 の端末固定割り当て機能を利用できます。

この要求コードのデータ名 G に設定した端末識別情報は、クライアント環境定義 DCCLTRAPHOST に DCCM3 論理端末のホスト名およびポート番号を指定し、かつ CBLDCCLS('CONNECT ')のデータ名 C に 0 を設定した場合だけ、有効です。この要求コードのあとに実行する CBLDCCLS('CONNECT ')で参照され、DCCM3 論理端末に通知されます。

この要求コードを実行した場合、クライアント環境定義 DCCLTCONNECTINF に指定した端末識別情報は、CBLDCRPS('OPEN ')を再び実行するまで参照されません。

また、この要求コードを複数回実行した場合は、CBLDCCLS('CONNECT ')実行直前に設定した端末識別情報が、有効となります。

(3) UAP で値を設定するデータ領域

- データ名 A

端末識別情報の設定を示す要求コードとして、「VALUE 'STCONINF'」を設定します。

- データ名 C

0 を設定します。

- データ名 D

端末識別情報長を設定します。

- データ名 E

OpenTP1 で使用する領域です。

- データ名 F

CBLDCCLS('CLTIN '), または CBLDCCLS('EXCLTIN ') で受け取ったクライアント ID を設定します。

- データ名 G

端末識別情報を設定します。

(4) 値が返されるデータ領域

- データ名 B

ステータスコードが、5 けたの数字で返されます。

(5) ステータスコード

ステータスコード	意味
00000	正常終了しました。
02501	データ名に指定した値が間違っています。要求コード (データ名 A) が間違っている場合も含まれます。
02502	CBLDCRPS('OPEN ') が発行されていません。
02504	必要なバッファが確保できませんでした。
02544	データ名 F に指定したクライアント ID は CBLDCCLS('CLTIN '), または CBLDCCLS('EXCLTIN ') で受け取ったクライアント ID と異なります。

(6) 注意事項

- 端末識別情報を通知することで、DCCM3 の端末固定割り当て機能を利用できるのは、DCCM3 のバージョン 09-03 以降です。端末固定割り当て機能については、マニュアル「VOS3 データマネジメントシステム XDM E2 系 解説」を参照してください。
- CBLDCCLS('STCONINF') で定義した端末識別情報と一致する DCCM3 論理端末の論理端末名称が DCCM3 側で定義されていない場合、CBLDCCLS('CONNECT ') はステータスコード 02506 を返します。

6.5 トランザクション制御

6.5.1 CBLDCTRS('BEGIN ') – トランザクションの開始

(1) 形式

(a) マルチスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCTRS' USING 一意名1
```

■ DATA DIVISION の指定

```
01 一意名1.  
02 データ名A PIC X(8) VALUE 'BEGIN' .  
02 データ名B PIC X(5).  
02 FILLER PIC X(3).  
02 データ名C PIC 9(9) COMP-X.
```

(b) シングルスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCTRN' USING 一意名1
```

■ DATA DIVISION の指定

```
01 一意名1.  
02 データ名A PIC X(8) VALUE 'BEGIN' .  
02 データ名B PIC X(5).
```

(2) 機能

グローバルトランザクションを、CBLDCTRS('BEGIN ')を実行する CUP のプロセスから開始します。

CBLDCTRS('BEGIN ')は、CBLDCRPS('OPEN ')を実行したあとに発行してください。

CBLDCTRS('BEGIN ')を実行してから、トランザクションの同期点（コミットの要求）までが一つのグローバルトランザクションとなります。

また、グローバルトランザクション中では、CBLDCTRS('BEGIN ')を重複して発行できません（SPPでのCBLDCTRS('BEGIN ')を含みます）。発行した場合はエラーリターンします。

SPP のトランザクション属性は、ユーザサービス定義の atomic_update の指定に従います。

(3) UAP で値を設定するデータ領域

- データ名 A

トランザクションの開始を示す要求コードを「VALUE 'BEGIN△△△」と設定します。

- データ名 C

CBLDCCLS('CLTIN '), または CBLDCCLS('EXCLTIN ') で受け取ったクライアント ID を指定します。

(4) 値が返されるデータ領域

- データ名 B

ステータスコードが、5 けたの数字で返されます。

(5) ステータスコード

ステータスコード	意味
00000	正常終了しました。
02501	要求コード (データ名 A) に指定した値が間違っています。
02502	誤ったコンテキスト (例 すでにトランザクション内にいる) からプログラムを発行しています。 または、次の条件が重なった環境下から関数を呼び出しています。 <ul style="list-style-type: none">クライアント環境定義 DCUTOKEY を指定しています。rap サーバとの常設コネクションが確立中です。
02504	メモリ不足が発生しました。
02506	ネットワーク障害が発生しました。
02507	CBLDCTRS('BEGIN ') の処理時間で時間切れ (タイムアウト) が発生しました。
02510	クライアント拡張サービスが起動されていません。システムサービス構成定義 clt_conf の指定が正しいか見直してください。または、トランザクショナル RPC 実行プロセスが起動されていません。クライアントサービス定義 clt_trn_conf の指定が正しいか見直してください。
02515	OpenTP1 が起動されていません。
02517	トランザクションプロセス内でメモリ不足が発生しました。
02518	システムエラーが発生しました。
02542	CUP 実行プロセス側から常設コネクションが解放されました。
02544	データ名 C に指定したクライアント ID は CBLDCCLS('CLTIN '), または CBLDCCLS('EXCLTIN ') で受け取ったクライアント ID と異なります。
02545	サーバのトランザクション処理に負荷が掛かり過ぎているため、トランザクションを開始できません。 このステータスコードが戻った場合は、再び実行すれば成功する可能性が高いので、再実行してください。

ステータスコード	意味
02547	指定したポート番号は使用されています。または、OSが自動的に割り当てるポート番号が不足しています。
03406	リソースマネージャ (RM) でエラーが発生しました。トランザクションは発生できませんでした。
03407	トランザクションサービスでエラーが発生したので、トランザクションを開始できませんでした。このステータスコードが戻った場合は、再び実行すれば成功する可能性が高いので、再び実行してください。

6.5.2 CBLDCTRS('C-COMMIT') – 連鎖モードのコミット

(1) 形式

(a) マルチスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCTRS' USING 一意名1
```

■ DATA DIVISION の指定

```
01 一意名1.
02 データ名A PIC X(8) VALUE 'C-COMMIT'.
02 データ名B PIC X(5).
02 FILLER PIC X(3).
02 データ名C PIC 9(9) COMP-X.
```

(b) シングルスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCTRN' USING 一意名1
```

■ DATA DIVISION の指定

```
01 一意名1.
02 データ名A PIC X(8) VALUE 'C-COMMIT'.
02 データ名B PIC X(5).
```

(2) 機能

トランザクションの同期点を取得します。

CBLDCTRS('C-COMMIT')が正常終了すると、新しいグローバルトランザクションが発生し、以降実行するプログラムは新しいグローバルトランザクションの範囲になります。

(3) UAP で値を設定するデータ領域

- データ名 A

連鎖モードのコミットを示す要求コードを「VALUE 'C-COMMIT」と設定します。

- データ名 C

CBLDCCLS('CLTIN '), または CBLDCCLS('EXCLTIN ') で受け取ったクライアント ID を指定します。

(4) 値が返されるデータ領域

- データ名 B

ステータスコードが、5 けたの数字で返されます。

(5) ステータスコード

ステータスコード	意味
00000	正常終了しました。
02501	要求コード (データ名 A) に指定した値が間違っています。
02502	誤ったコンテキストからプログラムを呼び出しています。
02504	メモリ不足が発生しました。
02506	ネットワーク障害が発生しました。
02507	CBLDCTRS('C-COMMIT') の処理時間で時間切れ (タイムアウト) が発生しました。
02515	OpenTP1 が起動されていません。
02517	トランザクションプロセス内でメモリ不足が発生しました。
02518	システムエラーが発生しました。
02542	常設コネクションが解放されました。
02544	データ名 C に指定したクライアント ID は CBLDCCLS('CLTIN '), または CBLDCCLS('EXCLTIN ') で受け取ったクライアント ID と異なります。
03402	現在のトランザクションは、コミットに失敗したためロールバックしました。 完了後、このプロセスはトランザクション下において、グローバルトランザクションの範囲内です。
03403	ヒューリスティック決定のため、あるトランザクションブランチはコミットとなり、あるトランザクションはロールバックとなりました。 このステータスコードは、ヒューリスティック決定の結果が、グローバルトランザクションの同期点の結果と一致しなかった場合に返ります。 このステータスコードが返る原因およびグローバルトランザクションの同期点の結果は、メッセージログファイルを参照してください。 このステータスコードが戻ったあともこのプロセスはトランザクション下において、グローバルトランザクションの範囲内です。

ステータスコード	意味
03404	グローバルトランザクションのトランザクションブランチがヒューリスティックに完了しました。しかし、障害のためヒューリスティックに完了したトランザクションブランチの同期点の結果が判明しません。このステータスコードが返る原因およびグローバルトランザクションの同期点の結果は、メッセージログファイルを参照してください。このステータスコードが戻ったあともこのプロセスはトランザクション下において、グローバルトランザクションの範囲内です。
03424	コミットは正常終了しましたが、新しいトランザクションは開始できませんでした。このステータスコードが返ったあとは、このプロセスはトランザクション下にはありません。
03425	現在のトランザクションは、コミットできないのでロールバックしました。その後、新しいトランザクションは開始できなかったため、このプロセスはトランザクション下にはありません。
03426	CBLDCTRS('C-COMMIT')を実行したグローバルトランザクションは、ヒューリスティック決定のため、トランザクションブランチにはコミットされたものと、されていないものがあります。このステータスコードは、ヒューリスティック決定の結果が、グローバルトランザクションの同期点の結果と一致しなかった場合に返ります。このステータスコードが返る原因となった UAP、リソースマネージャ、およびグローバルトランザクションの同期点の結果は、メッセージログファイルを参照してください。その後、新しいトランザクションを開始できなかったため、このプロセスはトランザクション下にはありません。
03427	グローバルトランザクションのトランザクションブランチが、ヒューリスティックに完了しました。しかし、障害のため、ヒューリスティックに完了したトランザクションブランチの同期点の結果が判明しません。このステータスコードが返る原因となった UAP、リソースマネージャ、およびグローバルトランザクションの同期点の結果は、メッセージログファイルを参照してください。その後、新しいトランザクションを開始できなかったため、このプロセスはトランザクション下にはありません。このステータスコードが戻ったあともこのプロセスはトランザクション下において、グローバルトランザクションの範囲内です。

(6) 注意事項

トランザクションをコミットしてから CUP のプロセスを終了させるときは、CBLDCTRS('U-COMMIT')を必ず実行してください。

6.5.3 CBLDCTRS('C-ROLL ') – 連鎖モードのロールバック

(1) 形式

(a) マルチスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCTRS' USING 一意名1
```

■ DATA DIVISION の指定

```
01 一意名1.  
02 データ名A PIC X(8) VALUE 'C-ROLL '.  
02 データ名B PIC X(5).  
02 FILLER PIC X(3).  
02 データ名C PIC 9(9) COMP-X.
```

(b) シングルスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCTRN' USING 一意名1
```

■ DATA DIVISION の指定

```
01 一意名1.  
02 データ名A PIC X(8) VALUE 'C-ROLL '.  
02 データ名B PIC X(5).
```

(2) 機能

トランザクションをロールバックします。

CBLDCTRS('C-ROLL ')が正常終了すると、新しいグローバルトランザクションが発生し、以降実行するプログラムは新しいグローバルトランザクションの範囲になります。

(3) UAP で値を設定するデータ領域

- データ名 A
連鎖モードのロールバックを示す要求コードを「VALUE 'C-ROLL△△」と設定します。
- データ名 C
CBLDCCLS('CLTIN '), または CBLDCCLS('EXCLTIN ')で受け取ったクライアント ID を指定します。

(4) 値が返されるデータ領域

- データ名 B

ステータスコードが、5 けたの数字で返されます。

(5) ステータスコード

ステータスコード	意味
00000	正常終了しました。
02501	要求コード (データ名 A) に指定した値が間違っています。
02502	誤ったコンテキストからプログラムを呼び出しています。
02504	メモリ不足が発生しました。
02506	ネットワーク障害が発生しました。
02507	CBLDCTRS('C-ROLL ')の処理時間で時間切れ (タイムアウト) が発生しました。
02515	OpenTP1 が起動されていません。
02517	トランザクションプロセス内でメモリ不足が発生しました。
02518	システムエラーが発生しました。
02542	常設コネクションが解放されました。
02544	データ名 C に指定したクライアント ID は CBLDCCLS('CLTIN '), または CBLDCCLS('EXCLTIN ')で受け取ったクライアント ID と異なっています。
03403	<p>ヒューリスティック決定のため、あるトランザクションブランチはコミットとなり、あるトランザクションはロールバックとなりました。</p> <p>このステータスコードは、ヒューリスティック決定の結果が、グローバルトランザクションの同期点の結果と一致しなかった場合に返ります。</p> <p>このステータスコードが返る原因およびグローバルトランザクションの同期点の結果は、メッセージログファイルを参照してください。</p> <p>このステータスコードが戻ったあともこのプロセスはトランザクション下において、グローバルトランザクションの範囲内です。</p>
03404	<p>グローバルトランザクションのトランザクションブランチがヒューリスティックに完了しました。しかし、障害のためヒューリスティックに完了したトランザクションブランチの同期点の結果が判明しません。</p> <p>このステータスコードが返る原因およびグローバルトランザクションの同期点の結果は、メッセージログファイルを参照してください。</p> <p>このステータスコードが戻ったあともこのプロセスはトランザクション下において、グローバルトランザクションの範囲内です。</p>
03424	ロールバックは正常終了しましたが、新しいトランザクションは開始できませんでした。このステータスコードが返ったあとは、このプロセスはトランザクション下にはありません。
03426	CBLDCTRS('C-COMMIT')を実行したグローバルトランザクションは、ヒューリスティック決定のため、トランザクションブランチにはコミットされたものと、されていないものがあります。

ステータスコード	意味
03426	<p>このステータスコードは、ヒューリスティック決定の結果が、グローバルトランザクションの同期点の結果と一致しなかった場合に返ります。</p> <p>このステータスコードが返る原因となった UAP、リソースマネージャ、およびグローバルトランザクションの同期点の結果は、メッセージログファイルを参照してください。</p> <p>その後、新しいトランザクションを開始できなかったため、このプロセスはトランザクション下ではありません。</p>
03427	<p>グローバルトランザクションのトランザクションブランチが、ヒューリスティックに完了しました。しかし、障害のため、ヒューリスティックに完了したトランザクションブランチの同期点の結果が判明しません。</p> <p>このステータスコードが返る原因となった UAP、リソースマネージャ、およびグローバルトランザクションの同期点の結果は、メッセージログファイルを参照してください。</p> <p>その後、新しいトランザクションを開始できなかったため、このプロセスはトランザクション下ではありません。</p> <p>このステータスコードが戻ったあともこのプロセスはトランザクション下であって、グローバルトランザクションの範囲内です。</p>

(6) 注意事項

トランザクションをロールバックしてから CUP のプロセスを終了させるときは、CBLDCTRS('U-ROLL')を必ず実行してください。

6.5.4 CBLDCTRS('U-COMMIT') – 非連鎖モードのコミット

(1) 形式

(a) マルチスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCTRS' USING 一意名1
```

■ DATA DIVISION の指定

```
01 一意名1.
02 データ名A PIC X(8) VALUE 'U-COMMIT'.
02 データ名B PIC X(5).
02 FILLER PIC X(3).
02 データ名C PIC 9(9) COMP-X.
```


(b) シングルスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCTRN' USING 一意名1
```

■ DATA DIVISION の指定

```
01 一意名1.  
02 データ名A PIC X(8) VALUE 'U-COMMIT'.  
02 データ名B PIC X(5).
```

(2) 機能

トランザクションの同期点を取得します。

CBLDCTRS('U-COMMIT')が正常終了すると、グローバルトランザクションは終了します。グローバルトランザクションの範囲外からは、SPP をトランザクションとして実行できません。

(3) UAP で値を設定するデータ領域

- データ名 A
非連鎖モードのコミットを示す要求コードを「VALUE 'U-COMMIT'」と設定します。
- データ名 C
CBLDCCLS('CLTIN '), または CBLDCCLS('EXCLTIN ')で受け取ったクライアント ID を指定します。

(4) 値が返されるデータ領域

- データ名 B
ステータスコードが、5 けたの数字で返されます。

(5) ステータスコード

ステータスコード	意味
00000	正常終了しました。
02501	要求コード（データ名 A）に指定した値が間違っています。
02502	誤ったコンテキストからプログラムを呼び出しています。
02504	メモリ不足が発生しました。
02506	ネットワーク障害が発生しました。
02507	CBLDCTRS('U-COMMIT')の処理時間で時間切れ(タイムアウト)が発生しました。
02515	OpenTP1 が起動されていません。

ステータスコード	意味
02517	トランザクションプロセス内でメモリ不足が発生しました。
02518	システムエラーが発生しました。
02542	常設コネクションが解放されました。
02544	データ名 C に指定したクライアント ID は CBLDCCLS('CLTIN '), または CBLDCCLS('EXCLTIN ') で受け取ったクライアント ID と異なります。
03402	コミットに失敗したためロールバックしました。 このステータスコードが戻ったあと、このプロセスはグローバルトランザクションの範囲外となります。
03403	ヒューリスティック決定によって、一部、またはすべてのトランザクションブランチはロールバックされました。詳細は、メッセージログファイルを参照してください。このステータスコードが戻ったあと、グローバルトランザクションの範囲外となります。
03404	ヒューリスティック決定でトランザクションが完了しましたが、障害のため結果がわかりません。詳細は、メッセージログファイルを参照してください。このステータスコードが戻ったあと、グローバルトランザクションの範囲外となります。

(6) 注意事項

CUP のプロセスを正常終了させるときは、CBLDCTRS('U-COMMIT') を必ず実行してトランザクションをコミットしてください。

6.5.5 CBLDCTRS('U-ROLL ') – 非連鎖モードのロールバック

(1) 形式

(a) マルチスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCTRS' USING 一意名1
```

■ DATA DIVISION の指定

```
01 一意名1.
  02 データ名A PIC X(8) VALUE 'U-ROLL '.
  02 データ名B PIC X(5).
  02 FILLER PIC X(3).
  02 データ名C PIC 9(9) COMP-X.
```

(b) シングルスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCTRN' USING 一意名1
```

■ DATA DIVISION の指定

```
01 一意名1.  
02 データ名A PIC X(8) VALUE 'U-ROLL '.  
02 データ名B PIC X(5).
```

(2) 機能

トランザクションをロールバックします。

CBLDCTRS('U-ROLL ')が正常終了すると、グローバルトランザクションは終了します。グローバルトランザクションの範囲外からは、SPP をトランザクションとして実行できません。

(3) UAP で値を設定するデータ領域

- データ名 A
非連鎖モードのロールバックを示す要求コードを「VALUE 'U-ROLL△△」と設定します。
- データ名 C
CBLDCCLS('CLTIN '), または CBLDCCLS('EXCLTIN ')で受け取ったクライアント ID を指定します。

(4) 値が返されるデータ領域

- データ名 B
ステータスコードが、5 けたの数字で返されます。

(5) ステータスコード

ステータスコード	意味
00000	正常終了しました。
02501	要求コード（データ名 A）に指定した値が間違っています。
02502	誤ったコンテキストからプログラムを呼び出しています。
02504	メモリ不足が発生しました。
02506	ネットワーク障害が発生しました。
02507	CBLDCTRS('U-ROLL ')の処理時間で時間切れ（タイムアウト）が発生しました。
02515	OpenTP1 が起動されていません。

ステータスコード	意味
02517	トランザクションプロセス内でメモリ不足が発生しました。
02518	システムエラーが発生しました。
02542	常設コネクションが解放されました。
02544	データ名 C に指定したクライアント ID は CBLDCCLS('CLTIN '), または CBLDCCLS('EXCLTIN ') で受け取ったクライアント ID と異なっています。
03403	ヒューリスティック決定によって、一部、またはすべてのトランザクションブランチはロールバックされました。詳細は、メッセージログファイルを参照してください。このステータスコードが戻ったあと、グローバルトランザクションの範囲外となります。
03404	ヒューリスティック決定でトランザクションが完了しましたが、障害のため結果がわかりません。詳細は、メッセージログファイルを参照してください。このステータスコードが戻ったあと、グローバルトランザクションの範囲外となります。

(6) 注意事項

トランザクションをロールバックしてから CUP のプロセスを終了させるときは、必ず CBLDCTRS('U-ROLL ') を実行してください。

6.5.6 CBLDCTRS('INFO ') – 現在のトランザクションに関する情報の報告

(1) 形式

(a) マルチスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCTRS' USING 一意名1
```

■ DATA DIVISION の指定

```
01 一意名1.
02 データ名A PIC X(8) VALUE 'INFO '.
02 データ名B PIC X(5).
02 FILLER PIC X(1).
02 データ名C.
03 データD PIC S9(4) COMP VALUE ZERO.
02 データ名E PIC 9(9) COMP-X.
```

(b) シングルスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCTRN' USING 一意名1
```

■ DATA DIVISION の指定

```
01 一意名1.  
02 データ名A PIC X(8) VALUE 'INFO ' .  
02 データ名B PIC X(5).  
02 FILLER PIC X(1).  
02 データ名C.  
03 データD PIC S9(4) COMP VALUE ZERO.
```

(2) 機能

CBLDCTRS('INFO ')を発行した CUP が現在トランザクションとして稼働しているかどうかを報告します。

(3) UAP で値を設定するデータ領域

- データ名 A

現在のトランザクションに関する情報の報告を示す要求コードを「VALUE 'INFO△△△△」と設定します。

- データ名 D

0を設定します。

- データ名 E

CBLDCCLS('CLTIN '), または CBLDCCLS('EXCLTIN ')で受け取ったクライアント ID を指定します。

(4) 値が返されるデータ領域

- データ名 B

ステータスコードが、5 けたの数字で返されます。

(5) ステータスコード

ステータスコード	意味
00001	CBLDCTRS('INFO ')を実行した CUP のプロセスは、トランザクションの範囲内にあります。
00000	CBLDCTRS('INFO ')を実行した CUP のプロセスは、トランザクションの範囲外にあります。
02501	要求コード (データ名 A) に指定した値が間違っています。
02544	データ名 E に指定したクライアント ID は CBLDCCLS('CLTIN '), または CBLDCCLS('EXCLTIN ')で受け取ったクライアント ID と異なります。

6.5.7 CBLDCCLS('GETTRNID') – 現在のトランザクションに関する識別子の取得

(1) 形式

(a) マルチスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCCLS' USING 一意名1
```

■ DATA DIVISION の指定

```
01 一意名1.  
02 データ名A PIC X(8) VALUE 'GETTRNID'.  
02 データ名B PIC X(5).  
02 FILLER PIC X(3).  
02 データ名C PIC X(17).  
02 データ名D PIC X(17).  
02 FILLER PIC X(2).  
02 データ名E PIC 9(9) COMP-X.
```

(b) シングルスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCCLT' USING 一意名1
```

■ DATA DIVISION の指定

```
01 一意名1.  
02 データ名A PIC X(8) VALUE 'GETTRNID'.  
02 データ名B PIC X(5).  
02 FILLER PIC X(3).  
02 データ名C PIC X(17).  
02 データ名D PIC X(17).
```

(2) 機能

現在のトランザクショングローバル識別子、およびトランザクションブランチ識別子を取得します。

この識別子は、次に示すプログラムを発行してトランザクションが起動されたときに、OpenTP1 が割り当てたものです。

- CBLDCTRS('BEGIN ')
- CBLDCTRS('C-COMMIT')
- CBLDCTRS('C-ROLL ')

(3) UAP で値を設定するデータ領域

- データ名 A

現在のトランザクションに関する識別子の取得を示す要求コードを「VALUE 'GETTRNID」 と設定します。

- データ名 E

CBLDCCLS('CLTIN '), または CBLDCCLS('EXCLTIN ') で受け取ったクライアント ID を指定します。

(4) 値が返されるデータ領域

- データ名 B

ステータスコードが、5 けたの数字で返されます。

- データ名 C

トランザクショングローバル識別子が返されます。

- データ名 D

トランザクションブランチ識別子が返されます。

(5) ステータスコード

ステータスコード	意味
00000	正常終了しました。
02501	データ名に指定した値が間違っています。要求コード (データ名 A) が間違っている場合も含まれます。
02502	誤ったコンテキストからプログラムを呼び出しています。
02504	メモリ不足が発生しました。
02544	データ名 E に指定したクライアント ID は CBLDCCLS('CLTIN '), または CBLDCCLS('EXCLTIN ') で受け取ったクライアント ID と異なります。

6.6 TCP/IP 通信機能

6.6.1 CBLDCCLS('SEND ') – メッセージの送信

(1) 形式

(a) マルチスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCCLS' USING 一意名1
```

■ DATA DIVISION の指定

```
01 一意名1.  
02 データ名A PIC X(8) VALUE 'SEND' .  
02 データ名B PIC X(5).  
02 FILLER PIC X(3).  
02 データ名C PIC S9(9) COMP VALUE ZERO.  
02 データ名D PIC S9(9) COMP.  
02 データ名E PIC X(64).  
02 データ名F PIC 9(4) COMP.  
02 FILLER PIC X(2).  
02 データ名H PIC 9(9) COMP-X.  
02 データ名I PIC X(n).
```

(b) シングルスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCCLT' USING 一意名1
```

■ DATA DIVISION の指定

```
01 一意名1.  
02 データ名A PIC X(8) VALUE 'SEND' .  
02 データ名B PIC X(5).  
02 FILLER PIC X(3).  
02 データ名C PIC S9(9) COMP VALUE ZERO.  
02 データ名D PIC S9(9) COMP.  
02 データ名E PIC X(64).  
02 データ名F PIC 9(4) COMP.  
02 FILLER PIC X(2).  
02 データ名G PIC 9(9) COMP.  
02 データ名I PIC X(n).
```


(2) 機能

MHP へメッセージを送信します。

CBLDCCLS('SEND ')を実行する場合、データ名 C に 4 または 16 を指定した CBLDCRPS('OPEN ')を、あらかじめ実行しておく必要があります。

(3) UAP で値を設定するデータ領域

- データ名 A
メッセージの送信を示す要求コードを「VALUE 'SEND△△△△」と設定します。
- データ名 C
メッセージを送信後に、接続を解放するかどうかを指定します。
0：メッセージ送信後、接続を解放しません。
1：メッセージ送信後、接続を解放します。
0 を指定した場合、CBLDCRPS('CLOSE ')を実行するまで接続を解放しません。ただし、障害時は除きます。
- データ名 D
送信するメッセージの長さを設定します。
- データ名 E
接続が確立されていない場合、接続するノードのホスト名を指定します。文字列の最後は空白文字を指定してください。ホスト名として、10 進ドット記法の IP アドレスを指定することもできます。先頭に空白を設定すると、CBLDCRPS('OPEN ')を実行したときに取得したクライアント環境定義 DCSNDHOST の内容を参照します。
- データ名 F
接続が確立されていない場合、接続を確立して接続するノードのポート番号を指定します。
0 を指定すると、CBLDCRPS('OPEN ')を実行したときに取得したクライアント環境定義 DCSNDPORT の内容を参照します。
- データ名 H
CBLDCCLS('CLTIN '), または CBLDCCLS('EXCLTIN ')で受け取ったクライアント ID を指定します。
- データ名 I
送信するメッセージを格納する領域を指定します。データ名 D で指定する長さ以上の領域を用意してください。

(4) 値が返されるデータ領域

- データ名 B

ステータスコードが、5 けたの数字で返されます。

- データ名 G

OpenTP1 で使用する領域です。

(5) ステータスコード

ステータスコード	意味
00000	正常終了しました。
02501	データ名に指定した値が間違っています。要求コード (データ名 A) が間違っている場合も含まれます。
02502	次のどちらかの要因が考えられます。 <ul style="list-style-type: none">• CBLDCRPS('OPEN ')が実行されていません。• CBLDCRPS('OPEN ')は実行されていますが、データ名 C に 4 または 16 を指定していません。
02504	メモリ不足が発生しました。
02506	ネットワーク障害が発生しました。
02507	コネクション確立要求時にタイムアウトになりました。
02518	システムエラーが発生しました。
02538	資源不足が発生しました。
02539	ホスト名が誤っています。または、データ名 E、および DCSNDHOST の両方にホスト名が指定されていません。
02541	相手システムに対するコネクションの確立要求が拒絶されました。
02544	データ名 H に指定したクライアント ID は CBLDCCLS('CLTIN '), または CBLDCCLS('EXCLTIN ')で受け取ったクライアント ID と異なります。
02547	OS が自動的に割り当てるポート番号が不足しています。

(6) 注意事項

メッセージ送信時に相手システムからコネクションが解放された場合、送信するメッセージ長によっては、コネクションの解放を検知できないことがあります。この場合は、CBLDCCLS('SEND ')の次以降に発行する要求文で検知することがあります。CUP を作成するときは、このことを考慮してください。

6.6.2 CBLDCCLS('EXSEND ') – メッセージの送信（ホスト名長の拡張時）

(1) 形式

(a) マルチスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCCLS' USING 一意名1 一意名2 一意名3
```

■ DATA DIVISION の指定

```
01 一意名1.  
  02 データ名A PIC X(8) VALUE 'EXSEND '.  
  02 データ名B PIC X(5).  
  02 FILLER PIC X(3).  
  02 データ名C PIC S9(9) COMP VALUE ZERO.  
  02 データ名D PIC S9(4) COMP.  
  02 FILLER PIC X(2).  
  02 データ名E PIC X(n).  
01 一意名2.  
  02 データ名F PIC S9(9) COMP.  
  02 データ名G PIC X(n).  
01 一意名3.  
  02 データ名H PIC 9(9) COMP-X.
```

(b) シングルスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCCLT' USING 一意名1 一意名2
```

■ DATA DIVISION の指定

```
01 一意名1.  
  02 データ名A PIC X(8) VALUE 'EXSEND '.  
  02 データ名B PIC X(5).  
  02 FILLER PIC X(3).  
  02 データ名C PIC S9(9) COMP VALUE ZERO.  
  02 データ名D PIC 9(4) COMP.  
  02 FILLER PIC X(2).  
  02 データ名E PIC X(n).  
01 一意名2.  
  02 データ名F PIC S9(9) COMP.  
  02 データ名G PIC X(n).
```

(2) 機能

MHP へメッセージを送信します。

CBLDCCLS('EXSEND ')を実行する場合、データ名 C に 4 または 16 を指定した CBLDCRPS('OPEN ')を、あらかじめ実行しておく必要があります。

ホスト名長の拡張機能を使用している場合、この関数を使用してください。

(3) UAP で値を設定するデータ領域

- データ名 A

メッセージの送信を示す要求コードを「VALUE 'EXSEND△△」と設定します。

- データ名 C

メッセージを送信後に、接続を解放するかどうかを指定します。

0：メッセージ送信後、接続を解放しません。

1：メッセージ送信後、接続を解放します。

0 を指定した場合、CBLDCRPS('CLOSE ')を実行するまで接続を解放しません。ただし、障害時は除きます。

- データ名 D

接続が確立されていない場合、接続を確立して接続するノードのポート番号を指定します。

0 を指定すると、CBLDCRPS('OPEN ')を実行したときに取得したクライアント環境定義 DCSNDPORT の内容を参照します。

- データ名 E

接続が確立されていない場合、接続するノードのホスト名を指定します。ホスト名として指定できる長さは、63 文字*までです。文字列の最後に空白文字を指定してください。

先頭に空白を指定すると、CBLDCRPS('OPEN ')を実行したときに取得したクライアント環境定義 DCSNDHOST の内容を参照します。

ホスト名として、10 進ドット記法の IP アドレスを指定することもできます。

注※

クライアント環境定義 DCCLTOPTION に 00000008 を指定した場合、ホスト名として指定できる長さは 255 文字までとなります。

- データ名 F

送信するメッセージの長さを設定します。

- データ名 G

送信するメッセージを格納する領域を指定します。データ名 F で指定する長さ以上の領域を用意してください。

- データ名 H

CBLDCCLS('CLTIN '), または CBLDCCLS('EXCLTIN ')で受け取ったクライアント ID を指定します。

(4) 値が返されるデータ領域

- データ名 B

ステータスコードが、5 けたの数字で返されます。

(5) ステータスコード

ステータスコード	意味
00000	正常終了しました。
02501	データ名に指定した値が間違っています。要求コード（データ名 A）が間違っている場合も含まれます。
02502	次のどちらかの要因が考えられます。 <ul style="list-style-type: none">CBLDCRPS('OPEN')が実行されていません。CBLDCRPS('OPEN')は実行されていますが、データ名 C に 4 または 16 を指定していません。
02504	メモリ不足が発生しました。
02506	ネットワーク障害が発生しました。
02507	コネクション確立要求時にタイムアウトになりました。
02518	システムエラーが発生しました。
02538	資源不足が発生しました。
02539	ホスト名が誤っています。または、データ名 E、および DCSNDHOST の両方にホスト名が指定されていません。
02541	相手システムに対するコネクションの確立要求が拒絶されました。
02544	データ名 H に指定したクライアント ID は CBLDCCLS('CLTIN'), または CBLDCCLS('EXCLTIN') で受け取ったクライアント ID と異なります。
02547	OS が自動的に割り当てるポート番号が不足しています。

(6) 注意事項

メッセージ送信時に相手システムからコネクションが解放された場合、送信するメッセージ長によっては、コネクションの解放を検知できないことがあります。この場合は、CBLDCCLS('EXSEND')の次以降に発行する要求文で検知することがあります。CUP を作成するときは、このことを考慮してください。

6.6.3 CBLDCCLS('RECEIVE ') – メッセージの受信

(1) 形式

(a) マルチスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCCLS' USING 一意名1
```

■ DATA DIVISION の指定

```
01 一意名1.  
02 データ名A PIC X(8) VALUE 'RECEIVE'.  
02 データ名B PIC X(5).  
02 FILLER PIC X(3).  
02 データ名C PIC S9(9) COMP VALUE ZERO.  
02 データ名D PIC S9(9) COMP.  
02 データ名E PIC S9(9) COMP.  
02 データ名G PIC 9(9) COMP-X.  
02 データ名H PIC X(n).
```

(b) シングルスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCCLT' USING 一意名1
```

■ DATA DIVISION の指定

```
01 一意名1.  
02 データ名A PIC X(8) VALUE 'RECEIVE'.  
02 データ名B PIC X(5).  
02 FILLER PIC X(3).  
02 データ名C PIC S9(9) COMP VALUE ZERO.  
02 データ名D PIC S9(9) COMP.  
02 データ名E PIC S9(9) COMP.  
02 データ名F PIC 9(9) COMP.  
02 データ名H PIC X(n).
```

(2) 機能

MHP が送信したメッセージを受信します。

CBLDCCLS('RECEIVE ')を実行する場合、データ名 C に 8 または 16 を指定した CBLDCRPS('OPEN ')を、あらかじめ実行しておく必要があります。

(3) UAP で値を設定するデータ領域

- データ名 A
メッセージの受信を示す要求コードを「VALUE 'RECEIVE△」と設定します。
- データ名 C
メッセージを受信後に、コネクションを解放するかどうかを指定します。
0：メッセージを受信後、コネクションを解放しません。
2：メッセージを受信後、コネクションを解放します。
0 を指定した場合、CBLDCRPS('CLOSE ')を実行するまでコネクションを解放しません。ただし、障害時を除きます。
- データ名 D
受信するメッセージの長さを指定します。
- データ名 E
メッセージ受信時の最大待ち時間（秒）を指定します。-1 から 65535 の整数を指定してください。
-1 を指定した場合は、メッセージを受信するまで無制限に待ちます。
0 を指定した場合は、メッセージの受信を待ちません。受信するメッセージがなかった場合は、02507 でエラーリターンします。
1 から 65535 を設定した場合は、指定した秒数だけメッセージの受信を待ちます。指定した秒数を過ぎてもメッセージを受信できない場合は、02507 でエラーリターンします。
- データ名 F
OpenTP1 で使用する領域です。
- データ名 G
CBLDCCLS('CLTIN '), または CBLDCCLS('EXCLTIN ')で受け取ったクライアント ID を指定します。
- データ名 H
受信したメッセージを格納する領域です。データ名 D で設定する長さ以上の領域を用意してください。

(4) 値が返されるデータ領域

- データ名 B
ステータスコードが、5 けたの数字で返されます。
- データ名 H
受信したメッセージが返されます。

(5) ステータスコード

ステータスコード	意味
00000	正常終了しました。

ステータスコード	意味
02501	データ名に指定した値が間違っています。要求コード（データ名 A）が間違っている場合も含まれます。
02502	次のどちらかの要因が考えられます。 <ul style="list-style-type: none"> • CBLDCRPS('OPEN')が実行されていません。 • CBLDCRPS('OPEN')は実行されていますが、CBLDCRPS('OPEN')のデータ名 C に 8 または 16 を指定していません。
02504	メモリ不足が発生しました。
02506	ネットワーク障害が発生しました。
02507	メッセージの受信時にタイムアウトになりました。
02518	システムエラーが発生しました。
02538	資源不足が発生しました。
02542	相手システムからの接続が解放されました。
02544	データ名 G に指定したクライアント ID は CBLDCCLS('CLTIN'), または CBLDCCLS('EXCLTIN') で受け取ったクライアント ID と異なります。

(6) 注意事項

- CBLDCCLS('RECEIVE')は、次に示す場合だけ、CUP に制御を戻します。
 - MHP からデータ名 D で指定した長さ分のメッセージを受信した場合（指定した長さより短いメッセージを受信しても CUP に制御は戻しません）
 - MHP からのメッセージ受信時に、時間切れ（タイムアウト）が発生した場合
 - MHP から接続が解放された場合
 - ネットワーク障害が発生した場合
- CBLDCCLS('RECEIVE')発行時に、MHP から接続が解放された場合、02542 でエラーリターンします。

6.6.4 CBLDCCLS('RECEIVE2') – メッセージの受信 (障害時メッセージ受信)

(1) 形式

(a) マルチスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCCLS' USING 一意名1
```

■ DATA DIVISION の指定

```
01 一意名1.  
02 データ名A PIC X(8) VALUE 'RECEIVE2'.  
02 データ名B PIC X(5).  
02 FILLER PIC X(3).  
02 データ名C PIC S9(9) COMP VALUE ZERO.  
02 データ名D PIC S9(9) COMP.  
02 データ名E PIC S9(9) COMP.  
02 データ名G PIC 9(9) COMP-X.  
02 データ名H PIC X(n).
```

(b) シングルスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCCLT' USING 一意名1
```

■ DATA DIVISION の指定

```
01 一意名1.  
02 データ名A PIC X(8) VALUE 'RECEIVE2'.  
02 データ名B PIC X(5).  
02 FILLER PIC X(3).  
02 データ名C PIC S9(9) COMP VALUE ZERO.  
02 データ名D PIC S9(9) COMP.  
02 データ名E PIC S9(9) COMP.  
02 データ名F PIC 9(9) COMP.  
02 データ名H PIC X(n).
```

(2) 機能

MHP が送信したメッセージを受信します。

CBLDCCLS('RECEIVE2')を実行する場合、データ名 C に 8 または 16 を指定した CBLDCRPS('OPEN')を、あらかじめ実行しておく必要があります。

(3) UAP で値を設定するデータ領域

- データ名 A
メッセージの受信を示す要求コードを「VALUE 'RECEIVE2」 と設定します。
- データ名 C
メッセージを受信後に、コネクションを解放するかどうかを指定します。
0：メッセージを受信後、コネクションを解放しません。
2：メッセージを受信後、コネクションを解放します。
0 を指定した場合、CBLDCRPS('CLOSE ')を実行するまでコネクションを解放しません。ただし、障害時を除きます。
- データ名 D
受信するメッセージの長さを指定します。
- データ名 E
メッセージ受信時の最大待ち時間（秒）を指定します。-1 から 65535 の整数を指定してください。
-1 を指定した場合
メッセージを受信するまで無制限に待ちます。
0 を指定した場合
メッセージの受信を待ちません。受信するメッセージがなかった場合は、02507 でエラーリターンします。
1 から 65535 を指定した場合
指定した秒数だけメッセージの受信を待ちます。指定した秒数を過ぎてもメッセージを受信できない場合は、02507 でエラーリターンします。
- データ名 F
OpenTP1 で使用する領域です。
- データ名 G
CBLDCCLS('CLTIN '), または CBLDCCLS('EXCLTIN ')で受け取ったクライアント ID を指定します。
- データ名 H
受信したメッセージを格納する領域です。データ名 D で設定する長さ以上の領域を用意してください。

(4) 値が返されるデータ領域

- データ名 B
ステータスコードが、5 けたの数字で返されます。
- データ名 D
受信したメッセージ長が返されます。
- データ名 H

受信したメッセージが返されます。

(5) ステータスコード

ステータスコード	意味
00000	正常終了しました。
02501	データ名に指定した値が間違っています。要求コード（データ名 A）が間違っている場合も含まれます。
02502	次のどちらかの要因が考えられます。 <ul style="list-style-type: none">• CBLDCRPS('OPEN')が実行されていません。• CBLDCRPS('OPEN')は実行されていますが、CBLDCRPS('OPEN')のデータ名 C に 8 または 16 を指定していません。
02504	メモリ不足が発生しました。
02506	ネットワーク障害が発生しました。
02507	メッセージの受信時にタイムアウトになりました。
02518	システムエラーが発生しました。
02538	資源不足が発生しました。
02542	相手システムから接続が解放されました。
02544	データ名 G に指定したクライアント ID は CBLDCCLS('CLTIN'), または CBLDCCLS('EXCLTIN') で受け取ったクライアント ID と異なります。

(6) 注意事項

- CBLDCCLS('RECEIVE2')は、次に示す場合だけ、CUP に制御を戻します。
 - MHP からデータ D で指定した長さ分のメッセージを受信した場合
指定した長さより短いメッセージを受信しても CUP に制御は戻しません
 - MHP からのメッセージ受信時に、時間切れ（タイムアウト）が発生した場合
 - MHP から接続が解放された場合
 - ネットワーク障害が発生した場合
- CBLDCCLS('RECEIVE2')発行時に、MHP から接続が解放された場合、02542 でエラーリターンします。

6.6.5 CBLDCCLS('ASMSSEND ') – 組み立てメッセージの送信

(1) 形式

(a) マルチスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCCLS' USING 一意名1 一意名2 一意名3
```

■ DATA DIVISION の指定

```
01 一意名1.  
02 データ名A PIC X(8) VALUE 'ASMSSEND'.  
02 データ名B PIC X(5).  
02 FILLER PIC X(3).  
02 データ名C PIC S9(9) COMP VALUE ZERO.  
02 データ名D PIC 9(4) COMP.  
02 FILLER PIC X(2).  
02 データ名E PIC S9(9) COMP.  
02 FILLER PIC X(4).  
02 データ名F PIC X(n).  
01 一意名2.  
02 データ名G PIC S9(9) COMP.  
02 データ名H PIC X(n).  
01 一意名3.  
02 データ名I PIC 9(9) COMP-X.
```

(b) シングルスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCCLT' USING 一意名1 一意名2
```

■ DATA DIVISION の指定

```
01 一意名1.  
02 データ名A PIC X(8) VALUE 'ASMSSEND'.  
02 データ名B PIC X(5).  
02 FILLER PIC X(3).  
02 データ名C PIC S9(9) COMP VALUE ZERO.  
02 データ名D PIC 9(4) COMP.  
02 FILLER PIC X(2).  
02 データ名E PIC S9(9) COMP.  
02 FILLER PIC X(4).  
02 データ名F PIC X(n).  
01 一意名2.  
02 データ名G PIC S9(9) COMP.  
02 データ名H PIC X(n).
```

(2) 機能

メッセージの組み立て機能を使用して、メッセージを送信します。この場合、4バイトのメッセージ情報を送信したあと、データ名 H に指定されたメッセージを送信します。相手システムとの接続が確立されていない場合は、データ名 F に指定されたホスト名とデータ名 D に指定されたポート番号を基に接続を確立し、メッセージを送信します。

また、クライアント環境定義 DCCLTDELIVERYCHECK に Y を指定した場合は、メッセージの送達確認機能を使用して、メッセージを送受信します。この場合、11バイトのメッセージ情報を送信したあと、データ名 H に指定されたメッセージを送信します。11バイトのメッセージ情報を受信したあと、CUP に制御を戻します。

CBLDCCLS('ASMSSEND ')を実行する場合、データ名 C に 4, または 16 を指定した CBLDCRPS('OPEN ')を、あらかじめ実行してください。

(3) UAP で値を設定するデータ領域

• データ名 A

メッセージの送信を示す要求コードを、「VALUE 'ASMSSEND△」と設定します。

• データ名 C

メッセージを送信後に、接続を解放するかどうかを指定します。

0:メッセージを送信後、接続を解放しません。CBLDCRPS('CLOSE ')を実行するまで、接続を解放しません。ただし、障害時を除きます。

1:メッセージを送信後、接続を解放します。メッセージの送達確認機能使用時は、メッセージ情報を受信したあと、接続を解放します。

• データ名 D

接続が確立されていない場合、接続を確立して接続するノードのポート番号を指定します。

0を指定すると、CBLDCRPS('OPEN ')を実行したときに取得したクライアント環境定義 DCSNDPORT の内容を参照します。

• データ名 E

メッセージの送達確認機能を使用するときに有効な引数です。応答専用データ受信時の最大待ち時間(秒)を指定します。-1 から 65535 の整数を設定してください。

-1 を指定した場合

応答専用データを受信するまで無制限に待ちます。

0 を指定した場合

応答専用データの受信を待ちません。受信する応答専用データがなかった場合は、02507 でエラーリターンします。

1 から 65535 を指定した場合

指定した秒数だけ応答専用データの受信を待ちます。指定した秒数を過ぎても応答専用データを受信できない場合は、02507 でエラーリターンします。

応答専用データが分割されて届いた場合は、11 バイトの応答専用データが届くまで受信処理を繰り返します。受信処理が発生すると、この引数に指定した最大待ち時間が毎回適用されます。この引数に指定した値を、クライアントの最大応答待ち時間として適用したい場合は、クライアント環境定義 DCCLTOPTION に、00000002 オプションを指定してください。

- **データ名 F**

コネクションが確立されていない場合、接続するノードのホスト名を指定します。ホスト名として指定できる長さは、63 文字※までです。文字列の最後に空白文字を指定してください。

先頭に空白を指定すると、CBLDCRPS('OPEN ')を実行したときに取得したクライアント環境定義 DCSNDHOST の内容を参照します。

ホスト名として、10 進ドット記法の IP アドレスを指定することもできます。

注※

クライアント環境定義 DCCLTOPTION に 00000008 を指定した場合、ホスト名として指定できる長さは 255 文字までとなります。

- **データ名 G**

送信するメッセージの長さを設定します。

- **データ名 H**

送信するメッセージを格納する領域を指定します。データ名 G で指定する長さ以上の領域を用意してください。

- **データ名 I**

CBLDCCLS('CLTIN '), または CBLDCCLS('EXCLTIN ')で受け取ったクライアント ID を指定します。

(4) 値が返されるデータ領域

- **データ名 B**

ステータスコードが、5 けたの数字で返されます。

(5) ステータスコード

ステータスコード	意味
00000	正常終了しました。
02501	引数に指定した値が誤っています。
02502	次のどちらかの要因が考えられます。 <ul style="list-style-type: none">• CBLDCRPS('OPEN ')が実行されていません。

ステータスコード	意味
02502	• CBLDCRPS('OPEN ')は実行されていますが、データ名 C に 4, または 16 を指定していません。
02504	メモリ不足が発生しました。
02506	ネットワーク障害が発生しました。コネクションは解放されます。
02507	コネクション確立要求時にタイムアウトになりました。または、メッセージの送達確認機能の使用時に、応答専用データの受信でタイムアウトになりました。コネクションは解放されます。
02518	システムエラーが発生しました。ネットワーク障害の場合、コネクションは解放されます。
02538	資源不足が発生しました。
02539	ホスト名が誤っています。または、データ名 F および DCSNDHOST の両方にホスト名が指定されていません。
02541	相手システムに対するコネクションの確立要求が拒絶されました。
02542	メッセージの送達確認機能の使用時に、相手システムからコネクションが解放されました。
02544	データ名 I に指定したクライアント ID は、CBLDCCLS('CLTIN '), または CBLDCCLS('EXCLTIN ')で受け取ったクライアント ID と異なっています。
02547	OS が自動的に割り当てるポート番号が不足しています。
02548	メッセージの送達確認機能の使用時に、不正なメッセージを受信しました。コネクションは解放されます。
02584	メッセージの送達確認機能の使用時に、メッセージが衝突しました。コネクションは解放されます。

(6) 注意事項

- メッセージ送信時に相手システムからコネクションが解放された場合、送信するメッセージ長によっては、コネクションの解放を検知できないことがあります。次に、使用する機能別に注意事項を示します。

メッセージ組み立て機能を使用する場合

CBLDCCLS('ASMSEND ')で検知できなかった場合は、次以降に発行する要求文で検知することができます。CUP を作成するときは、このことを考慮してください。

メッセージの送達確認機能を使用する場合

メッセージ送信時に検知できなかった場合、応答専用データ受信時に検知します。

- メッセージの組み立て機能および送達確認機能を使用する場合は、短いパケットで送受信するため、送信処理に時間が掛かることがあります。送信処理に時間が掛かる場合は、クライアント環境定義 DCCLTTCPNODELAY に Y を指定してください。

6.6.6 CBLDCCLS('ASMRECV ') – 組み立てメッセージの受信

(1) 形式

(a) マルチスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCCLS' USING 一意名1 一意名2 一意名3
```

■ DATA DIVISION の指定

```
01 一意名1.  
  02 データ名A PIC X(8) VALUE 'ASMRECV'.  
  02 データ名B PIC X(5).  
  02 FILLER PIC X(3).  
  02 データ名C PIC S9(9) COMP VALUE ZERO.  
  02 データ名D PIC S9(9) COMP.  
01 一意名2.  
  02 データ名E PIC S9(9) COMP.  
  02 データ名F PIC X(n).  
01 一意名3.  
  02 データ名G PIC 9(9) COMP-X.
```

(b) シングルスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCCLT' USING 一意名1 一意名2
```

■ DATA DIVISION の指定

```
01 一意名1.  
  02 データ名A PIC X(8) VALUE 'ASMRECV'.  
  02 データ名B PIC X(5).  
  02 FILLER PIC X(3).  
  02 データ名C PIC S9(9) COMP VALUE ZERO.  
  02 データ名D PIC S9(9) COMP.  
01 一意名2.  
  02 データ名E PIC S9(9) COMP.  
  02 データ名F PIC X(n).
```

(2) 機能

メッセージの組み立て機能を使用して、メッセージを受信します。この場合、4バイトのメッセージ情報を受信したあと、メッセージ情報に設定されたメッセージ長分のメッセージを受信し、データ名 F に格納します。ただし、メッセージ情報は、データ名 F には格納しません。データ名 E には、受信したメッセージ長を格納します。データ名 E に格納するメッセージ長には、メッセージ情報の長さは含みません。

また、クライアント環境定義 DCCLTDELIVERYCHECK に Y を指定した場合は、メッセージの送達確認機能を使用して、メッセージを送受信します。この場合、11 バイトのメッセージ情報を受信した後、メッセージ情報に設定されたメッセージ長分のメッセージを受信し、データ名 F に格納します。ただし、メッセージ情報は、データ名 F には格納しません。データ名 E には、受信したメッセージ長を格納します。データ名 E に格納するメッセージ長には、メッセージ情報の長さは含みません。受信したメッセージ情報に「応答要求」が設定されていた場合は、11 バイトのメッセージ情報を送信したあと、CUP に制御を戻します。

CBLDCCLS('ASMRECV ')を実行する場合、データ名 C に 8、または 16 を指定した CBLDCRPS('OPEN ')を、あらかじめ実行しておく必要があります。

(3) UAP で値を設定するデータ領域

- データ名 A

メッセージの受信を示す要求コードを「VALUE 'ASMRECV△」と設定します。

- データ名 C

メッセージを受信したあと、コネクションを解放するかどうかを指定します。

0：メッセージを受信したあと、CBLDCRPS('CLOSE ')実行時に、コネクションを解放します。ただし、障害時を除きます。

2：メッセージを受信したあと、コネクションを解放します。メッセージの送達確認機能の使用時に、受信したメッセージ情報に「応答要求」が設定されていた場合は、メッセージ情報を送信後、コネクションを解放します。

- データ名 D

メッセージ受信時の最大待ち時間(秒)を指定します。-1 から 65535 の整数を設定してください。

-1 を指定した場合

メッセージを受信するまで無制限に待ちます。

0 を指定した場合

メッセージの受信を待ちません。受信するメッセージがなかった場合は、02507 でエラーリターンします。

1 から 65535 を指定した場合

指定した秒数だけメッセージの受信を待ちます。指定した秒数を過ぎてもメッセージを受信できない場合は、02507 でエラーリターンします。

メッセージが分割されて届いた場合、メッセージ長分のメッセージが届くまで受信処理を繰り返し行います。受信処理が発生すると、この引数に指定した値が毎回適用されます。この引数に指定した値をクライアントの最大応答待ち時間として適用したい場合は、クライアント環境定義 DCCLTOPTION に、00000002 オプションを指定してください。

- データ名 E

受信するメッセージの長さを指定します。

- データ名 F

受信したメッセージを格納する領域です。データ名 E で設定する長さ以上の領域を用意してください。

- データ名 G

CBLDCCLS('CLTIN '), または CBLDCCLS('EXCLTIN ') で受け取ったクライアント ID を指定します。

(4) 値が返されるデータ領域

- データ名 B

ステータスコードが、5 けたの数字で返されます。

- データ名 E

受信したメッセージの長さが返されます。メッセージ情報の長さは、含みません。

タイムアウト発生時は、タイムアウトまでに受信できたメッセージの長さが格納されます。

- データ名 F

受信したメッセージが返されます。メッセージ情報は、含みません。

タイムアウト発生時は、タイムアウトまでに受信できたメッセージが格納されます。

(5) ステータスコード

ステータスコード	意味
00000	正常に終了しました。
02501	引数に指定した値が誤っています。
02502	次のどちらかの要因が考えられます。 <ul style="list-style-type: none">• CBLDCRPS('OPEN ')が実行されていません。• CBLDCRPS('OPEN ')は実行されていますが、データ名 C に 8, または 16 を指定していません。
02504	メモリ不足が発生しました。
02506	ネットワーク障害が発生しました。接続は解放されます。
02507	メッセージの受信時にタイムアウトになりました。接続は解放されます。
02518	システムエラーが発生しました。ネットワーク障害の場合、接続は解放されます。
02538	資源不足が発生しました。
02542	相手システムから接続が解放されました。
02544	データ名 G に指定したクライアント ID は、CBLDCCLS('CLTIN '), または CBLDCCLS('EXCLTIN ') で受け取ったクライアント ID と異なります。
02546	CUP で用意した領域が小さく、相手システムからのメッセージを受信することができません。接続は解放されます。
02548	不正なメッセージを受信しました。接続は解放されます。

(6) 注意事項

- CBLDCCLS('ASMRECV ')は、次に示す場合だけ CUP に制御を戻します。
 - メッセージ情報に設定されたメッセージ長分のメッセージを受信した場合
 - ネットワーク障害が発生した場合
 - メッセージ受信時に、タイムアウトが発生した場合
 - 相手システムからコネクションが解放された場合
 - メッセージを格納する領域（データ名 F）が小さく、通信相手が送信するメッセージを格納することができない場合
 - 不正なメッセージを受信した場合
- メッセージの組み立て機能およびメッセージの送達確認機能を使用する場合は、短いパケットで送受信するため、送信処理に時間が掛かることがあります。送信処理に時間が掛かる場合は、クライアント環境定義 DCCLTTCPNODELAY に Y を指定してください。

6.7 サーバからの一方通知受信機能

6.7.1 CBLDCCLS('NOTIFY') – 一方通知メッセージの受信

(1) 形式

(a) マルチスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCCLS' USING 一意名1 一意名2
```

■ DATA DIVISION の指定

```
01 一意名1.  
  02 データ名A PIC X(8) VALUE 'NOTIFY'.  
  02 データ名B PIC X(5).  
  02 FILLER PIC X(3).  
  02 データ名C PIC S9(9) COMP VALUE ZERO.  
  02 データ名D PIC 9(4) COMP.  
  02 FILLER PIC X(2).  
  02 データ名E PIC S9(9) COMP.  
  02 データ名F PIC X(64).  
  02 データ名G PIC X(8).  
  02 FILLER PIC 9(4) COMP.  
  02 FILLER PIC X(2).  
  02 データ名I PIC X(256).  
  
01 一意名2.  
  02 データ名J PIC S9(9) COMP.  
  02 データ名K PIC X(n).
```

(b) シングルスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCCLT' USING 一意名1 一意名2
```

■ DATA DIVISION の指定

```
01 一意名1.  
  02 データ名A PIC X(8) VALUE 'NOTIFY'.  
  02 データ名B PIC X(5).  
  02 FILLER PIC X(3).  
  02 データ名C PIC S9(9) COMP VALUE ZERO.  
  02 データ名D PIC 9(4) COMP.  
  02 FILLER PIC X(2).  
  02 データ名E PIC S9(9) COMP.  
  02 データ名F PIC X(64).
```

02 データ名G PIC X(8).

01 一意名2.

02 データ名J PIC S9(9) COMP.

02 データ名K PIC X(n).

(2) 機能

サーバ側の要求コード (CBLDCRPC('CLTSEND ')) によって通知されるメッセージを、データ名 E で指定した値まで待ち続けます。受信した時点で CUP に制御を戻し、ステータスコード、通知メッセージ、通知元サーバのホスト名、通知元サーバのノード識別子を返します。また、この要求コードの実行前に CBLDCCLS('CLTIN '), および CBLDCRPS('OPEN ') を実行しておく必要はありません。

(3) UAP で値を設定するデータ領域

• データ名 A

一方通知メッセージの受信を示す要求コードを「VALUE 'NOTIFY△△」と設定します。

• データ名 C

0 を指定します。

• データ名 D

クライアントのポート番号を指定します。5001 から 65535 の範囲で指定します。なお、同一マシン内で、複数のプロセス、または複数のスレッドを同時に実行する場合は、それぞれ異なるポート番号を指定してください。

• データ名 E

タイムアウト値 (秒) を指定します。0 から 65535 の範囲で指定します。

無限に待ち続ける場合は、0 を指定します。

• データ名 I

クライアント環境定義ファイルへのパス名を指定します。パス名には完全パス、またはカレントドライブ・ディレクトリからの相対パスが指定できます。パス名を指定した場合のファイルの読み込み順序を次に示します。

• TP1/Client/P の場合

クライアント環境定義ファイルの読み込み順序は次のとおりです。

- 1.Windows ディレクトリの BETRAN.INI ファイル
- 2.データ名 I に指定したクライアント環境定義ファイル

定義は、クライアント環境定義ファイルおよび BETRAN.INI ファイルのどちらのファイルに指定しても有効です。

両方のファイルに同じ定義を異なる値で指定した場合は、クライアント環境定義ファイルに指定した値が有効となります。

クライアント環境定義ファイルおよび BETRAN.INI ファイルのどちらにも指定がない場合は、デフォルト値で動作します。

- TP1/Client/W の場合

環境変数に指定されている定義は、すべて無効となります。データ名 I に指定したクライアント環境定義ファイルに指定されていない定義はデフォルト値で動作します。

また、データ名 I の先頭に空白を指定することでパス名を省略できます。省略時の動作を次に示します。

- TP1/Client/P の場合

Windows ディレクトリの BETRAN.INI ファイルをクライアント環境定義ファイルとして動作します。BETRAN.INI ファイルがない場合、または定義ファイルの内容が不正な場合はデフォルト値で動作します。

- TP1/Client/W の場合

環境変数の指定で動作します。環境変数が指定されていない場合は、デフォルト値で動作します。

データ名 I に指定したクライアント環境定義ファイルがない場合、または定義ファイルの内容が不正な場合の動作を次に示します。

- TP1/Client/P の場合

Windows ディレクトリの BETRAN.INI ファイルをクライアント環境定義ファイルとして動作します。BETRAN.INI ファイルがない場合、または定義ファイルの内容が不正な場合は、デフォルト値で動作します。

- TP1/Client/W の場合

デフォルト値で動作します。環境変数の指定は無効となります。

- **データ名 J**

サーバからの通知メッセージを格納する領域長（データ名 K の長さ）を指定します。0 から DCRPC_MAX_MESSAGE_SIZE* の範囲で指定します。

注※

クライアント環境定義 DCCLTRPCMAXMSGSIZE に 2 以上を指定した場合、DCRPC_MAX_MESSAGE_SIZE の値（1 メガバイト）ではなく、クライアント環境定義 DCCLTRPCMAXMSGSIZE に指定した値になります。

- **データ名 K**

サーバからの通知メッセージを格納する領域です。データ名 J で設定する長さ以上の領域を用意してください。

(4) 値が返されるデータ領域

- **データ名 B**

ステータスコードが、5 けたの数字で返されます。

- **データ名 F**

通知したサーバのホスト名が返されます。ホスト名への変換に失敗した場合、10 進ドット記法の IP アドレスが返されます。

- **データ名 G**

通知したサーバのノード識別子が返されます。ノード識別子のフォーマットは次のとおりです。

ノード識別子 (4バイト)	空白 (4バイト)
---------------	-----------

- データ名 J
サーバからの通知メッセージ長が返されます。
- データ名 K
サーバからの通知メッセージが返されます。

(5) ステータスコード

ステータスコード	意味
00000	正常終了しました。
02501	データ名に設定した値が間違っています。要求コード (データ名 A) が間違っている場合も含まれます。
02503	初期化に失敗しました。または、クライアント環境定義の指定に誤りがあります。
02504	必要なバッファが確保されませんでした。
02506	ネットワーク障害が発生しました。
02507	メッセージの受信時にタイムアウトになりました。
02518	システムエラーが発生しました。
02535	バージョン不一致が発生しました。
02546	受信したメッセージが、CUP で用意した領域に収まりません。収まり切らないメッセージは切り捨てました。データ名 F, データ名 G には、値を設定済みです。
02547	指定したポート番号は使用されています。
02548	不正なメッセージを受信しました。
02549	一方通知受信待ち状態が CBLDCCLS('CANCEL ')によって解除されました。引数データ名 F, データ名 J, およびデータ名 K には、値を設定済みです。

(6) 注意事項

同一マシン内で、複数のプロセス、または複数のスレッドを同時に実行する場合、データ名 D にはそれぞれ異なるポート番号を指定してください。また、データ名 D に指定できるポート番号でも、OS またはほかのプログラムが使用するポート番号は指定しないでください。指定した場合、応答データを正しく受信できないことがあります。なお、OS が使用するポート番号は、OS ごとに異なります。OS が使用するポート番号については、OS のマニュアルなどを参照してください。

6.7.2 CBLDCCLS('EXNACPT ') – 一方通知メッセージの受信（ホスト名長の拡張時）

(1) 形式

(a) マルチスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCCLS' USING 一意名1 一意名2 一意名3
```

■ DATA DIVISION の指定

```
01 一意名1.  
02 データ名A PIC X(8) VALUE 'EXNACPT'.  
02 データ名B PIC X(5).  
02 FILLER PIC X(3).  
02 データ名C PIC S9(9) COMP VALUE ZERO.  
02 データ名D PIC 9(4) COMP.  
02 FILLER PIC X(2).  
02 データ名E PIC S9(9) COMP.  
02 データ名F PIC X(8).  
02 データ名G PIC X(n).  
01 一意名2.  
02 データ名H PIC S9(9) COMP.  
02 データ名I PIC X(n).  
01 一意名3.  
02 FILLER PIC 9(9) COMP.  
02 FILLER PIC 9(4) COMP.  
02 FILLER PIC X(2).  
02 データ名J PIC X(n).
```

(b) シングルスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCCLT' USING 一意名1 一意名2
```

■ DATA DIVISION の指定

```
01 一意名1.  
02 データ名A PIC X(8) VALUE 'EXNACPT'.  
02 データ名B PIC X(5).  
02 FILLER PIC X(3).  
02 データ名C PIC S9(9) COMP VALUE ZERO.  
02 データ名D PIC 9(4) COMP.  
02 FILLER PIC X(2).  
02 データ名E PIC S9(9) COMP.  
02 データ名F PIC X(8).  
02 データ名G PIC X(n).  
01 一意名2.
```


(2) 機能

サーバ側の要求コード (CBLDCRPC('CLTSEND ')) によって通知されるメッセージを、データ名 E で指定した値まで待ち続けます。受信した時点で CUP に制御を戻し、ステータスコード、通知メッセージ、通知元サーバのホスト名、通知元サーバのノード識別子を返します。また、この要求コードの実行前に CBLDCCLS('CLTIN '), および CBLDCRPS('OPEN ') を実行しておく必要はありません。

ホスト名長の拡張機能を使用している場合、この関数を使用してください。

(3) UAP で値を設定するデータ領域

- データ名 A
一方通知メッセージの受信を示す要求コードを「VALUE 'EXNACPT△」と設定します。
- データ名 C
0 を指定します。
- データ名 D
クライアントのポート番号を指定します。5001 から 65535 の範囲で指定します。なお、同一マシン内で、複数のプロセス、または複数のスレッドを同時に実行する場合は、それぞれ異なるポート番号を指定してください。
- データ名 E
タイムアウト値 (秒) を指定します。0 から 65535 の範囲で指定します。
無限に待ち続ける場合は、0 を指定します。
- データ名 G
通知したサーバのホスト名を格納する 64 バイト*以上の領域を用意してください。

注※

クライアント環境定義 DCCLTOPTION に 00000008 を指定した場合、256 バイト以上の領域を用意してください。

- データ名 H
サーバからの通知メッセージを格納する領域長を指定します。0 からクライアント環境定義 DCRPCMAX_MESSAGE_SIZE*で指定した値の範囲で指定します。

注※

クライアント環境定義 DCCLTRPCMAXMSGSIZE に 2 以上を指定した場合、DCRPC_MAX_MESSAGE_SIZE の値 (1 メガバイト) ではなく、クライアント環境定義 DCCLTRPCMAXMSGSIZE に指定した値になります。

- データ名 I

サーバからの通知メッセージを格納する領域を指定します。データ名 H で設定する長さ以上の領域を用意してください。

• データ名 J

クライアント環境定義ファイルへのパス名を指定します。パス名には完全パス、またはカレントドライブ・ディレクトリからの相対パスが指定できます。パス名を指定した場合のファイルの読み込み順序を次に示します。

• TP1/Client/P の場合

クライアント環境定義ファイルの読み込み順序は次のとおりです。

- 1.Windows ディレクトリの BETRAN.INI ファイル
- 2.データ名 I に指定したクライアント環境定義ファイル

定義は、クライアント環境定義ファイルおよび BETRAN.INI ファイルのどちらのファイルに指定しても有効です。

両方のファイルに同じ定義を異なる値で指定した場合は、クライアント環境定義ファイルに指定した値が有効となります。

クライアント環境定義ファイルおよび BETRAN.INI ファイルのどちらにも指定がない場合は、デフォルト値で動作します。

• TP1/Client/W の場合

環境変数に指定されている定義は、すべて無効となります。データ名 J に指定したクライアント環境定義ファイルに指定されていない定義はデフォルト値で動作します。

また、データ名 J の先頭に空白を指定することでパス名を省略できます。省略時の動作を次に示します。

• TP1/Client/P の場合

Windows ディレクトリの BETRAN.INI ファイルをクライアント環境定義ファイルとして動作します。BETRAN.INI ファイルがない場合、または定義ファイルの内容が不正な場合はデフォルト値で動作します。

• TP1/Client/W の場合

環境変数の指定で動作します。環境変数が指定されていない場合は、デフォルト値で動作します。

データ名 J に指定したクライアント環境定義ファイルがない場合、または定義ファイルの内容が不正な場合の動作を次に示します。

• TP1/Client/P の場合

Windows ディレクトリの BETRAN.INI ファイルをクライアント環境定義ファイルとして動作します。BETRAN.INI ファイルがない場合、または定義ファイルの内容が不正な場合は、デフォルト値で動作します。

• TP1/Client/W の場合

デフォルト値で動作します。環境変数の指定は無効となります。

(4) 値が返されるデータ領域

• データ名 B

ステータスコードが、5 けたの数字で返されます。

- データ名 F

通知したサーバのノード識別子が返されます。ノード識別子のフォーマットは次のとおりです。

ノード識別子 (4バイト)	空白 (4バイト)
---------------	-----------

- データ名 G

通知したサーバのホスト名が返されます。ホスト名への変換に失敗した場合、10 進ドット記法の IP アドレスが返されます。

- データ名 H

サーバからの通知メッセージ長が返されます。

- データ名 I

サーバからの通知メッセージが返されます。

(5) ステータスコード

ステータスコード	意味
00000	正常終了しました。
02501	データ名に設定した値が間違っています。要求コード (データ名 A) が間違っている場合も含まれます。
02503	初期化に失敗しました。または、クライアント環境定義の指定に誤りがあります。
02504	必要なバッファが確保されませんでした。
02506	ネットワーク障害が発生しました。
02507	メッセージの受信時にタイムアウトになりました。
02518	システムエラーが発生しました。
02535	バージョン不一致が発生しました。
02546	受信したメッセージが、CUP で用意した領域に収まりません。収まり切らないメッセージは切り捨てました。データ名 F, データ名 G には、値を設定済みです。
02547	指定したポート番号は使用されています。
02548	不正なメッセージを受信しました。
02549	一方通知受信待ち状態が CBLDCCLS('CANCEL ')または CBLDCCLS('EXNCANCL')によって解除されました。引数データ名 G, データ名 H, およびデータ名 I には、値を設定済みです。

(6) 注意事項

同一マシン内で、複数のプロセス、または複数のスレッドを同時に実行する場合、データ名 D にはそれぞれ異なるポート番号を指定してください。また、データ名 D に指定できるポート番号でも、OS またはほかのプログラムが使用するポート番号は指定しないでください。指定した場合、応答データを正しく受信

できないことがあります。なお、OSが使用するポート番号は、OSごとに異なります。OSが使用するポート番号については、OSのマニュアルなどを参照してください。

6.7.3 CBLDCCLS('CANCEL') – 一方通知待ち状態のキャンセル

(1) 形式

(a) マルチスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCCLS' USING 一意名1 一意名2
```

■ DATA DIVISION の指定

```
01 一意名1.  
02 データ名A PIC X(8) VALUE 'CANCEL'.  
02 データ名B PIC X(5).  
02 FILLER PIC X(3).  
02 データ名C PIC S9(9) COMP VALUE ZERO.  
02 データ名D PIC 9(4) COMP.  
02 FILLER PIC X(2).  
02 データ名E PIC X(64).  
02 FILLER PIC 9(4) COMP.  
02 FILLER PIC X(2).  
02 データ名G PIC X(256).  
01 一意名2.  
02 データ名H PIC S9(9) COMP.  
02 データ名I PIC X(n).
```

(b) シングルスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCCLT' USING 一意名1 一意名2
```

■ DATA DIVISION の指定

```
01 一意名1.  
02 データ名A PIC X(8) VALUE 'CANCEL'.  
02 データ名B PIC X(5).  
02 FILLER PIC X(3).  
02 データ名C PIC S9(9) COMP VALUE ZERO.  
02 データ名D PIC 9(4) COMP.  
02 FILLER PIC X(2).  
02 データ名E PIC X(64).  
01 一意名2.  
02 データ名H PIC S9(9) COMP.  
02 データ名I PIC X(n).
```

(2) 機能

次のどれかの要求文を使用し、サーバからの一方通知待ち状態となっている CUP の、待ち状態を解除します。

- CBLDCCLS('NOTIFY ')
- CBLDCCLS('A-NOTIFY')
- CBLDCCLS('EXNACPT ')
- CBLDCCLS('EXNCACPT')

解除するときに、データ名 I に指定したメッセージを一方通知受信待ち状態の CUP に通知できます。

(3) UAP で値を設定するデータ領域

- データ名 A
一方通知待ち状態のキャンセルを示す要求コードを「VALUE 'CANCEL△△」と設定します。
- データ名 C
0 を指定します。
- データ名 D
一方通知受信要求時に指定したポート番号を指定します。5001 から 65535 の範囲で指定します。
- データ名 E
一方通知受信待ち状態の CUP が存在するホスト名を指定します。ホスト名として、10 進ドット記法の IP アドレスを指定することもできます。
- データ名 G
クライアント環境定義ファイルへのパス名を指定します。パス名には完全パス、またはカレントドライブ・ディレクトリからの相対パスが指定できます。パス名を指定した場合のファイルの読み込み順序を次に示します。
 - TP1/Client/P の場合
クライアント環境定義ファイルの読み込み順序は次のとおりです。
 - 1.Windows ディレクトリの BETRAN.INI ファイル
 - 2.データ名 I に指定したクライアント環境定義ファイル定義は、クライアント環境定義ファイルおよび BETRAN.INI ファイルのどちらのファイルに指定しても有効です。
両方のファイルに同じ定義を異なる値で指定した場合は、クライアント環境定義ファイルに指定した値が有効となります。
クライアント環境定義ファイルおよび BETRAN.INI ファイルのどちらにも指定がない場合は、デフォルト値で動作します。
 - TP1/Client/W の場合

環境変数に指定されている定義は、すべて無効となります。データ名 G に指定したクライアント環境定義ファイルに指定されていない定義はデフォルト値で動作します。

また、データ名 G の先頭に空白を指定することでパス名を省略できます。省略時の動作を次に示します。

- TP1/Client/P の場合

Windows ディレクトリの BETRAN.INI ファイルをクライアント環境定義ファイルとして動作します。BETRAN.INI ファイルがない場合、または定義ファイルの内容が不正な場合はデフォルト値で動作します。

- TP1/Client/W の場合

環境変数の指定で動作します。環境変数が指定されていない場合は、デフォルト値で動作します。

データ名 G に指定したクライアント環境定義ファイルがない場合、または定義ファイルの内容が不正な場合の動作を次に示します。

- TP1/Client/P の場合

Windows ディレクトリの BETRAN.INI ファイルをクライアント環境定義ファイルとして動作します。BETRAN.INI ファイルがない場合、または定義ファイルの内容が不正な場合は、デフォルト値で動作します。

- TP1/Client/W の場合

デフォルト値で動作します。環境変数の指定は無効となります。

- データ名 I

CUP に通知するメッセージを指定します。

- データ名 H

メッセージ長（データ名 I の長さ）を指定します。0 から DCRPC_MAX_MESSAGE_SIZE[※]の範囲で指定します。

0 を指定した場合は CUP にメッセージを通知しません。

注※

クライアント環境定義 DCCLTRPCMAXMSGSIZE に 2 以上を指定した場合、DCRPC_MAX_MESSAGE_SIZE の値（1 メガバイト）ではなく、クライアント環境定義 DCCLTRPCMAXMSGSIZE に指定した値になります。

(4) 値が返されるデータ領域

- データ名 B

ステータスコードが、5 けたの数字で返されます。

(5) ステータスコード

ステータスコード	意味
00000	正常終了しました。

ステータスコード	意味
02501	データ名に設定した値が間違っています。要求コード（データ名 A）が間違っている場合も含まれます。
02503	初期化に失敗しました。または、クライアント環境定義の指定に誤りがあります。
02504	必要なバッファが確保できませんでした。
02506	ネットワーク障害が発生しました。
02514	CUP が一方通知受信待ち状態ではありません。
02518	システムエラーが発生しました。
02539	ホスト名が不正です。
02547	OS が自動的に割り当てるポート番号が不足しています。

6.7.4 CBLDCCLS('EXNCANCL') – 一方通知待ち状態のキャンセル（ホスト名長の拡張時）

(1) 形式

(a) マルチスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCCLS' USING 一意名1 一意名2 一意名3
```

■ DATA DIVISION の指定

```
01 一意名1.
  02 データ名A PIC X(8) VALUE 'EXCANCEL'.
  02 データ名B PIC X(5).
  02 FILLER PIC X(3).
  02 データ名C PIC S9(9) COMP VALUE ZERO.
  02 データ名D PIC 9(4) COMP.
  02 FILLER PIC X(2).
  02 データ名E PIC X(n).
01 一意名2.
  02 データ名F PIC S9(9) COMP.
  02 データ名G PIC X(n).
01 一意名3.
  02 FILLER PIC 9(9) COMP.
  02 FILLER PIC 9(4) COMP.
  02 FILLER PIC X(2).
  02 データ名H PIC X(n).
```

(b) シングルスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCCLT' USING 一意名1 一意名2
```

■ DATA DIVISION の指定

```
01 一意名1.  
  02 データ名A PIC X(8) VALUE 'EXCANCEL'.  
  02 データ名B PIC X(5).  
  02 FILLER PIC X(3).  
  02 データ名C PIC S9(9) COMP VALUE ZERO.  
  02 データ名D PIC 9(4) COMP.  
  02 FILLER PIC X(2).  
  02 データ名E PIC X(n).  
01 一意名2.  
  02 データ名F PIC S9(9) COMP.  
  02 データ名G PIC X(n).
```

(2) 機能

次のどれかの要求文を使用し、サーバからの一方通知待ち状態となっている CUP の、待ち状態を解除します。

- CBLDCCLS('NOTIFY')
- CBLDCCLS('A-NOTIFY')
- CBLDCCLS('EXNACPT')
- CBLDCCLS('EXNCACPT')

解除するときに、データ名 I に指定したメッセージを一方通知受信待ち状態の CUP に通知できます。

ホスト名長の拡張機能を使用している場合、この関数を使用してください。

(3) UAP で値を設定するデータ領域

- データ名 A
一方通知待ち状態のキャンセルを示す要求コードを「VALUE 'EXCANCEL」と設定します。
- データ名 C
0 を指定します。
- データ名 D
一方通知受信要求時に指定したポート番号を指定します。5001 から 65535 の範囲で指定します。
- データ名 E
一方通知受信待ち状態の CUP が存在するホスト名を指定します。ホスト名として指定できる長さは、63 文字※までです。文字列の最後は空白文字を指定してください。

ホスト名として、10進ドット記法の IP アドレスを指定することもできます。

注※

クライアント環境定義 DCCLTOPTION に 00000008 を指定した場合、ホスト名として指定できる長さは 255 文字までとなります。

• データ名 F

メッセージ長を指定します。0 から DCRPC_MAX_MESSAGE_SIZE* の範囲で指定します。

0 を指定した場合は CUP にメッセージを通知しません。

注※

クライアント環境定義 DCCLTRPCMAXMSGSIZE に 2 以上を指定した場合、DCRPC_MAX_MESSAGE_SIZE の値 (1 メガバイト) ではなく、クライアント環境定義 DCCLTRPCMAXMSGSIZE に指定した値になります。

• データ名 G

CUP に通知するメッセージを格納する領域を指定します。データ名 F で指定する長さ以上の領域を用意してください。

• データ名 H

クライアント環境定義ファイルへのパス名を指定します。パス名には完全パス、またはカレントドライブ・ディレクトリからの相対パスが指定できます。パス名を指定した場合のファイルの読み込み順序を次に示します。

• TP1/Client/P の場合

クライアント環境定義ファイルの読み込み順序は次のとおりです。

- 1.Windows ディレクトリの BETRAN.INI ファイル
- 2.データ名 I に指定したクライアント環境定義ファイル

定義は、クライアント環境定義ファイルおよび BETRAN.INI ファイルのどちらのファイルに指定しても有効です。

両方のファイルに同じ定義を異なる値で指定した場合は、クライアント環境定義ファイルに指定した値が有効となります。

クライアント環境定義ファイルおよび BETRAN.INI ファイルのどちらにも指定がない場合は、デフォルト値で動作します。

• TP1/Client/W の場合

環境変数に指定されている定義は、すべて無効となります。データ名 H に指定したクライアント環境定義ファイルに指定されていない定義はデフォルト値で動作します。

また、データ名 H の先頭に空白を指定することでパス名を省略できます。省略時の動作を次に示します。

• TP1/Client/P の場合

Windows ディレクトリの BETRAN.INI ファイルをクライアント環境定義ファイルとして動作します。BETRAN.INI ファイルがない場合、または定義ファイルの内容が不正な場合はデフォルト値で動作します。

• TP1/Client/W の場合

環境変数の指定で動作します。環境変数が指定されていない場合は、デフォルト値で動作します。
データ名 H に指定したクライアント環境定義ファイルがない場合、または定義ファイルの内容が不正な場合の動作を次に示します。

- TP1/Client/P の場合

Windows ディレクトリの BETRAN.INI ファイルをクライアント環境定義ファイルとして動作します。BETRAN.INI ファイルがない場合、または定義ファイルの内容が不正な場合は、デフォルト値で動作します。

- TP1/Client/W の場合

デフォルト値で動作します。環境変数の指定は無効となります。

(4) 値が返されるデータ領域

- データ名 B

ステータスコードが、5 けたの数字で返されます。

(5) ステータスコード

ステータスコード	意味
00000	正常終了しました。
02501	データ名に設定した値が間違っています。要求コード（データ名 A）が間違っている場合も含まれます。
02503	初期化に失敗しました。または、クライアント環境定義の指定に誤りがあります。
02504	必要なバッファが確保できませんでした。
02506	ネットワーク障害が発生しました。
02514	CUP が一方通知受信待ち状態ではありません。
02518	システムエラーが発生しました。
02539	ホスト名が不正です。
02547	OS が自動的に割り当てるポート番号が不足しています。

6.7.5 CBLDCCLS('O-NOTIFY') – 一方通知受信の開始

(1) 形式

(a) マルチスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCCLS' USING 一意名1
```

■ DATA DIVISION の指定

```
01 一意名1.  
02 データ名A PIC X(8) VALUE 'O-NOTIFY'.  
02 データ名B PIC X(5).  
02 FILLER PIC X(3).  
02 データ名C PIC S9(9) COMP VALUE ZERO.  
02 データ名D PIC 9(4) COMP.  
02 FILLER PIC X(2).  
02 FILLER PIC 9(4) COMP.  
02 FILLER PIC X(2).  
02 データ名F PIC 9(9) COMP-X.  
02 データ名G PIC X(256).
```

(b) シングルスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCCLT' USING 一意名1
```

■ DATA DIVISION の指定

```
01 一意名1.  
02 データ名A PIC X(8) VALUE 'O-NOTIFY'.  
02 データ名B PIC X(5).  
02 FILLER PIC X(3).  
02 データ名C PIC S9(9) COMP VALUE ZERO.  
02 データ名D PIC 9(4) COMP.  
02 FILLER PIC X(2).
```

(2) 機能

サーバからの一方通知受信機能を使用するための環境を作成します。

この関数は、CBLDCCLS('C-NOTIFY')と対で発行します。

(3) UAP で値を設定するデータ領域

- データ名 A

一方通知受信の開始を示す要求コードを「VALUE 'O-NOTIFY」と指定します。

- データ名 C

0 を指定します。

- データ名 D

クライアントのポート番号を指定します。5001 から 65535 の範囲で指定します。なお、同一マシン内で、複数のプロセス、または複数のスレッドを同時に実行する場合は、それぞれ異なるポート番号を指定してください。

- データ名 G

クライアント環境定義ファイルへのパス名を指定します。パス名には完全パス、またはカレントドライブ・ディレクトリからの相対パスが指定できます。パス名を指定した場合のファイルの読み込み順序を次に示します。

- TP1/Client/P の場合

クライアント環境定義ファイルの読み込み順序は次のとおりです。

- 1.Windows ディレクトリの BETRAN.INI ファイル
- 2.データ名 I に指定したクライアント環境定義ファイル

定義は、クライアント環境定義ファイルおよび BETRAN.INI ファイルのどちらのファイルに指定しても有効です。

両方のファイルに同じ定義を異なる値で指定した場合は、クライアント環境定義ファイルに指定した値が有効となります。

クライアント環境定義ファイルおよび BETRAN.INI ファイルのどちらにも指定がない場合は、デフォルト値で動作します。

- TP1/Client/W の場合

環境変数に指定されている定義は、すべて無効となります。データ名 G に指定したクライアント環境定義ファイルに指定されていない定義はデフォルト値で動作します。

また、データ名 G の先頭に空白を指定することでパス名を省略できます。省略時の動作を次に示します。

- TP1/Client/P の場合

Windows ディレクトリの BETRAN.INI ファイルをクライアント環境定義ファイルとして動作します。BETRAN.INI ファイルがない場合、または定義ファイルの内容が不正な場合はデフォルト値で動作します。

- TP1/Client/W の場合

環境変数の指定で動作します。環境変数が指定されていない場合は、デフォルト値で動作します。

データ名 G に指定したクライアント環境定義ファイルがない場合、または定義ファイルの内容が不正な場合の動作を次に示します。

- TP1/Client/P の場合

Windows ディレクトリの BETRAN.INI ファイルをクライアント環境定義ファイルとして動作します。BETRAN.INI ファイルがない場合、または定義ファイルの内容が不正な場合は、デフォルト値で動作します。

- TP1/Client/W の場合
デフォルト値で動作します。環境変数の指定は無効となります。

(4) 値が返されるデータ領域

- データ名 B
ステータスコードが、5 けたの数字で返されます。
- データ名 F
一方通知受信 ID が設定されます。設定された一方通知受信 ID は、CBLDCCLS('C-NOTIFY')を実行するまで破壊してはなりません。

(5) ステータスコード

ステータスコード	意味
00000	正常終了しました。
02501	データ領域に指定した値が誤っています。
02502	すでに CBLDCCLT('O-NOTIFY')が実行されています。CBLDCCLS('O-NOTIFY')実行時、このステータスコードは返りません。
02503	初期化に失敗しました。または、クライアント環境定義の指定に誤りがあります。
02504	必要なバッファが確保できませんでした。
02547	指定したポート番号は使用されています。

(6) 注意事項

- この関数が正常終了した場合は必ず CBLDCCLS('C-NOTIFY')を発行してください。CBLDCCLS('C-NOTIFY')を発行しなかった場合、リソースが残ることがあります。
- 同一マシン内で、複数のプロセス、または複数のスレッドを同時に実行する場合、データ名 D にはそれぞれ異なるポート番号を指定してください。また、データ名 D に指定できるポート番号でも、OS またはほかのプログラムが使用するポート番号は指定しないでください。指定した場合、応答データを正しく受信できないことがあります。なお、OS が使用するポート番号は、OS ごとに異なります。OS が使用するポート番号については、OS のマニュアルなどを参照してください。

6.7.6 CBLDCCLS('C-NOTIFY') – 一方通知受信の終了

(1) 形式

(a) マルチスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCCLS' USING 一意名1
```

■ DATA DIVISION の指定

```
01 一意名1.  
02 データ名A PIC X(8) VALUE 'C-NOTIFY'.  
02 データ名B PIC X(5).  
02 FILLER PIC X(3).  
02 データ名C PIC S9(9) COMP VALUE ZERO.  
02 データ名D PIC 9(9) COMP-X.
```

(b) シングルスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCCLT' USING 一意名1
```

■ DATA DIVISION の指定

```
01 一意名1.  
02 データ名A PIC X(8) VALUE 'C-NOTIFY'.  
02 データ名B PIC X(5).  
02 FILLER PIC X(3).  
02 データ名C PIC S9(9) COMP VALUE ZERO.
```

(2) 機能

サーバからの一方通知受信機能を使用するための環境を削除します。

この関数は、CBLDCCLS('O-NOTIFY')と対で発行します。

(3) UAP で値を設定するデータ領域

- データ名 A
一方通知受信の終了を示す要求コードを「VALUE 'C-NOTIFY」と指定します。
- データ名 C
0 を指定します。
- データ名 D

CBLDCCLT('O-NOTIFY')で受け取った一方通知受信 ID を指定します。

(4) 値が返されるデータ領域

- データ名 B

ステータスコードが、5 けたの数字で返されます。

(5) ステータスコード

ステータスコード	意味
00000	正常終了しました。
02501	データ領域に指定した値が誤っています。
02504	必要なバッファが確保できませんでした。
02544	データ名 D に指定した一方通知受信 ID は、CBLDCCLS('O-NOTIFY')で受け取った一方通知受信 ID と異なっています。

6.7.7 CBLDCCLS('A-NOTIFY') – 一方通知受信

(1) 形式

(a) マルチスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCCLS' USING 一意名1 一意名2
```

■ DATA DIVISION の指定

```
01 一意名1.  
02 データ名A PIC X(8) VALUE 'A-NOTIFY'.  
02 データ名B PIC X(5).  
02 FILLER PIC X(3).  
02 データ名C PIC S9(9) COMP VALUE ZERO.  
02 データ名D PIC S9(9) COMP.  
02 データ名E PIC X(64).  
02 データ名F PIC X(8).  
02 データ名G PIC 9(9) COMP-X.  
01 一意名2.  
02 データ名H PIC S9(9) COMP.  
02 データ名I PIC X(n).
```

(b) シングルスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCCLT' USING 一意名1 一意名2
```

■ DATA DIVISION の指定

```
01 一意名1.  
02 データ名A PIC X(8) VALUE 'A-NOTIFY'.  
02 データ名B PIC X(5).  
02 FILLER PIC X(3).  
02 データ名C PIC S9(9) COMP VALUE ZERO.  
02 データ名D PIC S9(9) COMP.  
02 データ名E PIC X(64).  
02 データ名F PIC X(8).  
  
01 一意名2.  
02 データ名H PIC S9(9) COMP.  
02 データ名I PIC X(n).
```

(2) 機能

サーバ側の要求コード (CBLDCRPC('CLTSEND')) によって通知されるメッセージを、データ名 D で指定した値まで待ち続けます。受信した時点で CUP に制御を戻し、ステータスコード、通知メッセージ、通知メッセージ長、通知元サーバのホスト名、通知元サーバのノード識別子を返します。

この関数を発行する前に、CBLDCCLS('O-NOTIFY') を発行しておく必要があります。

(3) UAP で値を設定するデータ領域

- データ名 A
一方通知受信を示す要求コードを「VALUE 'A-NOTIFY」と指定します。
- データ名 C
0 を指定します。
- データ名 D
タイムアウト値 (秒) を指定します。0 から 65535 の範囲で指定します。無限に待ち続ける場合は、0 を指定します。
- データ名 G
CBLDCCLS('O-NOTIFY') で受け取った一方通知受信 ID を指定します。
- データ名 H
サーバからの通知メッセージを格納する領域長 (データ名 I の長さ) を指定します。0 から DCRPC_MAX_MESSAGE_SIZE* の範囲で指定します。

注※

クライアント環境定義 DCCLTRPCMAXMSGSIZE に 2 以上を指定した場合、DCRPC_MAX_MESSAGE_SIZE の値 (1 メガバイト) ではなく、クライアント環境定義 DCCLTRPCMAXMSGSIZE に指定した値になります。

- データ名 I

サーバからの通知メッセージを格納する領域です。データ名 H で設定する長さ以上の領域を用意してください。

(4) 値が返されるデータ領域

- データ名 B

ステータスコードが、5 けたの数字で返されます。

- データ名 E

通知したサーバのホスト名が返されます。

ホスト名への変換に失敗した場合、10 進ドット記法の IP アドレスが返されます。

- データ名 F

通知したサーバのノード識別子が返されます。ノード識別子のフォーマットは次のとおりです。

ノード識別子 (4バイト)	空白 (4バイト)
---------------	-----------

- データ名 H

サーバからの通知メッセージ長が返されます。

- データ名 I

サーバからの通知メッセージが返されます。

(5) ステータスコード

ステータスコード	意味
00000	正常終了しました。
02501	データ領域に指定した値が誤っています。
02502	CBLDCCLS('O-NOTIFY')が実行されていません。
02504	必要なバッファが確保できませんでした。
02506	ネットワーク障害が発生しました。
02507	メッセージの受信時にタイムアウトになりました。
02518	システムエラーが発生しました。
02535	バージョン不一致が発生しました。
02544	データ名 G に指定した一方通知受信 ID は、CBLDCCLS('O-NOTIFY')で受け取った一方通知受信 ID と異なっています。

ステータスコード	意味
02546	受信したメッセージが、CUP で用意した領域に収まりません。収まらないメッセージは切り捨てました。データ名 E, およびデータ名 F には、値を設定済みです。
02548	不正なメッセージを受信しました。
02549	一方通知受信待ち状態が CBLDCCLS('CANCEL ')によって解除されました。データ名 E, データ名 H, およびデータ名 I には、値を設定済みです。

6.7.8 CBLDCCLS('EXNCACPT') – 一方通知受信 (ホスト名長の拡張時)

(1) 形式

(a) マルチスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCCLS' USING 一意名1 一意名2 一意名3
```

■ DATA DIVISION の指定

```
01 一意名1.
  02 データ名A PIC X(8) VALUE 'EXNCACPT'.
  02 データ名B PIC X(5).
  02 FILLER PIC X(3).
  02 データ名C PIC S9(9) COMP VALUE ZERO.
  02 データ名D PIC S9(9) COMP.
  02 データ名E PIC X(8).
  02 データ名F PIC X(n).
01 一意名2.
  02 データ名G PIC S9(9) COMP.
  02 データ名H PIC X(n).
01 一意名3.
  02 データ名I PIC 9(9) COMP-X.
```

(b) シングルスレッド環境の場合

■ PROCEDURE DIVISION の指定

```
CALL 'CBLDCCLT' USING 一意名1 一意名2
```

■ DATA DIVISION の指定

```
01 一意名1.
  02 データ名A PIC X(8) VALUE 'EXNCACPT'.
  02 データ名B PIC X(5).
  02 FILLER PIC X(3).
  02 データ名C PIC S9(9) COMP VALUE ZERO.
```

```
02 データ名D PIC S9(9) COMP.
02 データ名E PIC X(8).
02 データ名F PIC X(n).
01 一意名2.
02 データ名G PIC S9(9) COMP.
02 データ名H PIC X(n).
```

(2) 機能

サーバ側の要求コード (CBLDCRPC('CLTSEND ')) によって通知されるメッセージを、データ名 D で指定した値まで待ち続けます。受信した時点で CUP に制御を戻し、ステータスコード、通知メッセージ、通知メッセージ長、通知元サーバのホスト名、通知元サーバのノード識別子を返します。

この関数を発行する前に、CBLDCCLS('O-NOTIFY')を発行しておく必要があります。

ホスト名長の拡張機能を使用している場合、この関数を使用してください。

(3) UAP で値を設定するデータ領域

• データ名 A

一方通知受信を示す要求コードを「VALUE 'EXNCACPT」と指定します。

• データ名 C

0 を指定します。

• データ名 D

タイムアウト値 (秒) を指定します。0 から 65535 の範囲で指定します。無限に待ち続ける場合は、0 を指定します。

• データ名 F

通知したサーバのホスト名を格納する 64 バイト*以上の領域を用意してください。

注※

クライアント環境定義 DCCLTOPTION に 00000008 を指定した場合、256 バイト以上の領域を用意してください。

• データ名 G

サーバからの通知メッセージを格納する領域長を指定します。0 から DCRPC_MAX_MESSAGE_SIZE * の範囲で指定します。

注※

クライアント環境定義 DCCLTRPCMAXMSGSIZE に 2 以上を指定した場合、DCRPC_MAX_MESSAGE_SIZE の値 (1 メガバイト) ではなく、クライアント環境定義 DCCLTRPCMAXMSGSIZE に指定した値になります。

• データ名 H

サーバからの通知メッセージを格納する領域を指定します。データ名 G で設定する長さ以上の領域を用意してください。

- データ名 I

CBLDCCLS('O-NOTIFY')で受け取った一方通知受信 ID を指定します。

(4) 値が返されるデータ領域

- データ名 B

ステータスコードが、5 けたの数字で返されます。

- データ名 E

通知したサーバのノード識別子が返されます。ノード識別子のフォーマットは次のとおりです。

ノード識別子 (4バイト)	空白 (4バイト)
---------------	-----------

- データ名 F

通知したサーバのホスト名が返されます。

ホスト名への変換に失敗した場合、10 進ドット記法の IP アドレスが返されます。

- データ名 G

サーバからの通知メッセージ長が返されます。

- データ名 H

サーバからの通知メッセージが返されます。

(5) ステータスコード

ステータスコード	意味
00000	正常終了しました。
02501	データ領域に指定した値が誤っています。
02502	CBLDCCLS('O-NOTIFY')が実行されていません。
02504	必要なバッファが確保できませんでした。
02506	ネットワーク障害が発生しました。
02507	メッセージの受信時にタイムアウトになりました。
02518	システムエラーが発生しました。
02535	バージョン不一致が発生しました。
02544	データ名 I に指定した一方通知受信 ID は、CBLDCCLS('O-NOTIFY')で受け取った一方通知受信 ID と異なっています。
02546	受信したメッセージが、CUP で用意した領域に収まりません。収まらないメッセージは切り捨てました。データ名 E、およびデータ名 F には、値を設定済みです。
02548	不正なメッセージを受信しました。
02549	一方通知受信待ち状態が CBLDCCLS('CANCEL ')または CBLDCCLS('EXNCANCL')によって解除されました。データ名 F、データ名 G、およびデータ名 H には、値を設定済みです。

6.8 文字コード変換機能（コードマッピングテーブルを使用しない場合）

文字コード変換機能は、TP1/Client/P でだけ使用できる機能です。

文字コード変換機能で提供する要求文は、マルチスレッド環境でも正しく動作します。

6.8.1 CBLDCUTL('CODECNV') - 文字コード変換

(1) 形式

- PROCEDURE DIVISION の指定

```
CALL 'CBLDCUTL' USING 一意名1 一意名2 一意名3
```

- DATA DIVISION の指定

```
01 一意名1.  
  02 データ名A PIC X(8) VALUE 'CODECNV'.  
  02 データ名B PIC X(5).  
  02 FILLER PIC X(3).  
  02 データ名C PIC S9(9) COMP VALUE ZERO.  
  02 データ名D PIC S9(9) COMP.  
  
01 一意名2.  
  02 データ名E PIC S9(9) COMP.  
  02 データ名F PIC X(n).  
  
01 一意名3.  
  02 データ名G PIC S9(9) COMP.  
  02 データ名H PIC X(n).
```

(2) 機能

- JIS コード、またはシフト JIS コードで構成される文字列を、EBCDIC コード、EBCDIK コード、または KEIS コードで構成される文字列に変換します。
- EBCDIC コード、EBCDIK コードまたは KEIS コードで構成される文字列を、JIS コード、またはシフト JIS コードで構成される文字列に変換します。

(3) UAP で値を設定するデータ領域

- データ名 A

文字コード変換を示す要求コードを「VALUE 'CODECNV△」と設定します。

- データ名 C

変換時の条件（変換オプション）を、使用するオプションの数値の和で指定します。

0：デフォルト（各オプションの指定がない場合、この条件になる）

- EBCDIK コードを使用します。
- 全角スペースを全角スペースのままにします。
- '83 版 KEIS コードを使用します。
- 無効コードがあった場合、エラーにします。
- タブコード、または制御コードを半角コードとして認識しません。直前、または直後のデータが全角コードの場合でもシフトコードは付けられません。

1：EBCDIC コードを使用します。

2：全角スペースを半角スペース 2 個に変換します。この指定は、データ名 D の値が 1 のときだけ有効です。

4：'78 版 KEIS コードを使用します。

8：無効コードがあった場合、スペースに変換します。

16：タブコードを半角コードとして認識します。

直前、または直後のデータが全角コードの場合はシフトコードが付けられます。

32：制御コードを半角コードとして認識します。

直前、または直後のデータが全角コードの場合はシフトコードが付けられます。

• データ名 D

変換の方法を指定します。

1

JIS コード、またはシフト JIS コードで構成される文字列を、EBCDIC コード、EBCDIK コード、または KEIS コードで構成される文字列に変換します。

2

EBCDIC コード、EBCDIK コード、または KEIS コードで構成される文字列を、JIS コード、またはシフト JIS コードで構成される文字列に変換します。

• データ名 E

変換する文字列長を指定します。1 から DCRPC_MAX_MESSAGE_SIZE までの範囲の長さが指定できます。

• データ名 F

変換する文字列を指定します。

• データ名 G

変換後の文字列を受け取る領域の長さを指定します。

• データ名 H

変換後の文字列を格納する領域です。データ名 G で設定する長さ以上の領域を用意してください。

(4) 値が返されるデータ領域

• データ名 B

ステータスコードが、5 けたの数字で返されます。

- データ名 G

変換後の文字列長が返されます。

- データ名 H

変換後の文字列が返されます。

(5) ステータスコード

ステータスコード	意味
00000	正常に終了しました。
02501	データ名に指定した値が誤っています。要求コード（データ名 A）が間違っている場合も含まれます。
02504	メモリ不足が発生しました。 このステータスコードは、変換する文字列に全角文字（2 バイト）が含まれているのに、全角文字の 1 バイト目までの文字列長が指定されている場合にも返されます。
02550	文字列中に無効コードがあります。
02551	変換後の文字列の長さが CUP で用意した領域を超えています。

(6) 注意事項

- データ名 D に 2 を指定し、データ名 C に 16 を指定、またはデータ名 D に 2 を指定し、データ名 C に 32 を指定した場合、タブコードまたは制御コードを半角コードとして意識させたデータを、あらかじめ用意しておく必要があります。
- コード変換の仕様の詳細については、「付録 A コード変換の仕様」を参照してください。

6.9 文字コード変換機能（コードマッピングテーブルを使用する場合）

文字コード変換機能は、TP1/Client/P でだけ使用できる機能です。

文字コード変換機能で提供する要求文は、マルチスレッド環境でも正しく動作します。

6.9.1 CBLDCUTL('CNVOPN') - 文字コード変換の開始

(1) 形式

(a) PROCEDURE DIVISION の指定

```
CALL 'CBLDCUTL' USING 一意名1 一意名1 一意名1
```

(b) DATA DIVISION の指定

TP1/Client/P の場合

```
01 一意名1.  
02 データ名A PIC X(8) VALUE 'CNVOPN'.  
02 データ名B PIC X(5).  
02 FILLER PIC X(3).  
02 データ名C PIC X(256).  
02 データ名D PIC S9(9) COMP.  
02 データ名E PIC 9(9) COMP.
```

TP1/Client/P(64)の場合

```
01 一意名1.  
02 データ名A PIC X(8) VALUE 'CNVOPN'.  
02 データ名B PIC X(5).  
02 FILLER PIC X(3).  
02 データ名C PIC X(256).  
02 データ名D PIC S9(9) COMP.  
02 FILLER PIC X(4).  
02 データ名E PIC 9(18) COMP.
```

(2) 機能

文字コード変換を開始し、使用するコードマッピングテーブルをメモリに確保します。

(3) UAP で値を設定するデータ領域

- データ名 A

文字コード変換の開始を示す要求コードを「VALUE 'CNVOPN△△」と設定します。

- データ名 C
空白を設定します。
- データ名 D
変換方法を指定します。
1：CommuniNet との連携による変換を行います。
0：コードマッピングテーブルを使用しないで、換算による変換を行います。

(4) 値が返されるデータ領域

- データ名 B
ステータスコードが、5 けたの数字で返されます。
- データ名 E
メモリ上に確保された文字コード変換制御テーブルのハンドルが返されます。

(5) ステータスコード

ステータスコード	意味
00000	正常終了しました。
02501	データ名に指定した値が誤っています。要求コード（データ名 A）が間違っている場合も含まれます。
02504	メモリ不足が発生しました。
02557	コードマッピングテーブルが存在しません。
02558	現在使用中のコードマッピングテーブルは未サポートです。このステータスコードは、CommuniNet インストール後、CommuniNet コードマッピングユーティリティでコードマッピングテーブルの保存を一度も行っていない場合にも返されます。
02559	コードマッピングテーブルの I/O エラーが起きました。

(6) 注意事項

- CALL 文の USING 句に一意名 1 を三つ指定してください。
- この機能を使用する場合は、CommuniNet のコードマッピングテーブルが必要です。CommuniNet のコードマッピングユーティリティで、コードマッピングテーブルを作成してから、この機能を使用してください。
- CommuniNet インストール後、一度も CommuniNet コードマッピングユーティリティで保存されていないコードマッピングテーブルは、使用できません。この機能を使用する前に、CommuniNet コードマッピングユーティリティでコードマッピングテーブルを保存してください。
- CommuniNet のコードマッピングテーブルのファイル名は、必ず CMAPEX.TBL とし、この機能を使用する前に Windows のディレクトリ下に格納してください。

- この機能を使用している途中で、CommuniNet のコードマッピングユティリティによって、コードマッピングテーブルの内容を変更しても、文字コード変換機能の処理には反映されません。
- この機能では、エラーログ、および UAP トレースの情報は取得しません。
- 文字コード変換の開始 (CBLDCUTL('CNVOPN ')) は、1 回だけ発行して、文字コード変換の実行 (CBLDCUTL('CNVEXEC ')) をしてください。メモリ不足となるおそれがあるため、文字コード変換の開始は複数回発行しないでください。複数回発行した場合は、発行した回数分、文字コード変換の終了 (CBLDCUTL('CNVCLS ')) を発行してください。

6.9.2 CBLDCUTL('CNVCLS ') – 文字コード変換の終了

(1) 形式

(a) PROCEDURE DIVISION の指定

```
CALL 'CBLDCUTL' USING 一意名1 一意名1 一意名1
```

(b) DATA DIVISION の指定

TP1/Client/P の場合

```
01 一意名1.
02 データ名A PIC X(8) VALUE 'CNVCLS '.
02 データ名B PIC X(5).
02 FILLER PIC X(3).
02 データ名C PIC S9(9) COMP VALUE ZERO.
02 データ名D PIC S9(9) COMP.
```

TP1/Client/P(64)の場合

```
01 一意名1.
02 データ名A PIC X(8) VALUE 'CNVCLS '.
02 データ名B PIC X(5).
02 FILLER PIC X(3).
02 データ名C PIC S9(9) COMP VALUE ZERO.
02 FILLER PIC X(4).
02 データ名D PIC 9(18) COMP.
```

(2) 機能

文字コード変換を終了し、コードマッピングテーブルが確保されたメモリ上の領域を解放します。

(3) UAP で値を設定するデータ領域

- データ名 A

文字コード変換の終了を示す要求コードを「VALUE 'CNVCLS△△」と設定します。

- データ名 C

0 を設定します。

- データ名 D

CBLDCUTL('CNVOPN ')で取得した文字コード変換で使用する制御テーブルのハンドルを指定します。

(4) 値が返されるデータ領域

- データ名 B

ステータスコードが、5 けたの数字で返されます。

(5) ステータスコード

ステータスコード	意味
00000	正常終了しました。
02501	データ名に指定した値が誤っています。要求コード（データ名 A）が間違っている場合も含まれます。
02504	メモリ不足が発生しました。

(6) 注意事項

- CALL 文の USING 句に一意名 1 を三つ指定してください。
- この機能を使用する場合は、CommuniNet のコードマッピングテーブルが必要です。CommuniNet のコードマッピングユティリティで、コードマッピングテーブルを作成してから、この機能を使用してください。
- CommuniNet インストール後、一度も CommuniNet コードマッピングユティリティで保存されていないコードマッピングテーブルは、使用できません。この機能を使用する前に、CommuniNet コードマッピングユティリティでコードマッピングテーブルを保存してください。
- CommuniNet のコードマッピングテーブルのファイル名は、必ず CMAPEX.TBL とし、この機能を使用する前に Windows のディレクトリ下に格納してください。
- この機能を使用している途中で、CommuniNet のコードマッピングユティリティによって、コードマッピングテーブルの内容を変更しても、文字コード変換機能の処理には反映されません。
- この機能では、エラーログ、および UAP トレースの情報は取得しません。
- 文字コード変換の開始 (CBLDCUTL('CNVOPN ')) は、1 回だけ発行して、文字コード変換の実行 (CBLDCUTL('CNVEXEC ')) をしてください。メモリ不足となるおそれがあるため、文字コード変換の開始は複数回発行しないでください。複数回発行した場合は、発行した回数分、文字コード変換の終了 (CBLDCUTL('CNVCLS ')) を発行してください。

6.9.3 CBLDCUTL('CNVEXEC ') – 文字コード変換の実行

(1) 形式

(a) PROCEDURE DIVISION の指定

```
CALL 'CBLDCUTL' USING 一意名1 一意名2 一意名3
```

(b) DATA DIVISION の指定

TP1/Client/P の場合

```
01 一意名1.  
02 データ名A PIC X(8) VALUE 'CNVEXEC'.  
02 データ名B PIC X(5).  
02 FILLER PIC X(3).  
02 データ名C PIC S9(9) COMP VALUE ZERO.  
02 データ名D PIC S9(9) COMP.  
02 データ名E PIC 9(9) COMP.  
  
01 一意名2.  
02 データ名F PIC S9(9) COMP.  
02 データ名G PIC X(n).  
  
01 一意名3.  
02 データ名H PIC S9(9) COMP.  
02 データ名I PIC X(n).
```

TP1/Client/P(64)の場合

```
01 一意名1.  
02 データ名A PIC X(8) VALUE 'CNVEXEC'.  
02 データ名B PIC X(5).  
02 FILLER PIC X(3).  
02 データ名C PIC S9(9) COMP VALUE ZERO.  
02 データ名D PIC S9(9) COMP.  
02 データ名E PIC 9(18) COMP.  
  
01 一意名2.  
02 データ名F PIC S9(9) COMP.  
02 データ名G PIC X(n).  
  
01 一意名3.  
02 データ名H PIC S9(9) COMP.  
02 データ名I PIC X(n).
```

(2) 機能

次に示す文字コード変換を実行します。

JIS コード、またはシフト JIS コードで構成される文字列を、EBCDIC コード、EBCDIK コード、または KEIS コードで構成される文字列に変換します。

または、EBCDIC コード、EBCDIK コード、または KEIS コードで構成される文字列を、JIS コード、またはシフト JIS コードで構成される文字列に変換します。

(3) UAP で値を設定するデータ領域

- データ名 A

文字コード変換の実行を示す要求コードを「VALUE 'CNVEXEC△」と設定します。

- データ名 C

変換時の条件（変換オプション）を、使用するオプションの数値の和で指定します。

0：デフォルトです（各オプションの指定がない場合、この条件になります）。

- EBCDIK コードを使用します。
- 全角スペースを全角スペースのままにします。
- '83 版 KEIS コードを使用します。
- 無効コードがあった場合、エラーにします。
- タブコード、または制御コードを半角コードとして認識しません。直前、または直後のデータが全角コードの場合でもシフトコードは付けられません。

1：EBCDIC コードを使用します。

2：全角スペースを半角スペース 2 個に変換します。

この指定は、データ名 D の値が 1 のときだけ有効です。

4：'78 版 KEIS コードを使用します。

8：無効コードがあった場合、スペースに変換します。

16：タブコードを半角コードとして認識します。

直前、または直後のデータが全角コードの場合はシフトコードが付けられます。

32：制御コードを半角コードとして認識します。

直前、または直後のデータが全角コードの場合はシフトコードが付けられます。

- データ名 D

変換の方法を指定します。

1

JIS コード、またはシフト JIS コードで構成される文字列を、EBCDIC コード、EBCDIK コード、または KEIS コードに変換します。

2

EBCDIC コード、EBCDIK コード、または KEIS コードで構成される文字列を、JIS コード、またはシフト JIS コードに変換します。

- データ名 E
CBLDCUTL('CNVOPN ')で取得したコード変換で使用する制御テーブルのハンドルを指定します。
- データ名 F
変換する文字列の長さを指定します。
1 から DCRPC_MAX_MESSAGE_SIZE までの範囲の長さが指定できます。
- データ名 G
変換する文字列を指定します。
- データ名 H
変換後の文字列を受け取る領域の長さを指定します。
- データ名 I
変換後の文字列を格納する領域です。データ名 H で設定する長さ以上の領域を用意してください。

(4) 値が返されるデータ領域

- データ名 B
ステータスコードが 5 けたの数字で返されます。
- データ名 H
変換後の文字列の長さが返されます。
- データ名 I
変換後の文字列が返されます。

(5) ステータスコード

ステータスコード	意味
00000	正常に終了しました。
02501	データ名に指定した値が誤っています。要求コード（データ名 A）が間違っている場合も含まれます。
02504	メモリ不足が発生しました。 このコードは、制御テーブルのハンドルの値が不正である場合、および変換する文字列に全角文字（2 バイト）が含まれているのに、全角文字の 1 バイト目までの文字列長が指定されている場合にも返されます。
02550	文字列中に無効コードがあります。
02551	変換後の文字列の長さが CUP で用意した領域を超えています。

(6) 注意事項

- この機能を使用する場合は、CommuniNet のコードマッピングテーブルが必要です。CommuniNet のコードマッピングユーティリティで、コードマッピングテーブルを作成してから、この機能を使用してください。

- CommuniNet インストール後、一度も CommuniNet コードマッピングユーティリティで保存されていないコードマッピングテーブルは、使用できません。この機能を使用する前に、CommuniNet コードマッピングユーティリティでコードマッピングテーブルを保存してください。
- CommuniNet のコードマッピングテーブルのファイル名は、必ず CMAPEX.TBL とし、この機能を使用する前に Windows のディレクトリ下に格納してください。
- この機能を使用している途中で、CommuniNet のコードマッピングユーティリティによって、コードマッピングテーブルの内容を変更しても、文字コード変換機能の処理には反映されません。
- この機能では、エラーログ、および UAP トレースの情報は取得しません。
- 文字コード変換の開始(CBLDCUTL('CNVOPN '))は、1 回だけ発行して、文字コード変換の実行(CBLDCUTL('CNVEXEC '))をしてください。メモリ不足となるおそれがあるため、文字コード変換の開始は複数回発行しないでください。複数回発行した場合は、発行した回数分、文字コード変換の終了(CBLDCUTL('CNVCLS '))を発行してください。
- データ名 D に 2 を指定し、データ名 C に 16 を指定、またはデータ名 D に 2 を指定し、データ名 C に 32 を指定した場合、タブコードまたは制御コードを半角コードとして意識させたデータを、あらかじめ用意しておく必要があります。
- コード変換の仕様の詳細については、「付録 A コード変換の仕様」を参照してください。

7

環境設定

TP1/Client/P の環境設定とその手順を説明します。

7.1 Windows の C ランタイムライブラリのインストール

TP1/Client/P は Visual Studio で作成されています。このため、TP1/Client/P の CUP を実行する場合、Windows の C ランタイムライブラリをインストールする必要があります。

インストールが必要な Windows の C ランタイムライブラリについては、「リリースノート」を参照してください。

8

定義

クライアント環境定義について説明します。

この章では、各関数の DLL を呼び出すときの C 言語の関数名 (dc_xxx_xxx_s) で説明します。通常オブジェクトライブラリの関数および COBOL 言語を使う場合は、各関数に対応する通常オブジェクトライブラリの関数名 (dc_xxx_xxx) および COBOL 言語の要求文に置き換えて読んでください。

8.1 定義の概要

ここでは、クライアント環境定義の一覧、および定義の規則について説明します。

8.1.1 クライアント環境定義の一覧

クライアント環境定義の一覧を、次の表に示します。

表 8-1 クライアント環境定義の一覧

項番	オペランド	定義内容	指定値
1	DCNAMPORT	ネームサービスのポート番号	〈符号なし整数〉 ((5001～65535)) 《10000》
2	DCHOST	窓口となる TP1/Server	〈文字列〉
3	DCWATCHTIM	最大応答待ち時間	〈符号なし整数〉 ((0～65535)) 《180》 (単位：秒)
4	DCCLTCONNECTTIMEOUT	コネクション確立最大監視時間	〈符号なし整数〉 ((0～65535)) 《0》 (単位：秒)
5	DCCLTTREXPTM	トランザクションブランチ限界経過時間	〈符号なし整数〉 ((0～65535)) (単位：秒)
6	DCCLTTREXPSP	トランザクションブランチの処理を監視するとき、「トランザクショナル RPC 実行プロセスのトランザクションブランチが、RPC 機能を使ってほかのトランザクションブランチを呼び出し、その処理が終わるのを待つ時間」を監視時間に含むかどうかを指定	Y N F
7	DCCLTTRWATTM	トランザクション問い合わせ間隔最大時間	〈符号なし整数〉 ((1～65535)) 《180》 (単位：秒)
8	DCCLTTRCPUTM	トランザクションブランチ CPU 監視時間	〈符号なし整数〉 ((0～65535)) (単位：秒)
9	DCCLTUTTRCMT	オンラインテストの機能を使って CUP からトランザクションを開始させた場合、トランザクションを同期点でコミットするかロールバックするかを指定	Y 《N》
10	DCRCVPORT	受信で使用する CUP のポート番号	〈符号なし整数〉 ((1～65535)) 《11000》
11	DCSNDHOST	接続先のノード名	〈文字列〉
12	DCSNDPORT	接続先のポート番号	〈符号なし整数〉 ((1～65535)) 《12000》

項番	オペランド	定義内容	指定値
13	DCSOCKOPENATRCV	TCP/IP 通信機能の使用時に、1 コネクションで送受信する場合の、受信用ソケットを開設する契機（送信相手からの接続を待ち受け始める契機）を指定	Y 《N》
14	DCCLTDELIVERYCHECK	メッセージの送達確認機能を使用するかどうかを指定	Y 《N》
15	DCUTOKEY	テストユーザ ID	〈1~4 文字の英数字〉
16	DCCACHE	サービス情報を一時的に格納する領域数	〈符号なし整数〉 ((2~10240)) 《8》
17	DCCLTCACHETIM	一時的に格納したサービス情報の有効時間	〈符号なし整数〉 ((0~65535)) 《30》 (単位：秒)
18	DCCLTLOADBALANCE	マルチノードサーバでの運用時、RPC 実行時に各ノードの負荷状態を TP1/Client の内部で評価して、負荷の軽いサーバに分散させる機能（ノード間負荷バランス機能）を使用するかどうかを指定	Y 《N》
19	DCCLTSERVICEGROUPLIST	サービスグループと RPC 受け付け窓口の対応を定義したファイル名	〈文字列〉
20	DCCLTCONNECTRETRY	コネクション確立の再試行回数	〈符号なし整数〉 ((0~255)) 《0》
21	DCSCDDIRECT	TP1/Server のネームサービスにサービス情報を問い合わせないで、直接スケジュールサービスに問い合わせる機能（ネームサービスを使用しない RPC）を使用するかどうかを指定	Y 《N》
22	DCSCDPORT	スケジュールサービスのポート番号	〈符号なし整数〉 ((5001~65535))
23	DCCLTDATACOMP	データ圧縮機能を使用するかどうかを指定	Y 《N》
24	DCEXTENDFUNCTION	RPC サービスの機能拡張レベル	〈符号なし 16 進整数〉 ((00000000~00000001)) 《00000000》
25	DCCLTINQUIRETIME	常設コネクション問い合わせ間隔最大時間	〈符号なし整数〉 ((0~1048575)) (単位：秒)
26	DCCLTPORT	クライアント拡張サービスのポート番号	〈符号なし整数〉 ((5001~65535))
27	DCCLTDCCMHOST	DCCM3 の論理端末に対して常設コネクションの確立要求をする場合に、確立要求先の論理端末のホスト名を指定	DCCM3 論理端末のホスト名
28	DCCLTDCCMPORT	DCCM3 論理端末のポート番号	〈符号なし整数〉 ((1~65535)) 《30000》

項番	オペランド	定義内容	指定値
29	DCCLTXATMI	XATMI インタフェースを使用して通信をするかどうかを指定	Y 《N》
30	DCWATCHTIMINHERIT	トランザクション制御, およびコネクション制御を行う場合, クライアント拡張サービスに CUP の最大応答待ち時間を引き継ぐかどうかを指定	Y 《N》
31	DCCLTDELAY	最大通信遅延時間	〈符号なし整数〉 ((0~65535)) 《0》 (単位: 秒)
32	DCCLTCUPSNDHOST	CUP の送信元ホスト	〈文字列〉
33	DCCLTCUPRCVPORT	CUP の受信で使用するポート番号	〈符号なし整数〉 ((5001~65535))
34	DCCLTRAPHOST	TP1/Server の rap リスナーのホスト名およびポート番号, または DCCM3 の論理端末のホスト名およびポート番号を指定	rap リスナー, または DCCM3 の論理端末
35	DCCLTRAPAUTOCONNECT	CUP と rap サーバまたは DCCM3 論理端末との間の常設コネクションを自動的に確立させるかどうかを指定	Y 《N》
36	DCCLTTRSTATISITEM	トランザクションブランチの統計情報を取得する項目を指定	統計情報項目 [,統計情報項目] ...
37	DCCLTTROPTIITEM	複数のユーザサーバで構成されるグローバルトランザクションの性能を向上させるための最適化項目を指定	トランザクション最適化項目 [,トランザクション最適化項目] ...
38	DCCLTTRWATCHTIME	トランザクション同期点処理時の最大通信待ち時間	〈符号なし整数〉 ((1~65535)) (単位: 秒)
39	DCCLTTRRINFO	トランザクションブランチがロールバックした場合に, ロールバック要因に関する情報をログに取得するかどうかを指定	no self remote all
40	DCCLTTRLIMITTIME	トランザクションブランチ最大実行可能時間	〈符号なし整数〉 ((0~65535)) (単位: 秒)
41	DCCLTTRRBRCV	RPC 先トランザクションブランチにロールバック指示を送信したあと, ロールバック完了通知を受信するかどうかを指定	Y N
42	DCCLTTRRECOVERYTYPE	UAP 障害時のトランザクション同期点処理方式を指定	type1 type2 type3
43	DCWATCHTIMRPCINHERIT	CUP の最大応答待ち時間を, サーバ側に引き継ぐかどうかを指定	Y 《N》
44	DCSYSWATCHTIM	OpenTP1 制御の最大応答待ち時間	〈符号なし整数〉 ((0~65535)) 《最大応答待ち時間》 (単位: 秒)
45	DCCLTAUTHENT	dc_clt_cltin_s 関数実行時のユーザ認証を行うかどうかを指定	《Y》 N

項番	オペランド	定義内容	指定値
46	DCCLTCONNECTINF	端末識別情報を指定	端末識別情報
47	DCSCDMULTI	マルチスケジューラ機能を使用するかどうかを指定	Y 《N》
48	DCSCDMULTICOUNT	マルチスケジューラデーモンのプロセス数	〈符号なし整数〉 ((1~4096)) 《1》
49	DCHOSTSELECT	窓口となる TP1/Server をランダムに選択するかどうかを指定	Y 《N》
50	DCSCDLOADPRIORITY	サービス要求を受け付けた窓口となる TP1/Server を優先して負荷分散するかどうかを指定	Y 《N》
51	DCCLTONLYTHISNODE	dc_rpc_call_to_s 関数発行時、指定したノードでサービスを実行するかどうかを指定	Y 《N》
52	DCCLTNOSERVER	TP1/Server とは通信することがない環境かどうかを指定	Y 《N》
53	DCHOSTCHANGE	サービス要求 (dc_rpc_call_s 関数実行) 時、スケジュールサービス開始処理中、およびスケジュールサービス終了処理中に、窓口となる TP1/Server からエラー応答を受信した場合、窓口となる TP1/Server をほかの TP1/Server に切り替えるかどうかを指定	《Y》 N
54	DCCLTOPTION	クライアントの機能拡張オプション	〈符号なし 16 進整数〉 ((00000000~00000008)) 《00000000》
55	DCCLTNAMEXTEND	マルチノードサーバで運用している場合に、クライアントで管理するサービス情報の取得数を拡張	《0》 1
56	DCTRCPATH	トレースファイル作成ディレクトリ	〈文字列〉
57	DCTRCERR	エラーログのサイズ	〈符号なし整数〉 ((0~1073741824)) 《4096》 (単位: バイト)
58	DCTRCUAP	UAP トレースのファイルサイズ	〈符号なし整数〉 ((4096~1073741824)) 《出力しない》 (単位: バイト)
59	DCTRCSOC	ソケットトレースのファイルサイズ	〈符号なし整数〉 ((4096~1073741824)) 《出力しない》 (単位: バイト)
60	DCTRCSOC SIZE	ソケットトレースのデータサイズ	〈符号なし整数〉 ((64~4096)) 《256》 (単位: バイト)

項番	オペランド	定義内容	指定値
61	DCTRCMDL	モジュールトレースのファイルサイズ	〈符号なし整数〉 ((4096～1073741824)) 《出力しない》 (単位：バイト)
62	DCCLTPRFINFOSEND	性能検証用トレースの識別情報を、TP1/Server に伝播するかどうかを指定	Y 《N》
63	DCCLTRPCMAXMSGSIZE	RPC 送受信電文の最大長	〈符号なし整数〉 ((1～8)) 《1》 (単位：メガバイト)
64	DCCLTRECVBUFFSIZE	TCP/IP の受信バッファサイズ	〈符号なし整数〉 ((8192～1048576)) (単位：バイト)
65	DCCLTSENDBUFFSIZE	TCP/IP の送信バッファサイズ	〈符号なし整数〉 ((8192～1048576)) (単位：バイト)
66	DCCLTTCPNODELAY	Nagle アルゴリズムを無効にするかどうかを指定	Y 《N》
67	DCCLTBACKLOGCOUNT ^{※1}	コネクション確立要求を格納するキューの数	〈符号なし整数〉 ((0～4096)) 《0》
68	dcselint ^{※2}	受信チェック間隔時間	〈符号なし整数〉 ((0～65535)) 《100》 (単位：ミリ秒)

注※1

TP1/Client/W 固有のオペランドです。

注※2

TP1/Client/P 固有のオペランドです。

8.1.2 定義の規則

定義の説明に使用する各種の記号を説明します。

ここで述べる文法記述記号、属性表示記号、および構文要素記号は実際の定義には記述しません。

(1) 文法記述記号

文法の記述について説明する記号です。

文法記述記号	意味
[]	この記号で囲まれている項目は省略できることを示します。 (例) ホスト名 [:ポート番号] これは、「ホスト名」と指定するか、または「ホスト名:ポート番号」と指定することを示します。
	この記号で仕切られた項目は選択できることを示します。

文法記述記号	意味
	(例) DCCLTTREXPSP=Y N これは、DCCLTTREXPSP=Y と指定するか、または DCCLTTREXPSP=N と指定することを示します。
…	記述が省略されていることを示します。この記号の直前に示された項目を繰り返し複数個指定できます。 (例) ホスト名 [:ポート番号] [,ホスト名 [:ポート番号] ,…] ホスト名 [:ポート番号] を続けて指定できることを示します。

(2) 属性記述記号

ユーザ指定値の範囲などを説明する記号です。

属性表示記号	意味
~	この記号のあとにユーザ指定値の属性を示します。
《 》	ユーザ指定値の省略値を示します。
〈 〉	ユーザ指定値の構文要素記号を示します。
(())	ユーザ指定値の指定範囲を示します。

(3) 構文要素記号

ユーザ指定値の内容を説明する記号です。

構文要素記号	意味
〈英字〉	アルファベット (A~Z, a~z), および_ (アンダースコア)
〈英数字〉	英字と数字 (0~9)
〈英字記号〉	アルファベット (A~Z, a~z), #, @, および¥
〈符号なし整数〉	数字 (0~9)
〈符号なし 16 進整数〉	数字 (0~9), A~F, a~f
〈記号名称〉	英字記号と数字の並び (先頭は英字記号)
〈文字列〉	任意の文字の配列
〈パス名〉	記号名称, /, および. (ピリオド) (ただし, パス名は使用する OS に依存)

8.2 クライアント環境定義の詳細

この節では、クライアント環境定義の形式、設定方法、およびオペランドについて説明します。

8.2.1 TP1/Client/W の形式

(1) sh (ボーンシェル) の場合

```
$ DCNAMPORT=ネームサービスのポート番号
$ DCHOST=窓口となるTP1/Server
$ DCWATCHTIM=最大応答待ち時間
$ DCCLTCONNECTTIMEOUT=コネクション確立最大監視時間
$ DCCLTTREXPTM=トランザクションブランチ限界経過時間
$ DCCLTTREXPSP=Y|N|F
$ DCCLTTRWATTM=トランザクション問い合わせ間隔最大時間
$ DCCLTTRRCPUTM=トランザクションブランチCPU監視時間
$ DCCLTUTTRCMT=Y|N
$ DCRCVPORT=受信で使用するCUPのポート番号
$ DCSNDHOST=接続先のノード名
$ DCSNDPORT=接続先のポート番号
$ DCSOCKOPENATRCV=Y|N
$ DCCLTDELIVERYCHECK=Y|N
$ DCUTOKEY=テストユーザID
$ DCCACHE=サービス情報を一時的に格納する領域数
$ DCCLTCACHETIM=一時的に格納したサービス情報の有効時間
$ DCCLTLOADBALANCE=Y|N
$ DCCLTSERVICEGROUPLIST=サービスグループとRPC受け付け窓口の対応を定義
    したファイル名
$ DCCLTCONNECTRETRY=コネクション確立の再試行回数
$ DCSCDDIRECT=Y|N
$ DCSCDPORT=スケジューラサービスのポート番号
$ DCCLTDATACOMP=Y|N
$ DCEXTENDFUNCTION=RPCサービスの機能拡張レベル
$ DCCLTINQUIRETIME=常設コネクション問い合わせ間隔最大時間
$ DCCLTPORT=クライアント拡張サービスのポート番号
$ DCCLTDCCMHOST=DCCM3論理端末のホスト名
$ DCCLTDCCMPORT=DCCM3論理端末のポート番号
$ DCCLTXATMI=Y|N
$ DCWATCHTIMINHERIT=Y|N
$ DCCLTDELAY=最大通信遅延時間
$ DCCLTCUPSNDHOST=CUPの送信元ホスト
$ DCCLTCUPRCVPORT=CUPの受信で使用するポート番号
$ DCCLTRAPHOST=rapリスナー, またはDCCM3の論理端末
$ DCCLTRAPAUTOCONNECT=Y|N
$ DCCLTTRSTATISITEM=統計情報項目
$ DCCLTTRROPTIITEM=トランザクション最適化項目
$ DCCLTTRWATCHTIME=トランザクション同期点処理時の最大通信待ち時間
$ DCCLTTRRINFO=no|self|remote|all
$ DCCLTTRLIMITTIME=トランザクションブランチ最大実行可能時間
$ DCCLTTRRBRCV=Y|N
$ DCCLTTRRECOVERYTYPE=type1|type2|type3
$ DCWATCHTIMRPCINHERIT=Y|N
```

```

$ DCSYSWATCHTIM=OpenTP1制御の最大応答待ち時間
$ DCCLTAUTHENT=Y|N
$ DCCLTCONNECTINF=端末識別情報
$ DCSCDMULTI=Y|N
$ DCSCDMULTICOUNT=マルチスケジューラデーモンのプロセス数
$ DCHOSTSELECT=Y|N
$ DCSCDLOADPRIORITY=Y|N
$ DCCLTONLYTHISNODE=Y|N
$ DCCLTNOSERVER=Y|N
$ DHOSTCHANGE=Y|N
$ DCCLTOPTION=クライアントの機能拡張オプション
$ DCCLTNAMEXTEND=0|1
$ DCCLTBACKLOGCOUNT=コネクション確立要求を格納するキューの数
$ DTRCPATH=トレースファイル作成ディレクトリ
$ DTRCERR=エラーログのサイズ
$ DTRCUAP=UAPトレースのファイルサイズ
$ DTRCSOC=ソケットトレースのファイルサイズ
$ DTRCSOCSIZE=ソケットトレースのデータサイズ
$ DTRCMDL=モジュールトレースのファイルサイズ
$ DCCLTPRFINFOSEND=Y|N
$ DCCLTRPCMAXMSGSIZE=RPC送受信電文の最大長
$ DCCLTRECVBUFFSIZE=TCP/IPの受信バッファサイズ
$ DCCLTSENBUFFSIZE=TCP/IPの送信バッファサイズ
$ DCCLTTCPNODELAY=Y|N
$ export DCNAMPORT DHOST DCWATCHTIM DCCLTCONNECTTIMEOUT
    DCCLTTREXPTM DCCLTTREXPSP DCCLTTRWATTM DCCLTTRCPUPTM
    DCCLTUTTRCMT DCRCVPORT DCSNDHOST DCSNDPORT
    DCSOCKOPENATRCV DCCLTDELIVERYCHECK DCUTOKEY
    DCCACHE DCCLTCACHETIM DCCLTLOADBALANCE
    DCCLTSERVICEGROUPLIST DCCLTCONNECTRETRY DCSCDDIRECT
    DCSCDPORT DCCLTDATACOMP DCEXTENDFUNCTION
    DCCLTINQUIRETIME DCCLTPORT DCCLTDCCMHOST
    DCCLTDCCMPORT DCCLTXATMI DCWATCHTIMINHERIT
    DCCLTDELAY DCCLTCUPSNDHOST DCCLTCUPRCVPORT DCCLTRAPHOST
    DCCLTRAPAUTOCONNECT
    DCCLTTRSTATISITEM DCCLTTROPTIITEM DCCLTTRWATCHTIME
    DCCLTTRRINFO DCCLTTRLIMITTIME DCCLTTRRBRV
    DCCLTTRRECOVERYTYPE DCWATCHTIMRPCINHERIT DCSYSWATCHTIM
    DCCLTAUTHENT DCCLTCONNECTINF DCSCDMULTI
    DCSCDMULTICOUNT DCHOSTSELECT DCSCDLOADPRIORITY
    DCCLTONLYTHISNODE DCCLTNOSERVER DHOSTCHANGE DCCLTOPTION
    DCCLTNAMEXTEND DCCLTBACKLOGCOUNT DTRCPATH DTRCERR DTRCUAP
    DTRCSOC DTRCSOCSIZE DTRCMDL DCCLTPRFINFOSEND DCCLTRPCMAXMSGSIZE
    DCCLTRECVBUFFSIZE DCCLTSENBUFFSIZE DCCLTTCPNODELAY

```

(2) csh (Cシェル) の場合

```

% setenv DCNAMPORT   ネームサービスのポート番号
% setenv DHOST      窓口となるTP1/Server
% setenv DCWATCHTIM  最大応答待ち時間
% setenv DCCLTCONNECTTIMEOUT  コネクション確立最大監視時間
% setenv DCCLTTREXPTM  トランザクションブランチ限界経過時間
% setenv DCCLTTREXPSP  Y|N|F
% setenv DCCLTTRWATTM  トランザクション問い合わせ間隔最大時間
% setenv DCCLTTRCPUPTM  トランザクションブランチCPU監視時間
% setenv DCCLTUTTRCMT  Y|N

```

```

% setenv DCRCVPORT 受信で使用するCUPのポート番号
% setenv DCSNDHOST 接続先のノード名
% setenv DCSNDPORT 接続先のポート番号
% setenv DC SOCKOPENATRCV Y|N
% setenv DCCLTDELIVERYCHECK Y|N
% setenv DCUTOKEY テストユーザID
% setenv DCCACHE サービス情報を一時的に格納する領域数
% setenv DCCLTCACHETIM 一時的に格納したサービス情報の有効時間
% setenv DCCLTLOADBALANCE Y|N
% setenv DCCLTSERVICEGROUPLIST サービスグループとRPC受け付け窓口の
    対応を定義したファイル名
% setenv DCCLTCONNECTRETRY コネクション確立の再試行回数
% setenv DCSCDDIRECT Y|N
% setenv DCSCDPORT スケジュールサービスのポート番号
% setenv DCCLTDATAComp Y|N
% setenv DCEXTENDFUNCTION RPCサービスの機能拡張レベル
% setenv DCCLTINQUIRETIME 常設コネクション問い合わせ間隔最大時間
% setenv DCCLTPORT クライアント拡張サービスのポート番号
% setenv DCCLTDCCMHOST DCCM3論理端末のホスト名
% setenv DCCLTDCCMPORT DCCM3論理端末のポート番号
% setenv DCCLTXATMI Y|N
% setenv DCWATCHTIMINHERIT Y|N
% setenv DCCLTDELAY 最大通信遅延時間
% setenv DCCLTCUPSNDHOST CUPの送信元ホスト
% setenv DCCLTCUPRCVPORT CUPの受信で使用するポート番号
% setenv DCCLTRAPHOST rapリスナー, またはDCCM3の論理端末
% setenv DCCLTRAPAUTOCONNECT Y|N
% setenv DCCLTTRSTATISITEM 統計情報項目
% setenv DCCLTTROPTIITEM トランザクション最適化項目
% setenv DCCLTTRWATCHTIME トランザクション同期点処理時の
    最大通信待ち時間
% setenv DCCLTTRRINFO no|self|remote|all
% setenv DCCLTTRLIMITTIME トランザクションブランチ最大実行可能時間
% setenv DCCLTTRRRCV Y|N
% setenv DCCLTTRRECOVERYTYPE type1|type2|type3
% setenv DCWATCHTIMRPCINHERIT Y|N
% setenv DCSYSWATCHTIM OpenTP1制御の最大応答待ち時間
% setenv DCCLTAUTHENT Y|N
% setenv DCCLTCONNECTINF 端末識別情報
% setenv DCSCDMULTI Y|N
% setenv DCSCDMULTICOUNT マルチスケジューラデーモンのプロセス数
% setenv DCHOSTSELECT Y|N
% setenv DCSCDLOADPRIORITY Y|N
% setenv DCCLTONLYTHISNODE Y|N
% setenv DCCLTNOSERVER Y|N
% setenv DCHOSTCHANGE Y|N
% setenv DCCLTOPTION クライアントの機能拡張オプション
% setenv DCCLTNAMEXTEND 0|1
% setenv DCCLTBACKLOGCOUNT コネクション確立要求を格納するキューの数
% setenv DCTRCPATH トレースファイル作成ディレクトリ
% setenv DCTRCERR エラーログのサイズ
% setenv DCTRCUAP UAPトレースのファイルサイズ
% setenv DCTRCSOC ソケットトレースのファイルサイズ
% setenv DCTRCOCsize ソケットトレースのデータサイズ
% setenv DCTRCMDL モジュールトレースのファイルサイズ
% setenv DCCLTPRFINFOSEND Y|N
% setenv DCCLTRPCMAXMSGsize RPC送受信電文の最大長
% setenv DCCLTRECvBUFSIZE TCP/IPの受信バッファサイズ

```

```
% setenv DCCLTSENDBUFSIZE TCP/IPの送信バッファサイズ
% setenv DCCLTTCNodelay Y|N
```

8.2.2 TP1/Client/P の形式

```
[betran]※
dcnamport=ネームサービスのポート番号
dchost=窓口となるTP1/Server
dcwatchtim=最大応答待ち時間
dccltconnecttimeout=コネクション確立最大監視時間
dcclttrexpmt=トランザクションブランチ限界経過時間
dcclttrexp= Y|N|F
dcclttrwattm=トランザクション問い合わせ間隔最大時間
dcclttrrcputm=トランザクションブランチCPU監視時間
dccltuttrcmt=Y|N
dcrvcport=受信で使用するCUPのポート番号
dcsndhost=接続先のノード名
dcsndport=接続先のポート番号
dcsockopenatrcv=Y|N
dccltdeliverycheck=Y|N
dcutokey=テストユーザID
dccache=サービス情報を一時的に格納する領域数
dccltcachetim=一時的に格納したサービス情報の有効時間
dccltloadbalance=Y|N
dccltserVICegrouplist=サービスグループとRPC受け付け窓口の対応を定義
                        したファイル名
dccltconnectretry=コネクション確立の再試行回数
dcsddirect=Y|N
dcsdport=スケジュールサービスのポート番号
dccltdatacomp=Y|N
dcextendfunction=RPCサービスの機能拡張レベル
dccltinquiretime=常設コネクション問い合わせ間隔最大時間
dccltport=クライアント拡張サービスのポート番号
dccltdccmhost=DCCM3論理端末のホスト名
dccltdccmport=DCCM3論理端末のポート番号
dccltxatmi=Y|N
dcwatchtiminherit=Y|N
dccltdelay=最大通信遅延時間
dccltcupsndhost=CUPの送信元ホスト
dccltcuprcvport=CUPの受信で使用するポート番号
dcclttraphost=rapリスナー, またはDCCM3の論理端末
dcclttrapautoconnect=Y|N
dcclttrstatisitem=統計情報項目
dcclttrroptiitem=トランザクション最適化項目
dcclttrwatchtime=トランザクション同期点処理時の最大通信待ち時間
dcclttrrbinfo=no|self|remote|all
dcclttrlimittime=トランザクションブランチ最大実行可能時間
dcclttrrbrcv=Y|N
dcclttrrecoverytype=type1|type2|type3
dcwatchtimrpcinherit=Y|N
dcsyswatchtim=OpenTP1制御の最大応答待ち時間
dccltauthent=Y|N
dccltconnectinf=端末識別情報
dcsdmulti=Y|N
```

```
dcscdmulticount=マルチスケジューラデーモンのプロセス数
dchostselect=Y|N
dcscdloadpriority=Y|N
dccltonlythisnode=Y|N
dccltnoserver=Y|N
dchselint=受信チェック間隔時間
dchostchange=Y|N
dccltoption=クライアントの機能拡張オプション
dccltnamextend=0|1
dccltbacklogcount=コネクション確立要求を格納するキューの数
dctrpath=トレースファイル作成ディレクトリ
dctrerr=エラーログのサイズ
dtrcuap=UAPトレースのファイルサイズ
dtrcsoc=ソケットトレースのファイルサイズ
dtrcsocsize=ソケットトレースのデータサイズ
dtrcmdl=モジュールトレースのファイルサイズ
dccltprfinfoSEND=Y|N
dccltRPCmaxmsgsize=RPC送受信電文の最大長
dccltrecvbufsize=TCP/IPの受信バッファサイズ
dccltsendbufsize=TCP/IPの送信バッファサイズ
dcclttcpnodelay=Y|N
```

注※

TP1/Client/P の定義開始宣言で、省略できません。[] は省略を示す記号ではありません。
それ以降は省略できます。

Windows ディレクトリの betran.ini ファイルに上記のクライアント環境定義を指定します。

_s 付き関数を使用する場合は、任意のファイルを作成し、そのファイルをクライアント環境定義ファイルとして使用できます。この場合、dc_clt_cltin_s 関数の引数 defpath に、完全パス、またはカレントドライブ・ディレクトリからのパス名で指定します。

サーバからの一方通知受信機能を使用する場合は、dc_clt_accept_notification_s 関数、dc_clt_cancel_notification_s 関数、または dc_clt_open_notification_s 関数に、任意のクライアント環境定義ファイルを指定できます。

8.2.3 TP1/Client/W の設定方法

クライアント環境定義は、環境変数、またはクライアント環境定義ファイルで定義します。

環境変数の場合、「8.2.1 TP1/Client/W の形式」の形式で設定します。環境変数の設定方法は、クライアントマシンの OS、および使用するシェルによって異なります。一般的に、環境変数は、sh (ボーンシェル) の場合は/etc/profile、または\$HOME/.profile に、csh (C シェル) の場合は/etc/cshrc、または\$HOME/.cshrc に格納します。

クライアント環境定義ファイルの場合、任意のファイルを作成し、そのファイル中に「8.2.2 TP1/Client/P の形式」の形式で定義します。このファイルは、次に示す関数の引数 defpath に絶対パス名、または相対パス名で指定し、定義を読み込みます。

- dc_clt_cltin_s 関数
- dc_clt_accept_notification_s 関数
- dc_clt_cancel_notification_s 関数
- dc_clt_open_notification_s 関数

引数 defpath にクライアント環境定義ファイルを指定した場合、環境変数に設定されている定義はすべて無効となります。クライアント環境定義ファイルに指定されていない定義はデフォルト値で動作します。引数 defpath に NULL を指定した場合、環境変数の設定で動作します。環境変数が設定されていない場合はデフォルト値で動作します。

同一マシンで CUP を複数実行する場合、CUP ごとに環境変数を設定、またはクライアント環境定義ファイルを作成してください。次に示すクライアント環境定義は、CUP ごとに異なる値を指定する必要があります。

- DCRCVPORT
- DCCLTCUPRCVPORT
- DCTRCPATH

マルチスレッドで動作する CUP でスレッドごとにクライアント環境定義を変更したい場合、スレッドごとにクライアント環境定義ファイルを作成し、各スレッドで発行する定義ファイルを読み込む関数の引数 defpath に指定するパス名を変更してください。

8.2.4 TP1/Client/P の設定方法

クライアント環境定義は、クライアント環境定義ファイルで定義します。

betran.ini という名称のファイルをクライアント環境定義ファイルとして Windows ディレクトリ配下 (%WINDIR%) に作成します。そのファイル中に「8.2.2 TP1/Client/P の形式」で示す形式でクライアント環境定義を定義します。また、任意のファイルを作成し、そのファイルをクライアント環境定義ファイルとして使用できます。この場合、次に示す関数の引数 defpath に、絶対パス名、または相対パス名で指定し、定義を読み込みます。

- dc_clt_cltin_s 関数
- dc_clt_accept_notification_s 関数
- dc_clt_cancel_notification_s 関数
- dc_clt_open_notification_s 関数

クライアント環境定義は、betran.ini ファイル、および任意のクライアント環境定義ファイルどちらのファイルに指定しても有効です。両方のファイルに同じ定義を異なる値で指定した場合は、クライアント環境定義ファイルに指定した値が有効となります。betran.ini ファイル、および任意のクライアント環境定義ファイルのどちらにも指定がない定義は、デフォルト値で動作します。

同一マシンで CUP を複数実行する場合、CUP ごとにクライアント環境定義ファイルを作成してください。次に示すクライアント環境定義は、CUP ごとに異なる値を指定する必要があります。

- DCRCVPORT
- DCCLTCUPRCVPORT
- DCTRCPATH

マルチスレッドで動作する CUP でスレッドごとにクライアント環境定義を変更したい場合、スレッドごとにクライアント環境定義ファイルを作成してください。各スレッドで発行する定義ファイルを読み込む関数の引数 defpath に指定するパス名を変更してください。

8.2.5 TP1/Client/W と TP1/Client/P で共通のオペランド

Windows の場合はオペランドは英字小文字で指定します。

TP1/Client/W でだけ使用できるオペランドについては、「DCCLTBACKLOGCOUNT オペランド」を、また、TP1/Client/P でだけ使用できるオペランドについては、「8.2.6 TP1/Client/P 固有のオペランド」を参照してください。

●DCNAMPORT=ネームサービスのポート番号

～ 〈符号なし整数〉 ((5001～65535)) 《10000》

ネームサービスのポート番号を指定します。通信相手の TP1/Server は、この定義で指定したポート番号でネームサービスを起動する必要があります。

ネームサービスのポート番号はクライアント環境定義 DCHOST に指定することもできます。

●DCHOST=窓口となる TP1/Server ～ 〈文字列〉

窓口となる TP1/Server のホスト名およびポート番号を指定します。区切り文字','を使用して、複数の窓口となる TP1/Server を指定できます。

形式

ホスト名 [:ポート番号] [,ホスト名 [:ポート番号] ,…]

- ホスト名 ～ 〈文字列〉
- ポート番号 ～ 〈符号なし整数〉 ((5001～65535))

ホスト名として指定できる長さは、63 文字までです。クライアント環境定義 DCCLTOPTION に 00000008 を指定した場合、255 文字までです。なお、このオペランドに指定できる長さは、1023 文字までです。

区切り文字','の後ろ以外は空白文字（スペースまたはタブ）を入れないでください。

ホスト名として、10 進ドット記法の IP アドレスを指定することもできます。

ポート番号省略時は、クライアント環境定義 DCNAMPORT の値が仮定されます。

DCHOST に TP1/Server を二つ以上指定し、窓口となる TP1/Server の障害を検出した場合、クライアント環境定義 DCHOSTSELECT が N のとき、現在窓口となっている TP1/Server の次の TP1/

Server を参照して切り替えを試みます。クライアント環境定義 DCHOSTSELECT が Y のとき、障害を検出した TP1/Server を除いて、窓口となる TP1/Server をランダムに選択して切り替えを試みます。クライアントユーザの認証要求時に窓口となる TP1/Server を指定した場合、クライアント環境定義 DCHOST は無効になります。クライアントユーザの認証要求時に窓口となる TP1/Server を指定していない場合でクライアント環境定義 DCHOST を指定していないときは、TP1/Client がブロードキャストを行い、窓口となる TP1/Server を決定します。TP1/Client/P でブロードキャストを行う場合、hosts ファイルにブロードキャストアドレスを指定する必要があります (ホスト名は broadcast としてください)。

●DCWATCHTIM=最大応答待ち時間

～ 〈符号なし整数〉 ((0~65535)) 《180》 (単位：秒)

応答型 RPC の場合に、CUP から SPP へサービス要求を送ってからサービスの応答が返るまでの待ち時間の最大値を指定します。

指定時間を過ぎても応答が返らない場合は、CUP へエラーリターンします。

0 を指定した場合は、応答を受信するまで無限に待ち続けます。

●DCCLTCONNECTTIMEOUT=コネクション確立最大監視時間

～ 〈符号なし整数〉 ((0~65535)) 《0》 (単位：秒)

データ送信時の、ノンブロッキングモードのコネクション確立に対する最大監視時間を指定します。

0 を指定した場合、またはこの指定を省略した場合は、コネクション確立はブロッキングモードで行われ、コネクション確立の監視処理は OS によって行われます。

定義の指定に誤りがあった場合は、エラーログに KFCA02401-E メッセージを出力し、dc_clt_cltin_s 関数が DCCLTER_FATAL でエラーリターンします。

相手システム未起動などの要因でコネクションの確立ができない場合、ここで指定した監視時間が経過する前に CUP から発行した関数がエラーリターンする場合があります。これは、ここで指定した最大監視時間よりも OS によるコネクション確立処理の監視時間が優先されるためです。OS によるコネクション確立処理の監視時間 (コネクション確立要求の再送回数, 間隔) はプラットフォームによって異なります。

この定義で指定する値は、TP1/Client が提供する関数で実行されるコネクションの確立に掛かる最大監視時間です。関数の処理時間ではないので注意してください。使用する関数や機能によっては、関数の処理時間がこの定義で指定した値よりも大きくなる場合があります。

●DCCLTTREXPTM=トランザクションブランチ限界経過時間

～ 〈符号なし整数〉 ((0~65535)) (単位：秒)

トランザクションブランチの処理時間の最大値を指定します。CUP からトランザクションを開始する場合だけ有効です。

指定時間を超えてもトランザクションブランチが完了しないとき、そのトランザクションブランチのプロセスを異常終了させて、ロールバックします。0 を指定した場合は、時間監視をしません。

この指定を省略した場合、クライアントサービス定義の trn_expiration_time オペランドの指定に従います。また、接続先が rap サーバの場合、rap リスナーサービス定義の trn_expiration_time オペランドの指定に従います。

なお、RPC 機能を使用した場合に、他プロセスで実行するトランザクションブランチの処理時間も監視時間を含むかどうかは、DCCLTTREXPSP で指定してください。

●DCCLTTREXPSP=Y | N | F

トランザクションブランチの処理を監視するとき、「トランザクショナル RPC 実行プロセスのトランザクションブランチが、RPC 機能を使ってほかのトランザクションブランチを呼び出し、その処理が終わるのを待つ時間」を監視時間を含むかどうかを指定します。

Y：監視時間を含みます。

N または F：監視時間を含みません。

この指定を省略した場合は、クライアントサービス定義の `trn_expiration_time_suspend` オペランドの指定に従います。また、接続先が rap サーバの場合、rap リスナーサービス定義の `trn_expiration_time_suspend` オペランドの指定に従います。

●DCCLTTRWATTM=トランザクション問い合わせ間隔最大時間

～ 〈符号なし整数〉 ((1～65535)) 《180》 (単位：秒)

トランザクションの処理で、CUP がサーバに問い合わせ（トランザクションを制御する関数、または `dc_rpc_call_s` 関数の実行）をしてから次の問い合わせをするまでの間隔の最大時間を指定します。CUP からトランザクションを開始する場合だけ有効です。

トランザクション問い合わせ間隔最大時間は、トランザクションブランチ限界経過時間より小さい値を指定してください。

指定時間を超えても問い合わせがない場合、サーバ側で処理しているトランザクションのプロセスをロールバックします。

●DCCLTTRCPU TM=トランザクションブランチ CPU 監視時間

～ 〈符号なし整数〉 ((0～65535)) (単位：秒)

トランザクションブランチが同期点処理までに使用できる CPU 時間を指定します。CUP からトランザクションを開始する場合だけ有効です。

0 を指定した場合、CPU 時間を監視しません。

指定時間を超えた場合は、そのトランザクションブランチのプロセスを異常終了させ、ロールバックします。

この指定を省略した場合は、クライアントサービス定義の `trn_cpu_time` オペランドの指定に従います。また、接続先が rap サーバの場合、rap リスナーサービス定義の `trn_cpu_time` オペランドの指定に従います。

Windows 版 TP1/Server では、CPU 監視時間の指定は、サポートしていません。窓口となる TP1/Server が Windows 版の場合、このオペランドは無効になります。

●DCCLTUTTRCMT=Y | N ~ 《N》

オンラインテストの機能を使って CUP からトランザクションを開始させた場合、トランザクションを同期点でコミットするかロールバックするかを指定します。

Y：コミットします。

N：ロールバックします。

●DCRCVPORT=受信で使用する CUP のポート番号

～ 〈符号なし整数〉 ((1~65535)) 《11000》

TCP/IP 通信機能を使用してメッセージを受信する場合、メッセージを受信する CUP のポート番号を指定します。メッセージを送信する側では、このポート番号を指定して送信してください。なお、同一マシン内で、複数のプロセス、または複数のスレッドを同時に実行する場合は、それぞれ異なるポート番号を指定してください。

指定できるポート番号でも、OS またはほかのプログラムで使用するポート番号は指定しないでください。指定した場合、応答データを正しく受信できないことがあります。なお、OS が使用するポート番号は、OS ごとに異なります。OS のマニュアルなどを参照してください。ほかのアプリケーションとの重複を避けるため、OS が任意に割り当てるポート番号（動的ポートまたは短命ポートと呼ばれるポート番号）を使用しないでください。

メッセージの一方送信機能（`dc_rpc_open_s` 関数の引数 `flags` に `DCCLT_ONEWAY_SND` を指定）の場合、デフォルト値を含めこの定義で指定したポートは使用されません。

●DCSNDHOST=接続先のノード名

～ 〈文字列〉

TCP/IP 通信機能を使用してメッセージを送信する場合、コネクションを確立して接続するノードのホスト名を指定します。

ホスト名として指定できる長さは、63 文字までです。クライアント環境定義 `DCCLTOPTION` に `00000008` を指定した場合、255 文字までです。

また、ホスト名として、10 進ドット記法の IP アドレスを指定することもできます。

●DCSNDPORT=接続先のポート番号

～ 〈符号なし整数〉 ((1~65535)) 《12000》

TCP/IP 通信機能を使用してメッセージを送信する場合、コネクションを確立して接続するノードのポート番号を指定します。

●DCSOCKOPENATRCV=Y | N ~ 《N》

TCP/IP 通信機能使用時に、メッセージの送受信をする場合の受信用ソケットの開設契機（送信相手からの接続を待ち受け始める契機）を指定します。この定義は、`dc_rpc_open_s` 関数の引数 `flags` に `DCCLT_SNDRCV` を指定したときだけ有効です。

MHP がサーバ型でメッセージの送受信を一つのコネクションで行う場合、CUP から MHP にメッセージを送信した際に確立したコネクションを使用して送受信を行うため、CUP は受信用のソケットは使用しません。この定義に Y を指定し、受信用ソケットを開設しないようにしてください。

MHP がクライアント型の場合、MHP からのメッセージを CUP が受信する際に受信用のソケットが必要なため、この定義に N を指定するか定義を省略してください。

この定義の具体的なユースケースは「[2.5.5 ユースケースごとの設定方法とポートの割り当て](#)」を参照してください。

Y：`dc_rpc_open_s` 関数の実行時に受信ソケットを開設しません。ただし、次の関数を実行したときに、コネクションが確立されていない場合は受信用ソケットを開設します。

- `dc_clt_receive_s` 関数

- dc_clt_receive2_s 関数
- dc_clt_assem_receive_s 関数

N: dc_rpc_open_s 関数実行時に、受信用ソケットを開設します。

●DCCLTDELIVERYCHECK=Y | N ~ 《N》

メッセージの送達確認機能を使用するかどうかを指定します。

Y: メッセージの送達確認機能を使用します。

N: メッセージの送達確認機能を使用しません。

メッセージの送達確認機能を使用すると、dc_clt_assem_send_s 関数発行時は、メッセージを送信したあと、応答用専用データを受信してリターンします。dc_clt_assem_receive_s 関数発行時は、メッセージを受信したあと、応答用データを送信してリターンします。

この定義に N を指定し、かつ dc_clt_assem_send_s 関数または dc_clt_assem_receive_s 関数を発行した場合は、メッセージの組み立て機能を使用した送受信となります。

●DCUTOKEY=テストユーザ ID

~ 〈1~4 文字の英数字〉

オンラインテスト機能を使用して CUP を実行するときに指定します。

テストユーザ ID を指定すると、CUP から起動される SPP はテストモードで実行できます。

ただし、次の場合はテストモードで実行できません。

- リモート API 機能を使用する場合
- クライアント拡張サービスを使用する場合

●DCCACHE=サービス情報を一時的に格納する領域数

~ 〈符号なし整数〉 ((2~10240)) 《8》

クライアントから RPC を実行するときに、窓口となる TP1/Server のネームサービスに問い合わせたサービス情報を格納するキャッシュ領域の数を指定します。一つのキャッシュ領域に一つのサービス情報が格納されます。

この定義は次の値を目安にして指定してください。

DCCLTLOADBALANCE=N の場合

クライアントから RPC 要求を実行するサーバ数を指定してください。

一度格納したサービス情報は、dc_rpc_close_s 関数を発行するまで、または空きキャッシュ領域が無いために別のサービス情報に上書きされるまで有効です。

DCCLTLOADBALANCE=Y の場合

マルチノードサーバで起動しているノード内のすべてのサーバ数を指定してください。

一度格納したサービス情報は、dc_rpc_close_s 関数を発行するまで、空きキャッシュ領域が無いために別のサービス情報に上書きされるまで、またはクライアント環境定義 DCCLTCACHETIM に指定した有効時間が過ぎるまで有効です。

一つのキャッシュ領域に対して消費するメモリ量は約 150 バイトです。なお、クライアント環境定義に「DCSCDDIRECT=Y」を指定している場合は、この定義は無効になります。

●DCCLTCACHETIM=一時的に格納したサービス情報の有効時間

～ 〈符号なし整数〉 ((0~65535)) 《30》 (単位：秒)

窓口となる TP1/Server のネームサービスに問い合わせ取得したサービス情報の、有効時間を指定します。有効時間が過ぎたサービス情報はキャッシュ領域から削除されます。0 を指定した場合は、有効時間はありません。一度格納したサービス情報は dc_rpc_close_s 関数を発行するまで、または空きキャッシュ領域が無いために別のサービス情報に上書きされるまで有効です。なお、この定義はクライアント環境定義に「DCCLTLOADBALANCE=Y」を指定している場合にだけ有効です。クライアント環境定義に「DCSCDDIRECT=Y」を指定している場合は、この定義の指定は無効になります。

●DCCLTLOADBALANCE=Y | N ~ 《N》

マルチノードサーバでの運用時、RPC 実行時に各ノードの負荷状態を TP1/Client の内部で評価して、負荷の軽いサーバに分散させる機能（ノード間負荷バランス機能）を使用するかどうかを指定します。

Y：ノード間負荷バランス機能を使用します。

N：ノード間負荷バランス機能を使用しません。

この定義に Y を指定する場合は、ノード数や RPC 要求を行うサーバ数に応じて、次のクライアント環境定義を指定してください。

- DCCACHE
- DCCLTCACHETIM
- DCCLTNAMEXTEND

この定義は、クライアント環境定義 DCSCDDIRECT=Y のときは無効です。

●DCCLTSERVICEGROUPLIST=サービスグループと RPC 受け付け窓口の対応を定義したファイル名
～ 〈文字列〉

OpenTP1 以外のサーバへ RPC を行う場合に使用する、サービスグループと、そのサーバの RPC 受け付け窓口との対応関係を定義した、テキストファイルの名前（パス名も含む）を指定します。

なお、このファイルには次の形式で定義してください。

サービスグループ名 サーバのホスト名 窓口のポート番号 [,サーバのホスト名 窓口のポート番号, …] [コメント]

テキストファイルの 1 行に、サービスグループと RPC 受け付け窓口を指定します。RPC 受け付け窓口は、複数指定できます。

サービスグループを複数指定する場合、複数行に分けて指定します。

各項目の間は、空白、またはタブ記号で区切ります。

各項目は次のように指定します。

サービスグループ名

31 文字以内の任意の文字列指定します。

サーバのホスト名

接続先となるホストのホスト名を指定します。ホスト名として指定できる長さは、63 文字までです。クライアント環境定義 DCCLTOPTION に 00000008 を指定した場合、255 文字までです。10 進ドット記法の IP アドレスを指定することもできます。

窓口のポート番号

RPC を受け付ける窓口のポート番号を数値で指定します (1~65535)。

コメント

先頭に"#"を置き、コメントを記述できます。"#以降、行末までのすべての文字は無視されます。また、コメントは省略できます。

RPC 受け付け窓口を複数指定した場合は、その中から一つをランダムに選択し、接続を試みます。接続を試みた先のホストとの接続に失敗したとき、この接続先を除き再びランダムに選択し、接続を試みます。これを繰り返し、すべての RPC 受け付け窓口との接続に失敗した場合、dc_rpc_call_s 関数はエラーを返します。

ファイルの内容の評価

ファイル中の定義内容が不正の場合、TP1/Client は不正であった行の内容を無視し、無視した行の行番号をエラーログへ出力します (KFCA02431-E メッセージ)。

同じサービスグループを複数行に分けて指定した場合、最初の行に指定した内容が有効となります。

dc_rpc_call_s 関数で呼び出したサービスグループ名がこのオペランドに指定したファイルに定義されていない場合、クライアント環境定義 DCCLTNOSERVER の指定によって次のように動作が異なります。

- DCCLTNOSERVER=Y の場合
即時に DCRPCER_NO_SUCH_SERVICE_GROUP となります。
- DCCLTNOSERVER=N, または指定を省略している場合
dc_rpc_call_s 関数で呼び出したサービスグループ名がこのオペランドに指定したファイルに定義されていないことを認識したあとに、TP1/Server へ RPC を行います。

●DCCLTCONNECTRETRY=コネクション確立の再試行回数

~ 〈符号なし整数〉 ((0~255)) 《0》

サーバとの回線が途絶えている、サーバマシンの電源がオフであるなどの理由で、コネクション確立要求がタイムアウトした場合に、再試行する回数を指定します。0 を指定した場合、またはこの指定を省略した場合は、再試行しません。指定に誤りがあった場合、0 が仮定されます。この定義は、ユーザ認証時 (DCCLTAUTHENT=Y を指定し、かつ dc_clt_cltin_s 関数の flags 引数に DCNOFLAGS を指定して実行した場合) に有効です。

●DCSCDDIRECT=Y | N ~ 《N》

TP1/Server のネームサービスにサービス情報を問い合わせないで、直接スケジュールサービスに問い合わせる機能 (ネームサービスを使用しない RPC) を使用するかどうかを指定します。

Y: ネームサービスを使用しない RPC を使用します。

N: ネームサービスを使用しない RPC を使用しません。

クライアント環境定義 DCSCDPORT にスケジュールサービスのポート番号を指定したときは、そのポート番号を使用して問い合わせを行います。DCSCDPORT が定義されていないときは、TP1/Server 側からスケジュールサービスのポート番号を取得したあと、問い合わせを行います。

この機能を使用した場合はソケット受信型の SPP を呼ぶことはできません。

また、クライアント環境定義 DCCACHE, DCCLTCACHETIM, DCCLTLOADBALANCE の定義は無視されます。なお、クライアント環境定義 DCCLTSERVICEGROUPLIST が設定されているときは、DCSCDDIRECT の定義は無視されます。

●DCSCDPORT=スケジュールサービスのポート番号

～ 〈符号なし整数〉 ((5001～65535))

スケジュールサービスのポート番号を指定します。クライアント環境定義 DCSCDMULTI=Y および DCSCDDIRECT=Y の場合は、マルチスケジューラデーモンのポート番号を指定します。マスタスケジューラデーモンとマルチスケジューラデーモンのベースとなるポート番号が連続している場合は、マスタスケジューラデーモン (=スケジュールサービス) のポート番号も指定できます。通信相手の TP1/Server は、この定義で指定したポート番号で、スケジュールサービスまたはマルチスケジューラデーモンを起動する必要があります。TP1/Server のスケジュールサービスまたはマルチスケジューラデーモンの指定については、マニュアル「OpenTP1 システム定義」を参照してください。

なお、窓口となる TP1/Server を複数指定した場合、指定したすべてのホストで、この定義に指定したポート番号でスケジュールサービスまたはマルチスケジューラデーモンを起動する必要があります。ホスト切り替え機能によってホストが切り替わったとき、指定したポート番号で起動していないと、再びホストが切り替わります。

この定義は、クライアント環境定義 DCSCDDIRECT=Y のときだけ有効です。

クライアント環境定義 DCSCDMULTI=Y および DCSCDDIRECT=Y の場合は、クライアント環境定義 DCSCDMULTICOUNT もあわせて参照してください。

この指定を省略した場合は、ネームサービスに、スケジュールサービスまたはマルチスケジューラデーモンの、ポート番号を問い合わせます。

●DCCLTDATACOMP=Y | N ~ 《N》

データ圧縮機能を使用するかどうかを指定します。

DCCLTDATACOMP=Y を指定した場合でも、サービス要求を受信する SPP のメッセージ格納バッファプール長 (message_store_bufllen オペランド) は、圧縮前のユーザデータ長でサイズを計算してください。

Y: データ圧縮機能を使用します。

N: データ圧縮機能を使用しません。

●DCEXTENDFUNCTION=RPC サービスの機能拡張レベル

～ 〈符号なし 16 進整数〉 ((00000000～00000001)) 《00000000》

RPC サービスの機能の拡張レベルを、次の中から指定します。0 の数を省略しないで指定してください。なお、指定できない値を指定した場合でも、エラーとならないで誤動作してしまうことがあります。

00000000

RPC サービスの機能を拡張しません。

00000001

サービス要求実行時の SPP が異常終了した場合に、dc_rpc_call_s 関数、または dc_rpc_call_to_s 関数で、エラーを切り分けられるエラーコード (DCRPCER_SERVICE_TERMINATED) を返し

ます。この指定をしなかった場合は、DCRPCER_TIMED_OUT（無限に待ち続ける指定の場合 DCRPCER_SERVICE_NOT_UP）を返します。

なお、常設コネクション確立中、またはトランザクションの範囲内でサービス要求（dc_rpc_call_s 関数）を実行した場合は、この定義は無効となります。常設コネクション確立中、またはトランザクションの範囲内では、ユーザサービスデフォルト定義の rpc_extend_function オペランドの指定が有効となります。

●DCCLTINQUIRETIME=常設コネクション問い合わせ間隔最大時間

～ 〈符号なし整数〉 ((0~1048575)) (単位：秒)

CUP がサーバに対して問い合わせをしてから、次の問い合わせをするまでの間隔の最大時間を指定します。常設コネクション問い合わせ間隔最大時間は CUP 実行プロセスまたは rap サーバで監視するタイマです。指定時間を超えても問い合わせがない場合、CUP 実行プロセス側または rap サーバ側で強制的に常設コネクションを解放します。

なお、DCCM3 の論理端末と常設コネクションを確立している場合、この定義は無効となります。DCCM3 では、この定義と同様の監視を端末放置監視時間値で行っています。端末放置監視時間値はデータコミュニケーション定義の TERMINAL 文の LEFTLIMIT 句で指定します。

また、トランザクション内で常設コネクション問い合わせ間隔最大時間に達したことを検知した場合は、該当するトランザクションを強制的にロールバックします。

この定義に 0 を指定した場合は、CUP からの問い合わせを無限に待ちます。この指定を省略した場合、クライアントサービス定義の clt_inquire_time オペランドの指定に従います。また、接続先が rap サーバの場合、rap リスナーサービス定義の rap_inquire_time オペランドの指定に従います。

この定義で指定する問い合わせ間隔最大時間は、常設コネクションを確立した時に、dc_clt_connect_s 関数から dc_clt_disconnect_s 関数までの問い合わせ間隔を監視します。これに対して、トランザクション問い合わせ間隔最大時間 (DCCLTTRWATTM) は常設コネクションを確立しないで dc_trn_begin_s 関数を呼び出した時に、dc_trn_unchained_commit_s 関数までの問い合わせ間隔を監視します。

●DCCLTPORT=クライアント拡張サービスのポート番号

～ 〈符号なし整数〉 ((5001~65535))

クライアント拡張サービスのポート番号を指定します。通信相手の TP1/Server は、この定義で指定したポート番号でクライアント拡張サービスを起動する必要があります。クライアント拡張サービスのポート番号は、クライアントサービス定義の clt_port オペランドで指定してください。

この指定を省略した場合、クライアント拡張サービスのポート番号をネームサービスに問い合わせます。

●DCCLTDCCMHOST=DCCM3 論理端末のホスト名

DCCM3 の論理端末に対して常設コネクションの確立要求をする場合に、確立要求先の論理端末のホスト名を指定します。この場合、dc_clt_connect_s 関数の引数 flags には DCCLT_DCCM3 を指定する必要があります。

形式は次のとおりです。

ホスト名: [ポート番号] [,ホスト名 [:ポート番号] ,...]

- ホスト名 ~ 〈文字列〉

- ポート番号 ~ 〈符号なし整数〉 ((1~65535))

ホスト名として指定できる長さは、63 文字までです。クライアント環境定義 DCCLTOPTION に 00000008 を指定した場合、255 文字までです。なお、このオペランドに指定できる長さは、1023 文字までです。

区切り文字','の後ろ以外は空白文字（スペースまたはタブ）を入れないでください。

ホスト名として、10 進ドット記法の IP アドレスを指定することもできます。

ポート番号省略時は、クライアント環境定義 DCCLTDCCMPORT に指定されている DCCM3 論理端末のポート番号が仮定されます。

複数の DCCM3 論理端末を指定した場合、その中から一つをランダムに選択し、接続を試みます。接続を試みた先の DCCM3 論理端末との接続に失敗したとき、この接続先を除いて再びランダムに選択し、接続を試みます。これを繰り返し、すべての DCCM3 論理端末との接続に失敗した場合、dc_clt_connect_s 関数はエラーを返します。

常設コネクションを確立して DCCM3 の論理端末と通信する場合、クライアント環境定義 DCCLTSERVICEGROUPLIST に指定した値は無視されます。また、データ圧縮機能は使用できません。

●DCCLTDCCMPORT=DCCM3 論理端末のポート番号

~ 〈符号なし整数〉 ((1~65535)) 《30000》

CUP が DCCM3 の論理端末に対して常設コネクションの接続要求をする場合に使用するポート番号を指定します。

●DCCLTXATMI=Y | N ~ 《N》

XATMI インタフェースを使用して通信をするかどうかを指定します。指定に誤りがあった場合、N が仮定されます。

Y：XATMI インタフェースを使用します。

N：XATMI インタフェースを使用しません。

●DCWATCHTIMINHERIT=Y | N ~ 《N》

トランザクション制御、およびコネクション制御を行う場合、クライアント拡張サービスに CUP の最大応答待ち時間を引き継ぐかどうかを指定します。

Y：クライアント拡張サービスに CUP の最大応答待ち時間を引き継ぎます。

N：クライアント拡張サービスに CUP の最大応答待ち時間を引き継ぎません。

この定義に Y を指定する場合は、クライアント環境定義 DCCLTDELAY を参照してください。

●DCCLTDELAY=最大通信遅延時間

~ 〈符号なし整数〉 ((0~65535)) 《0》 (単位：秒)

CUP とクライアント拡張サービス間の通信オーバーヘッドを考慮し、サーバ側の応答監視をクライアント側よりも早く終わらせる場合に指定します。この定義を指定すると、サーバ側の監視を指定した時間分だけ早く終了させ、クライアント側の監視時間のタイムアウトによるメッセージのすれ違いを防ぎます。

この定義は、クライアント環境定義 DCWATCHTIMINHERIT=Y の場合だけ有効です。クライアント環境定義 DCWATCHTIM=0 の場合は、DCCLTDELAY の定義は無視されます。DCWATCHTIM

に指定した値から DCCLTDELAY に指定した値を引いたときに、0 または負の値になった場合は、DCCLTDELAY の定義は無視され、1 が仮定されます。

クライアント環境定義 DCWATCHTIM に指定した値は、dc_rpc_set_watch_time_s 関数を呼び出すことで動的に変更できます。動的に変更された場合は、DCWATCHTIM に指定した値の代わりに、dc_rpc_set_watch_time_s 関数で動的に変更した値が算出されます。

●DCCLTCUPSNDHOST=CUP の送信元ホスト ～〈文字列〉

コネクション確立要求時の、送信元ホストを指定します。

ホスト名として指定できる長さは、63 文字までです。クライアント環境定義 DCCLTOPTION に 00000008 を指定した場合、255 文字までです。

また、ホスト名として、10 進ドット記法の IP アドレスを指定することもできます。

ホスト名として localhost を指定した場合、または 127 で始まる IP アドレスを指定した場合、dc_clt_cltin_s 関数は DCCLTER_FATAL でエラーリターンします。

CUP を実行するマシン上に存在しないホストを指定した場合、通信関数は、DCCLTER_NET_DOWN、または DCRPCER_NET_DOWN でエラーリターンします。

この定義を省略すると、送信元ホストは任意に割り当てられます。

●DCCLTCUPRCVPORT=CUP の受信で使用するポート番号

～〈符号なし整数〉((5001~65535))

サーバからのメッセージ受信を行う CUP のポート番号を指定します。

この定義で指定したポート番号は、次の機能を使用する場合に有効となります。

- 通常 RPC 機能 (受信時)
- トランザクション制御機能
- 常設コネクション確立機能

この定義を省略すると、システムが任意に割り当てたポート番号を使用します。

同一マシン内で、複数のプロセス、または複数のスレッドを同時に実行する場合は、それぞれ異なるポート番号を指定してください。

指定できるポート番号でも、OS またはほかのプログラムで使用するポート番号は指定しないでください。指定した場合、応答データを正しく受信できないことがあります。なお、OS が使用するポート番号は、OS ごとに異なります。OS のマニュアルなどを参照してください。

●DCCLTRAPHOST=rap リスナー、または DCCM3 の論理端末

TP1/Server の rap リスナーのホスト名およびポート番号、または DCCM3 の論理端末のホスト名およびポート番号を指定します。

形式は次のとおりです。

ホスト名:ポート番号 [,ホスト名:ポート番号,...]

- ホスト名 ～〈文字列〉
- ポート番号 ～〈符号なし整数〉((5001~65535))

ホスト名として指定できる長さは、63 文字までです。クライアント環境定義 DCCLTOPTION に 00000008 を指定した場合、255 文字までです。なお、このオペランドに指定できる長さは、1023 文字までです。

区切り文字','の後ろ以外は空白文字（スペースまたはタブ）を入れないでください。

dc_clt_connect_s 関数の flags に DCNOFLAGS を指定して発行した場合、この定義が定義されていると TP1/Server の rap リスナー、または DCCM3 の論理端末に対して常設コネクションの確立を要求します。定義されていない場合は、TP1/Server のクライアント拡張サービスに対して常設コネクションの確立を要求します。

DCCM3 の論理端末に対して常設コネクションの確立要求をする場合、データ圧縮機能は使用できません。

また、CUP と rap リスナー、または DCCM3 の論理端末の間にファイアウォールが存在する場合は、DCCLTRAPHOST に指定するホスト名およびポート番号はファイアウォールのホスト名およびポート番号になるので注意してください。

ホスト名として、10 進ドット記法の IP アドレスを指定することもできます。

rap リスナーのホスト名およびポート番号、または DCCM3 の論理端末のホスト名およびポート番号をそれぞれ複数指定した場合、指定したホスト名およびポート番号の中から一つをランダムに選択し、接続を試みます。接続を試みた先のホストとの接続に失敗したとき、この接続先を除いて再びランダムに選択し、接続を試みます。これを繰り返す、すべてのホストとの接続に失敗した場合、dc_clt_connect_s 関数はエラーを返します。

常設コネクションを確立して rap リスナーまたは DCCM3 の論理端末と通信する場合、クライアント環境定義 DCCLTSERVICEGROUPLIST に指定した値は無視されます。

dc_clt_connect_s 関数の flags およびクライアント環境定義と常設コネクション確立要求先との関係は次のとおりです。

引数 flags	クライアント環境定義		常設コネクション確立要求先
	DCCLTDCCMHOST	DCCLTRAPHOST	
DCNOFLAGS	○	○	rap サーバ、または DCCM3 の論理端末※1
		—	CUP 実行プロセス
	—	○	rap サーバ、または DCCM3 の論理端末※1
		—	CUP 実行プロセス
DCCLT_DCCM3	○	○	DCCM3 の論理端末※2
		—	DCCM3 の論理端末※2
	—	○	エラーリターン
		—	エラーリターン

(凡例)

○：指定あり

- : 指定なし

注※1

DCCLTRAPHOST に指定された DCCM3 の論理端末に対して常設コネクションの確立を要求します。

注※2

DCCLTDCCMHOST に指定された DCCM3 の論理端末に対して常設コネクションの確立を要求します。

●DCCLTRAPAUTOCONNECT=Y | N ~ 《N》

CUP と rap サーバまたは DCCM3 論理端末との間の常設コネクションを自動的に確立させるかどうかを指定します。

Y : 常設コネクションを自動的に確立させます。

N : 常設コネクションを自動的に確立させません。

この定義に Y を指定した場合、次の関数実行時に、常設コネクション未確立の場合は、自動的に常設コネクションが確立されます。常設コネクション確立要求先は、クライアント環境定義 DCCLTRAPHOST に指定したの rap リスナーまたは DCCM3 論理端末です。

1. dc_rpc_call_s 関数
2. dc_trn_begin_s 関数

ただし、DCCM3 論理端末に対して 2.の関数を実行した場合、エラーが返ります。

この定義に Y を指定した場合、dc_clt_connect_s 関数を実行する必要はありません。また、dc_rpc_close_s 関数実行時、自動的に常設コネクションが解放されるので、dc_clt_disconnect_s 関数も実行する必要はありません。

●DCCLTTRSTATISITEM=統計情報項目〔統計情報項目〕…

トランザクションブランチの統計情報を取得する項目を、次の文字列で指定します。CUP からトランザクションを開始する場合だけ有効です。

nothing

統計情報を取得しません。

base

基本情報として、次の情報を取得します。

- トランザクションブランチの識別子
- トランザクションブランチの決着結果
- トランザクションブランチの実行プロセス種別
- トランザクションブランチの実行サーバ名
- トランザクションブランチの実行サービス名

executiontime

基本情報とトランザクションブランチの実行時間情報を取得します。

cputime

基本情報とトランザクションブランチの CPU 時間情報を取得します。

nothing の指定は、一つしかできません。また、nothing とほかの統計情報項目を同時に指定した場合、nothing の指定は無効になります。

トランザクションに関する統計情報を取得する場合は、次のどちらかを指定してください。

- トランザクションサービス定義で trn_tran_statistics=Y と指定
- trnstics コマンドで-s オプションを指定

この定義を省略した場合は、クライアントサービス定義の trn_statistics_item オペランドの指定に従います。また、接続先が rap サーバの場合、rap リスナーサービス定義の trn_statistics_item オペランドの指定に従います。

●DCCLTTROPTIITEM=トランザクション最適化項目〔トランザクション最適化項目〕…

複数のユーザサーバで構成されるグローバルトランザクションの性能を向上させるための最適化項目を、次の文字列で指定します。CUP からトランザクションを開始する場合だけ有効です。

base

同期点取得処理全体（プリペア処理、コミット処理、およびロールバック処理）を最適化します。OpenTP1 のトランザクション制御は 2 相コミット方式で実行しているため、二つのトランザクションブランチ間のコミット制御には、4 回のプロセス間通信が必要となります。

次の条件をすべて満たす場合、親トランザクションブランチが子トランザクションブランチのコミット処理を代わりに実行することで、コミット制御に必要な 4 回のプロセス間通信を削減します。

- 親トランザクションブランチと、子トランザクションブランチが同じ OpenTP1 下にあること。
- 親トランザクションブランチが、子トランザクションブランチを同期応答型 RPC で呼び出していること。
- 子トランザクションブランチでアクセスしたリソースマネージャの XA インタフェース用オブジェクトが、親トランザクションブランチにもリンクされていること。

asyncprepare

base の指定条件を満たしていないため同期点取得処理全体の最適化ができない場合に、プリペア処理を最適化します。

次の条件をすべて満たす場合、親トランザクションブランチから発行された RPC によって子トランザクションブランチがサービス要求を実行したときに、RPC のリターン前にプリペア処理を実行することで、2 回のプロセス間通信を削減します。

- base を指定した最適化ができないこと。
- 親トランザクションブランチが、子トランザクションブランチを同期応答型 RPC で呼び出していること。

ただし、この最適化を実行した場合、親トランザクションブランチが発行した同期応答型 RPC の応答時間が遅くなります。また、子トランザクションブランチは、プリペア処理からコミット処理までの間隔（親トランザクションブランチからの指示がないとトランザクションを決着できない状態）が大きくなります。そのため、親トランザクションブランチの OpenTP1 がシステムダウンし、トランザクションブランチ間の連絡ができなくなると、ジャーナルファイルのスワップやチェックポイントダンプファイルの有効化が遅れ、子トランザクションブランチの OpenTP1 もシステムダウンする場合があります。

トランザクション最適化項目は、重複して指定できます。ただし、優先順位は次のようになります (1 > 2)。

1 : base

2 : asyncprepare

この指定を省略した場合は、クライアントサービス定義の `trn_optimum_item` オペランドの指定に従います。また、接続先が rap サーバの場合、rap リスナーサービス定義の `trn_optimum_item` オペランドの指定に従います。

●DCCLTTRWATCHTIME=トランザクション同期点処理時の最大通信待ち時間

～ 〈符号なし整数〉 ((1~65535)) (単位：秒)

トランザクションの同期点処理で、トランザクションブランチ間で行う通信（プリペア、コミット、ロールバック指示、応答など）の受信待ち時間の最大値を指定します。CUP からトランザクションを開始する場合だけ有効です。

指定時間を過ぎてても指示または応答がない場合は、該当するトランザクションブランチが 2 相コミットの 1 相目であればロールバックさせ、1 相目完了後であればトランザクションサービスのシステムプロセスでトランザクション決着処理を再試行します。

この定義を省略した場合は、クライアントサービス定義の `trn_watch_time` オペランドの指定に従います。また、接続先が rap サーバの場合、rap リスナーサービス定義の `trn_watch_time` オペランドの指定に従います。

●DCCLTTRRBINFO=no | self | remote | all

トランザクションブランチがロールバックした場合に、ロールバック要因に関する情報をログに取得するかどうかを指定します。CUP からトランザクションを開始する場合だけ有効です。

no

ロールバック情報を取得しません。

self

ロールバック要因が発生したトランザクションブランチでだけ、ログにロールバック情報を取得します。

remote

self に加え、他ノードのトランザクションブランチからロールバック要求されたトランザクションブランチでも、ログにロールバック情報を取得します。

all

remote に加え、自ノードのトランザクションブランチからロールバック要求されたトランザクションブランチでも、ログにロールバック情報を取得します。

この指定を省略した場合は、クライアントサービス定義の `trn_rollback_information_put` オペランドの指定に従います。また、接続先が rap サーバの場合、rap リスナーサービス定義の `trn_rollback_information_put` オペランドの指定に従います。

●DCCLTTRLIMITTIME=トランザクションブランチ最大実行可能時間

～ 〈符号なし整数〉 ((0~65535)) (単位：秒)

トランザクションブランチの最大実行可能時間を指定します。CUP からトランザクションを開始する場合だけ有効です。

トランザクションブランチを開始してから同期点処理が終了するまでの時間がこのオペランドの指定時間を超えないように `dc_rpc_call_s` 関数および同期点処理内で行う通信のタイムアウト時間を次のように自動設定します。

- `dc_rpc_call_s` 関数のタイムアウト時間

「 $K \geq$ このオペランドの指定時間」の場合は、要求処理を実行しないで、タイムアウトでエラーリターンします。

「 $K <$ このオペランドの指定時間」でかつ「 $(\text{このオペランドの指定時間} - K) \geq W$ 」の場合は、 W をタイムアウト時間とします。

「 $K <$ このオペランドの指定時間」でかつ「 $(\text{このオペランドの指定時間} - K) < W$ 」の場合は、(このオペランドの指定時間 - K) をタイムアウト時間とします。

K : 現時刻 - トランザクションブランチ開始時刻

W : `DCWATCHTIM` オペランド指定時間

- 同期点処理内で行う通信のタイムアウト時間

「 $K \geq$ このオペランドの指定時間」の場合は、タイムアウト時間を 1 秒とします。

「 $K <$ このオペランドの指定時間」でかつ「 $(\text{このオペランドの指定時間} - K) \geq W$ 」の場合は、 W をタイムアウト時間とします。

「 $K <$ このオペランドの指定時間」でかつ「 $(\text{このオペランドの指定時間} - K) < W$ 」の場合は、(このオペランドの指定時間 - K) をタイムアウト時間とします。

K : 現時刻 - トランザクションブランチ開始時刻

W : `DCCLTTRWATCHTIME` オペランド指定時間 (`DCCLTTRWATCHTIME` オペランドを省略した場合は `DCWATCHTIM` オペランド指定時間)

上記の受信待ち以外の処理で時間が掛かった場合は、このオペランドの指定時間以内にトランザクションブランチが終了しないことがあります。

同期点処理開始前にこのオペランドの指定時間が経過した場合、そのトランザクションはロールバックされます。

0 を指定した場合は、時間監視を行いません。

この定義を省略した場合は、クライアントサービス定義の `trn_limit_time` オペランドの指定に従います。また、接続先が rap サーバの場合、rap リスナーサービス定義の `trn_limit_time` オペランドの指定に従います。

●DCCLTTRBRCV=Y | N

RPC 先トランザクションブランチにロールバック指示を送信したあと、ロールバック完了通知を受信するかどうかを指定します。CUP からトランザクションを開始する場合だけ有効です。

Y : ロールバック完了通知を受信します。

N : ロールバック完了通知を受信しません。

N を指定した場合、RPC 先トランザクションブランチからのロールバック完了通知を受信しないで (RPC 先トランザクションブランチのロールバック処理の完了を待たずに) 自トランザクションブランチを終了します。

この定義を省略した場合は、クライアントサービス定義の `trn_rollback_response_receive` オペランドの指定に従います。また、接続先が rap サーバの場合、rap リスナーサービス定義の `trn_rollback_response_receive` オペランドの指定に従います。

●DCCLTTRRECOVERYTYPE=type1 | type2 | type3

UAP 障害時のトランザクション同期点処理方式を指定します。CUP からトランザクションを開始する場合だけ有効です。

RPC がタイムアウトし、RPC 発行先プロセスのアドレスが未解決の場合やトランザクション実行中の UAP がダウンした場合に、トランザクションブランチ間の連絡がスムーズに行えず、トランザクションの決着に時間が掛かることがあります。

次の障害が発生した場合のトランザクション同期点処理方式を三つの方式から選択します。

type1

障害 1：RPC がタイムアウトした場合

この場合、RPC 発行元トランザクションブランチは、サービス要求がどのプロセスで実行されているかがわからないため、RPC 発行先トランザクションブランチにトランザクション同期点メッセージを送信できず、RPC 発行元トランザクションブランチ、RPC 発行先トランザクションブランチ 共にトランザクション同期点メッセージ待ちとなり、トランザクションの決着に時間が掛かります。

type2

障害 2：RPC 発行元 UAP が RPC の応答受信前にダウンした場合

この場合、RPC 発行元トランザクションブランチは、サービス要求がどのプロセスで実行されているかがわからないため、RPC 発行先トランザクションブランチにトランザクション同期点メッセージを送信できず、RPC 発行先トランザクションブランチはトランザクション同期点メッセージ待ちとなり、トランザクションの決着に時間が掛かります。

type3

障害 3：RPC 発行先 UAP からの応答受信後に RPC 発行元 UAP と RPC 発行先 UAP がほぼ同時にダウンした場合

この場合、それぞれのトランザクションブランチを引き継いだトランザクション回復プロセスは、相手 UAP プロセスのダウンを知らないため、すでに存在しない UAP プロセスにトランザクション同期点メッセージを送信してしまい、トランザクションの決着に時間が掛かることがあります。

次の場合、このオペランドに type2 または type3 を指定しても、トランザクションの決着に時間が掛かることがあります。

- RPC 実行中に、RPC 発行先 UAP の状態が変更となり (負荷増加、UAP 終了および閉塞など)、ほかのノードの同じ UAP にサービス要求が再転送された場合
- 相手先の OpenTP1 がこのオプションをサポートしていないバージョンの場合
- 相手先トランザクションブランチがトランザクション同期点メッセージ受信処理以外で時間が掛かっている場合

この定義を省略した場合は、クライアントサービス定義の `trn_partial_recovery_type` オペランドの指定に従います。また、接続先が rap サーバの場合、rap リスナーサービス定義の `trn_partial_recovery_type` オペランドの指定に従います。

●DCWATCHTIMRPCINHERIT=Y | N ~ 《N》

CUP の最大応答待ち時間を、サーバ側に引き継ぐかどうかを指定します。CUP の最大応答待ち時間を引き継ぐと、CUP がタイムアウトしているのに TP1/Server 側でサービスを実行することを防止できません。

Y：サーバ側に CUP の最大応答待ち時間を引き継ぎます。

N：サーバ側に CUP の最大応答待ち時間を引き継ぎません。

●DCSYSWATCHTIM=OpenTP1 制御の最大応答待ち時間

~ 〈符号なし整数〉 ((0~65535)) 《最大応答待ち時間》 (単位：秒)

OpenTP1 を制御する場合に、要求を送ってから応答が返るまでの待ち時間の最大値を指定します。指定時間を過ぎても応答が返らない場合は、CUP へエラーリターンします。

0 を指定した場合は、応答が返るまで無限に待ち続けます。この定義を省略した場合は、クライアント環境定義 DCWATCHTIM の指定に従います。

なお、クライアント環境定義 DCCLTOPTION に 00000002 を指定し、かつ次のどちらかに該当する場合、この定義は無効となります。

- `dc_rpc_call_s` 関数、または `dc_rpc_call_to_s` 関数実行時。
- クライアント環境定義 DCCLTRAPAUTOCONNECT=Y、かつ `dc_trn_begin_s` 関数実行時。

また、OpenTP1 制御の最大応答待ち時間は、動的に変更できません。

●DCCLTAUTHENT=Y | N ~ 《Y》

`dc_clt_cltin_s` 関数実行時のユーザ認証を行うかどうかを指定します。

Y：ユーザ認証を行います。

N：ユーザ認証を抑止します。

●DCCLTCONNECTINF=端末識別情報

端末識別情報を指定します。16 進数で指定する場合は、64 バイト以内（先頭に 0x を付け、128 文字まで（先頭 0x を含まない））で指定します。文字列で指定する場合は、64 文字以内（NULL 文字を含まない）で指定します。

常設コネクションを使用して DCCM3 論理端末と通信する場合は、端末識別情報として DCCM3 論理端末の論理端末名称を EBCDIK コードで指定します。ただし、DCCM3 側では先頭 8 バイト目までに指定した値だけが有効になります（9 バイト目以降に指定された値は無視されます）。

この定義に指定した端末識別情報は、`dc_rpc_open_s` 関数実行時に参照され、DCCM3 論理端末に通知されます。

この定義を省略した場合は、DCCM3 論理端末に端末識別情報は通知しません。

ただし、`dc_clt_set_connect_inf_s` 関数を実行すれば、この関数に指定した端末識別情報が、`dc_clt_connect_s` 関数実行時、DCCM3 論理端末に通知されます。

この定義は、クライアント環境定義 DCCLTRAPHOST に DCCM3 論理端末のホスト名およびポート番号を指定して、dc_clt_connect_s 関数（引数 flags に DCNOFLAGS を指定）を実行した場合に有効となります。

●DCSCDMULTI=Y | N ~ 《N》

マルチスケジューラ機能を使用するかどうかを指定します。

Y：マルチスケジューラ機能を使用します。

N：マルチスケジューラ機能を使用しません。

マルチスケジューラ機能を使用すると、複数起動されたマルチスケジューラデーモンの中から、一つをランダムに選択することで、スケジューリングの負荷を軽減できます。

この定義に Y を指定した場合、クライアント環境定義 DCSCDDIRECT, DCSCDPORT, DCSCDMULTICOUNT もあわせて参照してください。また、この定義は、dc_rpc_call_to_s 関数実行時は無効です。

●DCSCDMULTICOUNT=マルチスケジューラデーモンのプロセス数

~ 〈符号なし整数〉 ((1~4096)) 《1》

スケジューラデーモンのプロセス数を指定します。指定するプロセス数は、クライアント環境定義 DCSCDPORT に指定するポート番号の内容によって、次のとおり異なります。

DCSCDPORT 指定値	DCSCDMULTICOUNT 指定値	
マルチスケジューラデーモンのベースとなるポート番号	マルチスケジューラデーモンのプロセス数と同値か、それ以下の値を指定する。	
マルチスケジューラデーモンの任意のポート番号	「マルチスケジューラデーモンのポート番号の最大 - DCSCDPORT に指定したポート番号 + 1」の値か、それ以下の値を指定する。	
マスタスケジューラデーモンのポート番号	マスタスケジューラデーモンとマルチスケジューラデーモンのベースとなるポート番号が連続している場合	「マルチスケジューラデーモンのプロセス数 + 1」の値か、それ以下の値を指定する。
	上記以外	1 を指定するか、または指定を省略する（マルチスケジューラデーモンは使用できない）。

この定義は、クライアント環境定義 DCSCDMULTI=Y および DCSCDDIRECT=Y で、かつ DCSCDPORT を指定した場合に有効です。なお、この場合、ポート番号は、次に示す範囲のポート番号からランダムに選択します。

- 下限値：クライアント環境定義 DCSCDPORT に指定したポート番号の値
- 上限値：下限値 + クライアント環境定義 DCSCDMULTICOUNT に指定したプロセス数 - 1

●DCHOSTSELECT=Y | N ~ 《N》

窓口となる TP1/Server をランダムに選択するかどうかを指定します。この定義は、窓口となる TP1/Server を複数指定した場合だけ有効です。

Y：窓口となる TP1/Server をランダムに選択します。

N：窓口となる TP1/Server をランダムに選択しません。

Y を指定した場合、ユーザ認証時に指定した TP1/Server から、窓口となる TP1/Server をランダムに選択します。

窓口となる TP1/Server のネームサービスへ問い合わせ中に障害を検出した場合、障害を検出した TP1/Server を除いて、再び窓口となる TP1/Server をランダムに選択し、切り替えを試みます。

N を指定した場合、ユーザ認証時に指定した TP1/Server の中から、またはクライアント環境定義 DCHOST に指定した TP1/Server の中から、窓口となる TP1/Server を先頭から順番に選択します。窓口となる TP1/Server のネームサービスへ問い合わせ中に障害を検出した場合、指定された次の TP1/Server に切り替えを試みます。

クライアント環境定義 DCSCDDIRECT に Y を指定している場合、該当するポート番号への送信が失敗した時点で、窓口となる TP1/Server の切り替えが発生します。

●DCSCDLOADPRIORITY=Y | N ~ 《N》

サービス要求を受け付けた窓口となる TP1/Server を優先して負荷分散するかどうかを指定します。

Y：サービス要求を受け付けた窓口となる TP1/Server を優先して負荷分散します。

N：スケジュールサービス定義の scd_this_node_first オペランドの指定に従います。

この定義は、ネームサービスを使用しない RPC を行う場合（クライアント環境定義 DCSCDDIRECT に Y を指定）だけ、有効です。

●DCCLTONLYTHISNODE=Y | N ~ 《N》

dc_rpc_call_to_s 関数発行時、指定したノードでサービスを実行するかどうかを指定します。

Y：指定したノードでサービスを実行します。指定したノード以外でサービスが実行されることはありません。

N：指定したノードを優先してサービスを実行します。サービスの状態によっては、他ノードへ転送されることもあります。

●DCCLTNOSERVER=Y | N ~ 《N》

TP1/Server とは通信することがない環境かどうかを指定します。

Y：TP1/Server とは通信しません。DCCM3 論理端末だけと通信する場合に指定します。

N：TP1/Server と通信します。

Y を指定した場合でも、dc_clt_cltin_s 関数は必ず発行してください。引数 logname には、NULL 以外の任意の値を指定してください。NULL を指定した場合、dc_clt_cltin_s 関数は DCCLTER_INVALID_ARGS でエラーリターンします。引数 passwd には、任意の値を指定してください。NULL を指定しても問題ありません。

●DCHOSTCHANGE=Y | N ~ 《Y》

dc_rpc_call_s 関数実行時、スケジュールサービス開始処理中、およびスケジュールサービス終了処理中に、窓口となる TP1/Server からエラー応答を受信した場合、窓口となる TP1/Server をほかの TP1/Server に切り替えるかどうかを指定します。

Y：窓口となる TP1/Server を切り替えます。

N：窓口となる TP1/Server を切り替えません。

指定に誤りがあった場合、Y が指定されていると仮定されます。

N を指定した場合、dc_rpc_call_s 関数は即時にエラーリターンします。スケジュールサービス開始処理中のときは、DCRPCER_OLTF_INITIALIZING でエラーリターンします。スケジュールサービス終了処理中のときは、DCRPCER_OLTF_NOT_UP でエラーリターンします。

この定義は、dc_clt_cltin_s 関数の引数 target_host、またはクライアント環境定義 DCHOST に複数の窓口となる TP1/Server を指定した場合に有効です。なお、常設コネクション確立中、またはトランザクションの範囲内で dc_rpc_call_s 関数を実行した場合は、この定義は無効となります。

●DCCLTOPTION=クライアントの機能拡張オプション

～〈符号なし 16 進整数〉 ((00000000～00000008)) 《00000000》

クライアントの機能拡張オプションを指定します。複数のオプションを指定する場合、それぞれの指定値の論理和を指定してください。なお、指定できない値を指定した場合でも、エラーとならないで誤動作してしまふことがあります。

00000000

このオプションを指定した場合、機能を拡張しません。

00000002

このオプションを指定した場合、通信系関数の受信処理では、クライアントの最大応答待ち時間（クライアント環境定義 DCSYSWATCHTIM 指定値、または DCWATCHTIM 指定値）を適用しますが、この時間を受信処理が発生するつど適用するのではなく、通信系関数の応答待ち時間として適用します。

なお、次のどちらかに該当する場合は、DCSYSWATCHTIM 指定値は無効となり、DCWATCHTIM 指定値を通信系関数の応答待ち時間として適用します。

- dc_rpc_call_s 関数、または dc_rpc_call_to_s 関数実行時。
- クライアント環境定義 DCCLTRAPAUTOCONNECT=Y、かつ dc_trn_begin_s 関数実行時。

また、応答待ち時間の端数が生じた場合など、処理が間延びする場合がありますので、ご注意ください。

00000008

このオプションを指定した場合、TP1/Client で扱うホスト名長を 63 文字から 255 文字に拡張します。

指定に誤りがあった場合、エラーログに KFCA02401-E メッセージを出力し、次の関数のどれかが DCCLTER_FATAL でエラーリターンします。

- dc_clt_cltin_s 関数
- dc_clt_accept_notification_s 関数
- dc_clt_cancel_notification_s 関数
- dc_clt_open_notification_s 関数

●DCCLTNAMEXTEND=0 | 1 ～《0》

マルチノードサーバで運用している場合に、クライアントで管理するサービス情報の取得数を拡張します。

0：最大 128 個のサービス情報を取得します。

1：最大 512 個のサービス情報を取得します。

サーバが 129 以上のマルチノードサーバ構成で、窓口となる TP1/Server でネームサービス定義に「nam_service_extend=1」を指定している場合は、この定義に 1 を指定してください。

なお、この定義はクライアント環境定義に「DCCLTLOADBALANCE=Y」を指定している場合にだけ有効です。また、クライアント環境定義に「DCSCDDIRECT=Y」を指定している場合は、この定義の指定は無効になります。

●DCCLTBACKLOGCOUNT=コネクション確立要求を格納するキューの数

～ 〈符号なし整数〉 ((0~4096)) 《0》

コネクション確立要求を格納するキューの数を指定します。0 を指定した場合、または指定を省略した場合、キューの数は次のようになります。

- AIX 5L の場合：1024
- Linux の場合：128
- HI-UX/WE2 の場合：20
- HP-UX の場合：20
- Solaris の場合：5
- Windows の場合：5

なお、実際のキューの数は、指定した値より大きくなる場合があります。キューの数の上限値および下限値は、OS によって異なります。OS によってキューの数の上限値および下限値が制限されている場合、指定した値が有効にならないことがあります。

コネクション確立要求を格納するキューについての詳細は、OS のマニュアルまたは TCP/IP のドキュメントを参照してください。

定義の指定に誤りがあった場合は、次の関数のどれかが DCCLTER_FATAL でエラーリターンし、エラーログには KFCA02401-E メッセージを出力します。

- dc_clt_cltin_s 関数
- dc_clt_open_notification_s 関数
- dc_clt_accept_notification_s 関数

●DCTRCPATH=トレースファイル作成ディレクトリ

～ 〈文字列〉

エラーログファイル、および各種トレースファイルを作成するディレクトリ名を完全パス名で指定します。

指定したディレクトリがない場合、または指定に誤りがあった場合は、ファイルは出力されません。

このオペランドの指定を省略すると、カレントディレクトリ（CUP 実行ディレクトリ）が仮定されます。

●DCTRCERR=エラーログのサイズ

～ 〈符号なし整数〉 ((0~1073741824)) 《4096》 (単位：バイト)

エラーログのファイルサイズを指定します。エラーログは、dcerr1.trc および dcerr2.trc というファイル名で出力されます。出力先は、クライアント環境定義 DCTRCPATH に指定したディレクトリ、または CUP を実行したディレクトリの下です。

0 を指定した場合、指定に誤りがあった場合、または出力する情報がない場合、エラーログは出力されません。

なお、1 ギガバイトまで指定できますが、使用するマシンのスペックによってはファイルを開けない場合が考えられますので、ご注意ください。使用する環境に応じて、適切な値を指定してください。

●DCTRCUAP=UAP トレースのファイルサイズ

～ 〈符号なし整数〉 ((4096~1073741824)) 《出力しない》 (単位：バイト)

UAP トレースのファイルサイズを指定します。UAP トレースは、dcuap1.trc および dcuap2.trc というファイル名で出力されます。出力先は、クライアント環境定義 DCTRCPATH に指定したディレクトリ、または CUP を実行したディレクトリの下です。

指定に誤りがあった場合、指定を省略した場合、または出力する情報がない場合、UAP トレースは出力されません。

なお、1 ギガバイトまで指定できますが、UAP トレースを編集出力してファイルに保存する場合、使用するマシンのスペックによってはファイルを開けない場合が考えられますので、ご注意ください。使用する環境に応じて、適切な値を指定してください。

●DCTRCSOC=ソケットトレースのファイルサイズ

～ 〈符号なし整数〉 ((4096~1073741824)) 《出力しない》 (単位：バイト)

ソケットトレースのファイルサイズを指定します。ソケットトレースは、dcsoc1.trc および dcsoc2.trc というファイル名で出力されます。出力先は、クライアント環境定義 DCTRCPATH に指定したディレクトリ、または CUP を実行したディレクトリの下です。

指定に誤りがあった場合、指定を省略した場合、または出力する情報がない場合、ソケットトレースは出力されません。

なお、1 ギガバイトまで指定できますが、ソケットトレースを編集出力してファイルに保存する場合、使用するマシンのスペックによってはファイルを開けない場合が考えられますので、ご注意ください。使用する環境に応じて、適切な値を指定してください。

●DCTRCSOCSIZE=ソケットトレースのデータサイズ

～ 〈符号なし整数〉 ((64~4096)) 《256》 (単位：バイト)

ソケットトレースを出力する場合、ソケットトレースのデータサイズを指定します。

指定に誤りがあった場合、または指定を省略した場合、省略値が有効となります。

●DCTRCMDL=モジュールトレースのファイルサイズ

～ 〈符号なし整数〉 ((4096~1073741824)) 《出力しない》 (単位：バイト)

モジュールトレースのファイルサイズを指定します。モジュールトレースは、dcmdl1.trc および dcmdl2.trc というファイル名で出力されます。出力先は、クライアント環境定義 DCTRCPATH に指定したディレクトリ、または CUP を実行したディレクトリの下です。

指定に誤りがあった場合、指定を省略した場合、または出力する情報がない場合、モジュールトレースは出力されません。

なお、1 ギガバイトまで指定できますが、モジュールトレースを編集出力してファイルに保存する場合、使用するマシンのスペックによってはファイルを開けない場合が考えられますので、ご注意ください。使用する環境に応じて、適切な値を指定してください。

●DCCLTPRFINFOSEND=Y | N ~ 《N》

性能検証用トレースの識別情報を、TP1/Server に伝播するかどうかを指定します。

Y：性能検証用トレースの識別情報を、TP1/Server に伝播します。

N：性能検証用トレースの識別情報を、TP1/Server に伝播しません。

この定義に Y を指定し、かつクライアント環境定義 DCTRCUAP に UAP トレースを取得するように指定した場合、UAP トレースに性能検証用トレース情報を出力します。このとき、UAP トレースを取得し、性能検証用トレースの識別情報を TP1/Server に伝播すると、TP1/Client の関数の実行に掛かった時間と、TP1/Server のサービスの実行に掛かった時間との照合ができます。また、障害が発生したときに、処理がどこまで到達しているかを確認できます。

●DCCLTRPCMAXMSGSIZE=RPC 送受信電文の最大長

~ 〈符号なし整数〉 ((1~8)) 《1》 (単位：メガバイト)

RPC、またはサーバからの一方通知受信機能を使用したときに送受信できるユーザデータの最大長を指定します。この定義は次に示す関数と引数を指定したときに有効になります。

関数	引数	
	入力パラメタ長	出力パラメタ長
dc_rpc_call_s 関数	in_len	out_len
dc_rpc_call_to_s 関数	in_len	out_len
dc_clt_accept_notification_s 関数	—	inf_len
dc_clt_cancel_notification_s 関数	inf_len	—
dc_clt_chained_accept_notification_s 関数	—	inf_len

(凡例)

—：該当しません。

このオペランドに 2 以上を指定した場合、送受信できるユーザデータの最大長は DCRPC_MAX_MESSAGE_SIZE の値ではなく、このオペランドに指定した値になります（「このオペランドで指定した値×1024×1024」バイトになります）。上記の関数の引数に「このオペランドに指定した値×1024×1024」よりも大きい値を指定した場合、DCRPCER_INVALID_ARGS、または DCRPCER_MESSAGE_TOO_BIG でエラーリターンします。

引数 inf_len に指定した値よりも大きいユーザデータを受信した場合、一方通知受信機能の関数の dc_clt_accept_notification_s 関数、または dc_clt_chained_accept_notification_s 関数は DCCLTER_INF_TOO_BIG でエラーリターンします。

このオペランドに 2 以上の値を指定し、DCRPC_MAX_MESSAGE_SIZE の値よりも大きいユーザデータを送受信する場合は、次のことに注意してください。

- サービス要求先となる TP1/Server のシステム共通定義の `rpc_max_message_size` オペランドにも適切な値を指定する必要があります。また、システム共通定義の `all_node` オペランドに指定したすべてのノードで同じ値を指定してください。
- システム共通定義の `rpc_max_message_size` オペランドをサポートしていない TP1/Server にサービス要求した場合の動作は保証できません。次に示すどちらかの RPC を実行した場合、TP1/Server がダウンすることがあります。
 - ・クライアント環境定義 `DCSCDDIRECT` に Y を指定し、`dc_rpc_call_s` 関数を発行。
 - ・クライアント環境定義 `DCCLTONLYTHISNODE` に N を指定し、`dc_rpc_call_to_s` 関数を発行。
- サービス関数内で「システム定義 `rpc_max_message_size` × 1024 × 1024」より大きい応答データ長を指定すると、`dc_rpc_call_s` 関数、または `dc_rpc_call_to_s` 関数は `DCRPCER_INVALID_REPLY` でエラーリターンします。
- サービス要求が TP1/Server のノード間負荷バランス機能によって別のノードへ転送されると、`DCRPCER_NET_DOWN` でエラーリターンすることがあります。
- 常設コネクション確立機能、またはトランザクション制御機能を使用する場合、システム共通定義の `rpc_max_message_size` オペランドに適切な値を指定しないと、`dc_rpc_call_s` 関数は `DCRPCER_MESSAGE_TOO_BIG`、または `DCRPCER_INVALID_ARGS` でエラーリターンします。
- システム共通定義の `rpc_max_message_size` オペランドをサポートしている TP1/Server 上にサービスが起動されていない場合、`dc_rpc_call_s` 関数、または `dc_rpc_call_to_s` 関数は `DCRPCER_NO_SUCH_SERVICE_GROUP`、または `DCRPCER_TRNCHK` でエラーリターンします。

●DCCLTRECVBUFFSIZE=TCP/IP の受信バッファサイズ

～ 〈符号なし整数〉 ((8192～1048576)) (単位：バイト)

コネクションごとに確保される TCP/IP の受信バッファのサイズを指定します。この値を相手システムのバッファ長に合わせて変更することで、通信効率の向上が図れます。

注

TCP は、受信したデータに対し、送達確認 (ACK) パケットを返信します。

受信バッファのサイズに対し、受信したデータが小さいと、データを受信しても直ちに ACK を返信しません (遅延 ACK)。

このオペランドに大きな値を指定し、小さいデータをやり取りし合うような通信処理の場合、遅延 ACK の影響によって性能が悪くなるおそれがあります。遅延 ACK についての詳細は、TCP/IP のドキュメントを参照してください。

なお、このオペランドの値は、OS で使用できる TCP/IP の受信バッファのサイズ以下の値を指定してください。

この定義を省略した場合は、システムによって決められた値を使用します。

●DCCLTSENBUFFSIZE=TCP/IP の送信バッファサイズ

～ 〈符号なし整数〉 ((8192～1048576)) (単位：バイト)

コネクションごとに確保される TCP/IP の送信バッファのサイズを指定します。この値を相手システムのバッファ長に合わせて変更することで、通信効率の向上が図れます。

注

このオペランドの値は、OS で使用できる TCP/IP の送信バッファのサイズ以下の値を指定してください。

この定義を省略した場合は、システムによって決められた値を使用します。

●DCCLTTCPNODELAY=Y | N ~ 《N》

Nagle アルゴリズムを無効にするかどうかを指定します。

Y : Nagle アルゴリズムを無効にします。

N : Nagle アルゴリズムを無効にしません。

この定義に Y を指定した場合、INET ドメイン通信時の送信効率が低下し、ネットワークの負荷が大きくなる場合があります。この定義に Y を指定する場合は、クライアント環境定義 DCCLTSENDBUFSIZE、クライアント環境定義 DCCLTRECVBUFFER、ネットワークの帯域など、この機能の必要性を十分に検討してください。

なお、Nagle アルゴリズムについての詳細は、TCP/IP のドキュメントを参照してください。

8.2.6 TP1/Client/P 固有のオペランド

●dcselint=受信チェック間隔時間

~ 〈符号なし整数〉 ((0~65535)) 《100》 (単位：ミリ秒)

サーバの応答を監視するために、サーバの応答があったかどうかチェックする間隔時間を指定します。

0 を指定した場合、応答性能は良くなりますが、サーバの応答を受信するまで応答待ちスレッドのほかのウィンドウが動作できなくなります。

なお、この定義には、クライアント環境定義 DCSYSWATCHTIM および DCWATCHTIM よりも小さい値を指定してください。

8.2.7 TP1/Client/W の注意事項

- CUP を実行中にクライアント環境定義を変更しないでください。
- TP1/Client/W では、dc_clt_cltin_s 関数の呼び出しごとにクライアント環境定義を定義できます。dc_clt_cltin_s 関数の呼び出しごとにクライアント環境定義を定義するには、dc_clt_cltin_s 関数の呼び出しごとに、異なるファイルをクライアント環境定義ファイルとして作成して、そのファイル名を dc_clt_cltin_s 関数の引数 defpath に指定してください。また、次に示す関数でも同様に指定できます。
 - dc_clt_accept_notification_s 関数
 - dc_clt_cancel_notification_s 関数
 - dc_clt_open_notification_s 関数

8.2.8 TP1/Client/P の注意事項

- CUP を実行中にクライアント環境定義を変更しないでください。
- TP1/Client/P では dc_clt_cltin_s 関数の呼び出しごとにクライアント環境定義を定義できます。dc_clt_cltin_s 関数の呼び出しごとにクライアント環境定義を定義するには、dc_clt_cltin_s 関数の呼び出しごとに、異なるファイルをクライアント環境定義ファイルとして作成して、そのファイル名を dc_clt_cltin_s 関数の引数 defpath に指定してください。また、次に示す関数でも同様に指定できます。
 - dc_clt_accept_notification_s 関数
 - dc_clt_cancel_notification_s 関数
 - dc_clt_open_notification_s 関数

9

運用コマンド

使用できる運用コマンドの記述形式，および運用コマンドの内容を説明します。

この章では，各関数の DLL を呼び出すときの C 言語の関数名（dc_xxx_xxx_s）で説明します。通常オブジェクトライブラリの関数および COBOL 言語を使う場合は，各関数に対応する通常オブジェクトライブラリの関数名（dc_xxx_xxx）および COBOL 言語の要求文に置き換えて読んでください。

9.1 運用コマンドの記述形式

運用コマンドの記述形式を次に示します。

コマンド名△オプション

コマンド名

実行するコマンドの名称です。

オプション

TP1/Client/W の場合

マイナス記号 (-) で始まる文字列で、フラグ引数をとらないか、または一個のフラグ引数をとります。

オプションの記述形式を次に示します。

-オプションフラグ

または

-オプションフラグ△フラグ引数

(凡例)

オプションフラグ：1 文字の英数字

フラグ引数：オプションフラグに対する引数

TP1/Client/P の場合

スラント (/) で始まる文字列で、フラグ引数をとらないか、または 1 個のフラグ引数をとります。

オプションの記述形式を次に示します。

/オプションフラグ

または

/オプションフラグ△フラグ引数

(凡例)

オプションフラグ：1 文字の英数字

フラグ引数：オプションフラグに対する引数

9.2 運用コマンドの詳細

TP1/Client で使用する運用コマンドを次に説明します。

9.2.1 cltdump (トレースの編集出力)

(1) 形式

TP1/Client/P を使用する場合、cltdump コマンドは MS-DOS プロンプトから入力してください。

(a) TP1/Client/W の場合

```
cltdump [-u|-s|-m] [-n] [-f ファイル名]
```

(b) TP1/Client/P の場合

```
cltdmp32 [/u|/s|/m] [/n] [/f ファイル名]
```

(c) TP1/Client/P(64)の場合

```
cltdmp64 [/u|/s|/m] [/n] [/f ファイル名]
```

(2) 機能

TP1/Client が提供しているトレースを編集し、標準出力に出力します。ファイルに保存する場合は、任意のファイルに標準出力をリダイレクトしてください。このコマンドを使用すると、次のトレースを編集できます。

- UAP トレース (dcuap1.trc および dcuap2.trc)
- ソケットトレース (dcsoc1.trc および dcsoc2.trc)
- モジュールトレース (dcmdl1.trc および dcmdl2.trc)

各トレースには、二つのトレースファイルがあります。cltdump コマンドを実行すると、ファイルを取得時刻の順に選択して、古いものから出力します。

なお、ソケットトレースおよびモジュールトレースの出力内容は、公開しておりませんが、保守員が障害調査資料として使用場合がありますので、保守員の指示に従ってください。

このコマンドは、トレースを出力した TP1/Client のバージョンと同じバージョンのコマンドの使用をお勧めします。異なるバージョンのコマンドを使用すると、トレースの出力内容が正しく編集されない場合があります。

(3) オプション

- -u, または/u
UAP トレース (dcuap1.trc および dcuap2.trc) を編集出力します。
- -s, または/s
ソケットトレース (dcsoc1.trc および dcsoc2.trc) を編集出力します。
ソケットトレースの出力内容については、公開していません。
- -m, または/m
モジュールトレース (dcmdl1.trc および dcmdl2.trc) を編集出力します。
モジュールトレースの出力内容については、公開していません。
- -n, または/n
UAP トレースを編集出力する場合、ユーザが発行した関数を「関数名」で出力します。このオプションを省略した場合、ユーザが発行した関数を「関数コード」で出力します。
UAP トレース以外を編集出力する場合にこのオプションを指定した場合、このオプションの指定は無視されます。
- -f ファイル名または/f ファイル名
特定のトレースファイルを標準出力に編集出力する場合、完全パス名、またはファイル名を指定します。ファイル名だけを指定すると、カレントディレクトリ (CUP 実行ディレクトリ) 上のファイルとみなされます。
このオプションを指定すると、指定したファイルだけが編集出力され、二つのファイル (UAP トレースの場合、dcuap1.trc と dcuap2.trc) はマージされません。
このオプションの指定を省略すると、クライアント環境定義 DCTRCPATH (環境変数で定義) に指定したディレクトリに出力された二つのファイル (UAP トレースの場合、dcuap1.trc と dcuap2.trc) がマージされ、編集出力します。
また、オプションの指定とクライアント環境定義 DCTRCPATH (環境変数で定義) の指定の両方を省略した場合、カレントディレクトリ (CUP 実行ディレクトリ) 上の二つのファイルがマージされ、編集出力します。

(4) 出力例

ソケットトレースおよびモジュールトレースの出力内容は公開していません。ここでは UAP トレースの出力例を示します。

DATE = aaaaaaaaa	TIME = bbbbbbbbbb	PID = ccccc	SIZE = ddd
CODE1 = eeeeeeee	CODE2 = f	RETURN = ggggg	
ADDRESS	+0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +a +b +c +d +e +f	0123456789abcdef	
00000000	73 61 69 73 70 70 00 00 00 00 00 00 00 00 00	saispp.....] 1
00000010	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000020	73 61 69 73 76 72 30 31 00 00 00 00 00 00 00	saisvr01.....	
:	:	:	
└──────────┘		└──────────┘	
2	3	4	

aaaaaaaaaa

UAP トレース取得日

yyyy/mm/dd (yyyy・・・年, mm・・・月, dd・・・日) の形式

bbbbbbbbbbb

UAP トレース取得時刻

hh:mm:ss.sss (hh・・・時, mm・・・分, ss・・・秒, sss・・・ミリ秒) の形式

cccc

UAP トレースを取得したプロセスのプロセス ID

TP1/Client/W の場合は, プロセス ID:クライアント ID という形式で出力される。

TP1/Client/P の場合は, プロセス ID:スレッド ID という形式で出力される。

ddd

データサイズ

eeeeeee

呼び出された関数の種別コード 1

180000 (16 進数)・・・TP1/Client からの要求であることを示す。

f

呼び出された関数の種別コード 2

1・・・dc_rpc_open_s 関数であることを示す。

2・・・dc_rpc_close_s 関数であることを示す。

3・・・dc_rpc_call_s 関数であることを示す。

4・・・dc_clt_cltin_s 関数であることを示す。

5・・・dc_clt_cltout_s 関数であることを示す。

6・・・dc_clt_send_s 関数であることを示す。

7・・・dc_clt_receive_s 関数であることを示す。

8・・・dc_trn_begin_s 関数であることを示す。

9・・・dc_trn_chained_commit_s 関数であることを示す。

- 1a・・・dc_clt_set_raphost_s 関数であることを示す。
- 1b・・・dc_clt_get_raphost_s 関数であることを示す。
- 1c・・・dc_clt_assem_send_s 関数であることを示す。
- 1d・・・dc_clt_assem_receive_s 関数であることを示す。
- a・・・dc_trn_chained_rollback_s 関数であることを示す。
- b・・・dc_trn_unchained_commit_s 関数であることを示す。
- c・・・dc_trn_unchained_rollback_s 関数であることを示す。
- d・・・dc_trn_info_s 関数であることを示す。
- e・・・dc_clt_get_trnid_s 関数であることを示す。
- f・・・dc_rpc_get_watch_time_s 関数であることを示す。
- 10・・・dc_rpc_set_watch_time_s 関数であることを示す。
- 13・・・dc_clt_connect_s 関数であることを示す。
- 14・・・dc_clt_disconnect_s 関数であることを示す。
- 17・・・dc_clt_receive2_s 関数であることを示す。
- 18・・・dc_clt_set_connect_inf_s 関数であることを示す。
- 19・・・dc_rpc_call_to_s 関数であることを示す。
- 100・・・dc_clt_accept_notification_s 関数であることを示す。
- 101・・・dc_clt_cancel_notification_s 関数であることを示す。
- 102・・・dc_clt_open_notification_s 関数であることを示す。
- 103・・・dc_clt_close_notification_s 関数であることを示す。
- 104・・・dc_clt_chained_accept_notification_s 関数であることを示す。
- 200・・・tpalloc 関数であることを示す。
- 201・・・tpfree 関数であることを示す。
- 202・・・tpconnect 関数であることを示す。
- 203・・・tpdiscon 関数であることを示す。
- 204・・・tpsend 関数であることを示す。
- 205・・・tprecv 関数であることを示す。

ggggg

呼び出された関数のリターンコード (10 進数)

リターンコードの表す実行結果については、「4. TP1/Client で使用できる関数 (C 言語編)」または「6. TP1/Client で使用できる要求文 (COBOL 言語編)」を参照してください。

1. OpenTP1 関数の呼び出し情報※
2. OpenTP1 関数の呼び出し情報のロケーション
3. OpenTP1 関数の呼び出し情報の 16 進表示のデータ部
4. OpenTP1 関数の呼び出し情報の ASCII 文字表示部

注※

各関数の呼び出し情報の形式を、「(5) 出力形式」および「(6) 出力形式 (TP1/Client/P(64)およびTP1/Client/W(64)の場合)」に示します。

(5) 出力形式

表 9-1 dc_rpc_open_s 関数の呼び出し情報 (関数コード : 1)

種類	長さ (バイト)	領域名	位置 (16 進)	内容	
関数入り口 情報	8	4	cltid	0	dc_rpc_open_s 関数の引数 cltid 指定値 dc_rpc_open 関数の場合、TP1/Client が使用する領域
		4	flags	4	dc_rpc_open_s 関数の引数 flags 指定値

注

関数出口情報はありませぬ。

表 9-2 dc_rpc_close_s 関数の呼び出し情報 (関数コード : 2)

種類	長さ (バイト)	領域名	位置 (16 進)	内容	
関数入り口 情報	8	4	cltid	0	dc_rpc_close_s 関数の引数 cltid 指定値 dc_rpc_close 関数の場合、TP1/Client が使用する領域
		4	flags	4	dc_rpc_close_s 関数の引数 flags 指定値

注

関数出口情報はありませぬ。

表 9-3 dc_rpc_call_s 関数の呼び出し情報 (関数コード : 3)

種類	長さ (バイト)	領域名	位置 (16 進)	内容	
関数入り口 情報	140	4	cltid	0	dc_rpc_call_s 関数の引数 cltid 指定値 dc_rpc_call 関数の場合、TP1/Client が使用する領域
		32	group	4	dc_rpc_call_s 関数の引数 group 指定値
		32	service	24	dc_rpc_call_s 関数の引数 service 指定値
		60	in	44	dc_rpc_call_s 関数の引数 in 指定値
		4	in_len	80	dc_rpc_call_s 関数の引数 in_len 指定値
		4	out_len	84	dc_rpc_call_s 関数の引数 out_len 指定値
		4	flags	88	dc_rpc_call_s 関数の引数 flags 指定値
関数出口 情報	64	60	out	0	dc_rpc_call_s 関数の引数 out に返された値
		4	out_len	3C	dc_rpc_call_s 関数の引数 out_len に返された値

表 9-4 dc_clt_cltin_s 関数の呼び出し情報 (関数コード : 4)

種類	長さ (バイト)	領域名	位置 (16 進)	内容	
関数入り口 情報	168	4	hWnd	0	dc_clt_cltin_s 関数の引数 hWnd の指定値 dc_clt_cltin 関数の場合、0 クリア
		64	defpath	4	dc_clt_cltin_s 関数の引数 defpath の指定値 dc_clt_cltin 関数の場合、0 クリア
		64	target_host	44	dc_clt_cltin_s 関数の引数 target_host の指定値
		32	logname	84	dc_clt_cltin_s 関数の引数 logname の指定値
		4	flags	A4	dc_clt_cltin_s 関数の引数 flags の指定値
関数出口 情報	68	4	cltid	0	dc_clt_cltin_s 関数の引数 cltid に返された値 dc_clt_cltin 関数の場合、TP1/Client が使用する領域
		64	set_host	4	dc_clt_cltin_s 関数の引数 set_host に返された値

表 9-5 dc_clt_cltout_s 関数の呼び出し情報 (関数コード : 5)

種類	長さ (バイト)	領域名	位置 (16 進)	内容	
関数入り口 情報	8	4	cltid	0	dc_clt_cltout_s 関数の引数 cltid 指定値 dc_clt_cltout 関数の場合、TP1/Client が使用する領域
		4	flags	4	dc_clt_cltout_s 関数の引数 flags 指定値

注

関数出口情報はありません。

表 9-6 dc_clt_send_s 関数の呼び出し情報 (関数コード : 6)

種類	長さ (バイト)	領域名	位置 (16 進)	内容	
関数入り口 情報	144	4	cltid	0	dc_clt_send_s 関数の引数 cltid 指定値 dc_clt_send 関数の場合、TP1/Client が使用する領域
		64	buff	4	dc_clt_send_s 関数の引数 buff 指定値
		4	sendleng	44	dc_clt_send_s 関数の引数 sendleng 指定値
		64	hostname	48	dc_clt_send_s 関数の引数 hostname 指定値
		2	portnum	88	dc_clt_send_s 関数の引数 portnum 指定値
		2	—	8A	未使用の領域
		4	flags	8C	dc_clt_send_s 関数の引数 flags 指定値

(凡例)

－：該当しません。

注

関数出口情報はありません。

表 9-7 dc_clt_receive_s 関数の呼び出し情報 (関数コード：7)

種類	長さ (バイト)	領域名	位置 (16 進)	内容	
関数入り口 情報	16	4	cltid	0	dc_clt_receive_s 関数の引数 cltid 指定値 dc_clt_receive 関数の場合、TP1/Client が使用する領域
		4	recvleng	4	dc_clt_receive_s 関数の引数 recvleng 指定値
		4	timeout	8	dc_clt_receive_s 関数の引数 timeout 指定値
		4	flags	C	dc_clt_receive_s 関数の引数 flags 指定値
関数出口 情報	64	64	buff	0	dc_clt_receive_s 関数の引数 buff に返された値

表 9-8 dc_trn_begin_s 関数の呼び出し情報 (関数コード：8)

種類	長さ (バイト)	領域名	位置 (16 進)	内容	
関数入り口 情報	4	4	cltid	0	dc_trn_begin_s 関数の引数 cltid 指定値 dc_trn_begin 関数の場合、TP1/Client が使用する領域
関数出口 情報	32	16	trngid	0	発生したグローバルトランザクションの、トランザクショング ローバル識別子
		16	trnbid	10	発生したグローバルトランザクションの、トランザクションプ ランチ識別子

表 9-9 dc_trn_chained_commit_s 関数の呼び出し情報 (関数コード：9)

種類	長さ (バイト)	領域名	位置 (16 進)	内容	
関数入り口 情報	36	4	cltid	0	dc_trn_chained_commit_s 関数の引数 cltid 指定値 dc_trn_chained_commit 関数の場合、TP1/Client が使用する 領域
		16	trngid	4	現在のグローバルトランザクションの、トランザクショング ローバル識別子
		16	trnbid	14	現在のグローバルトランザクションの、トランザクションプ ランチ識別子
関数出口 情報	32	16	trngid	0	新しく発生したグローバルトランザクションの、トランザクシ ョングローバル識別子

種類	長さ (バイト)	領域名	位置 (16 進)	内容	
関数出口 情報	32	16	trnbid	10	新しく発生したグローバルトランザクションの、トランザクションブランチ識別子

表 9-10 dc_clt_set_raphost_s 関数の呼び出し情報 (関数コード : 1a)

種類	長さ (バイト)	領域名	位置 (16 進)	内容	
関数入り口 情報	136	4	cltid	0	dc_clt_set_raphost_s 関数の引数 cltid 指定値 dc_clt_set_raphost 関数の場合、TP1/Client が使用する領域
		128	raphost	4	dc_clt_set_raphost_s 関数の引数 raphost 指定値
		4	flags	84	dc_clt_set_raphost_s 関数の引数 flags 指定値

注

関数出口情報はありませぬ。

表 9-11 dc_clt_get_raphost_s 関数の呼び出し情報 (関数コード : 1b)

種類	長さ (バイト)	領域名	位置 (16 進)	内容	
関数入り口 情報	8	4	cltid	0	dc_clt_get_raphost_s 関数の引数 cltid 指定値 dc_clt_get_raphost 関数の場合、TP1/Client が使用する領域
		4	flags	4	dc_clt_get_raphost_s 関数の引数 flags 指定値
関数出口 情報	128	128	raphost	0	dc_clt_get_raphost_s 関数の引数 raphost に返された値

表 9-12 dc_clt_assem_send_s 関数の呼び出し情報 (関数コード : 1c)

種類	長さ (バイト)	領域名	位置 (16 進)	内容	
関数入り口 情報	148	4	cltid	0	dc_clt_assem_send_s 関数の引数 cltid 指定値 dc_clt_assem_send 関数の場合、TP1/Client が使用する領域
		64	buff	4	dc_clt_assem_send_s 関数の引数 buff 指定値
		4	sendleng	44	dc_clt_assem_send_s 関数の引数 sendleng 指定値
		64	hostname	48	dc_clt_assem_send_s 関数の引数 hostname 指定値
		2	portnum	88	dc_clt_assem_send_s 関数の引数 portnum 指定値
		2	—	8A	未使用の領域
		4	timeout	8C	dc_clt_assem_send_s 関数の引数 timeout 指定値
		4	flags	90	dc_clt_assem_send_s 関数の引数 flags 指定値

(凡例)

－：該当しません。

注

関数出口情報はありません。

表 9-13 dc_clt_assem_receive_s 関数の呼び出し情報 (関数コード : 1d)

種類	長さ (バイト)	領域名	位置 (16 進)	内容	
関数入り口 情報	16	4	cltid	0	dc_clt_assem_receive_s 関数の引数 cltid 指定値 dc_clt_assem_receive 関数の場合、TP1/Client が使用する領域
		4	recvleng	4	dc_clt_assem_receive_s 関数の引数 recvleng 指定値
		4	timeout	8	dc_clt_assem_receive_s 関数の引数 timeout 指定値
		4	flags	C	dc_clt_assem_receive_s 関数の引数 flags 指定値
関数出口 情報	68	64	buff	0	dc_clt_assem_receive_s 関数の引数 buff に返された値
		4	recvleng	40	dc_clt_assem_receive_s 関数の引数 recvleng に返された値

表 9-14 dc_trn_chained_rollback_s 関数の呼び出し情報 (関数コード : a)

種類	長さ (バイト)	領域名	位置 (16 進)	内容	
関数入り口 情報	36	4	cltid	0	dc_trn_chained_rollback_s 関数の引数 cltid 指定値 dc_trn_chained_rollback 関数の場合、TP1/Client が使用する 領域
		16	trngid	4	現在のグローバルトランザクションの、トランザクショングロー バル識別子
		16	trnbid	14	現在のグローバルトランザクションの、トランザクションブラン チ識別子
関数出口 情報	32	16	trngid	0	新しく発生したグローバルトランザクションの、トランザクシ ョングローバル識別子
		16	trnbid	10	新しく発生したグローバルトランザクションの、トランザクシ ョンランチ識別子

表 9-15 dc_trn_unchained_commit_s 関数の呼び出し情報 (関数コード : b)

種類	長さ (バイト)	領域名	位置 (16 進)	内容	
関数入り口 情報	36	4	cltid	0	dc_trn_unchained_commit_s 関数の引数 cltid 指定値 dc_trn_unchained_commit 関数の場合、TP1/Client が使用す る領域

種類	長さ (バイト)	領域名	位置 (16進)	内容	
関数入り口 情報	36	16	trngid	4	現在のグローバルトランザクションの、トランザクショングローバル識別子
		16	trnbid	14	現在のグローバルトランザクションの、トランザクションブランチ識別子

注

関数出口情報はありませぬ。

表 9-16 dc_trn_unchained_rollback_s 関数の呼び出し情報 (関数コード : c)

種類	長さ (バイト)	領域名	位置 (16進)	内容	
関数入り口 情報	36	4	cltid	0	dc_trn_unchained_rollback_s 関数の引数 cltid 指定値 dc_trn_unchained_rollback 関数の場合、TP1/Client が使用する領域
		16	trngid	4	現在のグローバルトランザクションの、トランザクショングローバル識別子
		16	trnbid	14	現在のグローバルトランザクションの、トランザクションブランチ識別子

注

関数出口情報はありませぬ。

表 9-17 dc_trn_info_s 関数の呼び出し情報 (関数コード : d)

種類	長さ (バイト)	領域名	位置 (16進)	内容	
関数入り口 情報	8	4	cltid	0	dc_trn_info_s 関数の引数 cltid 指定値 dc_trn_info 関数の場合、TP1/Client が使用する領域
		4	flags	4	dc_trn_info_s 関数の引数 flags 指定値 (アドレス)

注

関数出口情報はありませぬ。

表 9-18 dc_clt_get_trnid_s 関数の呼び出し情報 (関数コード : e)

種類	長さ (バイト)	領域名	位置 (16進)	内容	
関数入り口 情報	4	4	cltid	0	dc_clt_get_trnid_s 関数の引数 cltid 指定値 dc_clt_get_trnid 関数の場合、TP1/Client が使用する領域
関数出口 情報	32	16	trngid	0	dc_clt_get_trnid_s 関数の引数 trngid に返された値
		16	trnbid	10	dc_clt_get_trnid_s 関数の引数 trnbid に返された値

表 9-19 dc_rpc_get_watch_time_s 関数の呼び出し情報 (関数コード : f)

種類	長さ (バイト)		領域名	位置 (16 進)	内容
関数入り口 情報	4	4	cltid	0	dc_rpc_get_watch_time_s 関数の引数 cltid 指定値 dc_rpc_get_watch_time 関数の場合、TP1/Client が使用する 領域

注

関数出口情報はありませぬ。

表 9-20 dc_rpc_set_watch_time_s 関数の呼び出し情報 (関数コード : 10)

種類	長さ (バイト)		領域名	位置 (16 進)	内容
関数入り口 情報	8	4	cltid	0	dc_rpc_set_watch_time_s 関数の引数 cltid 指定値 dc_rpc_set_watch_time 関数の場合、TP1/Client が使用する 領域
		4	var	4	dc_rpc_set_watch_time_s 関数の引数 var 指定値

注

関数出口情報はありませぬ。

表 9-21 dc_clt_connect_s 関数の呼び出し情報 (関数コード : 13)

種類	長さ (バイト)		領域名	位置 (16 進)	内容
関数入り口 情報	8	4	cltid	0	dc_clt_connect_s 関数の引数 cltid 指定値 dc_clt_connect 関数の場合、TP1/Client が使用する領域
		4	flags	4	dc_clt_connect_s 関数の引数 flags 指定値

注

関数出口情報はありませぬ。

表 9-22 dc_clt_disconnect_s 関数の呼び出し情報 (関数コード : 14)

種類	長さ (バイト)		領域名	位置 (16 進)	内容
関数入り口 情報	8	4	cltid	0	dc_clt_disconnect_s 関数の引数 cltid 指定値 dc_clt_disconnect 関数の場合、TP1/Client が使用する領域
		4	flags	4	dc_clt_disconnect_s 関数の引数 flags 指定値

注

関数出口情報はありませぬ。

表 9-23 dc_clt_receive2_s 関数の呼び出し情報 (関数コード : 17)

種類	長さ (バイト)	領域名	位置 (16 進)	内容	
関数入り口 情報	16	4	cltid	0	dc_clt_receive2_s 関数の引数 cltid 指定値 dc_clt_receive2 関数の場合, TP1/Client が使用する領域
		4	recvleng	4	dc_clt_receive2_s 関数の引数 recvleng 指定値
		4	timeout	8	dc_clt_receive2_s 関数の引数 timeout 指定値
		4	flags	C	dc_clt_receive2_s 関数の引数 flags 指定値
関数出口 情報	68	64	buff	0	dc_clt_receive2_s 関数の引数 buff に返された値
		4	recvleng	40	dc_clt_receive2_s 関数の引数 recvleng に返された値

表 9-24 dc_clt_set_connect_inf_s 関数の呼び出し情報 (関数コード : 18)

種類	長さ (バイト)	領域名	位置 (16 進)	内容	
関数入り口 情報	76	4	cltid	0	dc_clt_set_connect_inf_s 関数の引数 cltid 指定値 dc_clt_set_connect_inf 関数の場合, TP1/Client が使用する領域
		2	inf_len	4	dc_clt_set_connect_inf_s 関数の引数 inf_len 指定値
		2	—	6	未使用の領域
		64	inf	8	dc_clt_set_connect_inf_s 関数の引数 inf 指定値
		4	flags	48	dc_clt_set_connect_inf_s 関数の引数 flags 指定値

(凡例)

— : 該当しません。

注

関数出口情報はあります。

表 9-25 dc_rpc_call_to_s 関数の呼び出し情報 (関数コード : 19)

種類	長さ (バイト)	領域名	位置 (16 進)	内容	
関数入り口 情報	236	4	cltid	0	dc_rpc_call_to_s 関数の引数 cltid 指定値 dc_rpc_call_to 関数の場合, TP1/Client が使用する領域
		8	tbl_nid	4	未使用の領域 (0 クリア)
		64	tbl_hostnm	C	DCRPC_BINDING_TBL 構造体のメンバ hostnm 指定値
		2	tbl_portno	4C	DCRPC_BINDING_TBL 構造体のメンバ portno 指定値
		2	tbl_filler1	8E	未使用の領域

種類	長さ (バイト)	領域名	位置 (16進)	内容	
関数入り口 情報	236	4	tbl_flags	50	未使用の領域 (0x00000002)
		16	tbl_filler2	54	未使用の領域
		32	group	64	dc_rpc_call_to_s 関数の引数 group 指定値
		32	service	84	dc_rpc_call_to_s 関数の引数 service 指定値
		60	in	A4	dc_rpc_call_to_s 関数の引数 in 指定値
		4	in_len	E0	dc_rpc_call_to_s 関数の引数 in_len 指定値
		4	out_len	E4	dc_rpc_call_to_s 関数の引数 out_len 指定値
		4	flags	E8	dc_rpc_call_to_s 関数の引数 flags 指定値
関数出口 情報	64	60	out	0	dc_rpc_call_s 関数の引数 out に返された値
		4	out_len	3C	dc_rpc_call_s 関数の引数 out_len に返された値

表 9-26 dc_clt_accept_notification_s 関数の呼び出し情報 (関数コード : 100)

種類	長さ (バイト)	領域名	位置 (16進)	内容	
関数入り口 情報	84	4	hWnd	0	dc_clt_accept_notification_s 関数の引数 hWnd 指定値 dc_clt_accept_notification 関数の場合、0 クリア
		64	defpath	4	dc_clt_accept_notification_s 関数の引数 defpath 指定値 dc_clt_accept_notification 関数の場合、0 クリア
		4	inf_len	44	dc_clt_accept_notification_s 関数の引数 inf_len 指定値
		2	port	48	dc_clt_accept_notification_s 関数の引数 port 指定値
		2	—	4A	未使用の領域
		4	timeout	4C	dc_clt_accept_notification_s 関数の引数 timeout 指定値
		4	flags	50	dc_clt_accept_notification_s 関数の引数 flags 指定値
関数出口 情報	140	64	inf	0	dc_clt_accept_notification_s 関数の引数 inf に返された値
		4	inf_len	40	dc_clt_accept_notification_s 関数の引数 inf_len に返された値
		64	hostname	44	dc_clt_accept_notification_s 関数の引数 hostname に返された値
		8	nodeid	84	dc_clt_accept_notification_s 関数の引数 nodeid に返された値

(凡例)

— : 該当しません。

表 9-27 dc_clt_cancel_notification_s 関数の呼び出し情報 (関数コード : 101)

種類	長さ (バイト)		領域名	位置 (16 進)	内容
関数入り口 情報	208	4	hWnd	0	dc_clt_cancel_notification_s 関数の引数 hWnd 指定値 dc_clt_cancel_notification 関数の場合, 0 クリア
		64	defpath	4	dc_clt_cancel_notification_s 関数の引数 defpath 指定値 dc_clt_cancel_notification 関数の場合, 0 クリア
		64	inf	44	dc_clt_cancel_notification_s 関数の引数 inf 指定値
		4	inf_len	84	dc_clt_cancel_notification_s 関数の引数 inf_len 指定値
		2	port	88	dc_clt_cancel_notification_s 関数の引数 port 指定値
		2	—	8A	未使用の領域
		64	hostname	8C	dc_clt_cancel_notification_s 関数の引数 hostname 指定値
		4	flags	CC	dc_clt_cancel_notification_s 関数の引数 flags 指定値

(凡例)

— : 該当しません。

注

関数出口情報はあります。

表 9-28 dc_clt_open_notification_s 関数の呼び出し情報 (関数コード : 102)

種類	長さ (バイト)		領域名	位置 (16 進)	内容
関数入り口 情報	76	4	hWnd	0	dc_clt_open_notification_s 関数の引数 hWnd 指定値 dc_clt_open_notification 関数の場合, 0 クリア
		64	defpath	4	dc_clt_open_notification_s 関数の引数 defpath 指定値 dc_clt_open_notification 関数の場合, 0 クリア
		2	port	44	dc_clt_open_notification_s 関数の引数 port 指定値
		2	—	46	未使用の領域
		4	flags	48	dc_clt_open_notification_s 関数の引数 flags 指定値
関数出口 情報	4	4	ntfid	0	dc_clt_open_notification_s 関数の引数 ntfid に返された値 dc_clt_open_notification 関数の場合, TP1/Client が使用する 領域

(凡例)

— : 該当しません。

表 9-29 dc_clt_close_notification_s 関数の呼び出し情報 (関数コード : 103)

種類	長さ (バイト)	領域名	位置 (16 進)	内容	
関数入り口 情報	8	4	ntfid	0	dc_clt_close_notification_s 関数の引数 ntfid 指定値 dc_clt_close_notification 関数の場合, TP1/Client が使用する 領域
		4	flags	4	dc_clt_close_notification_s 関数の引数 flags 指定値

注

関数出口情報はありせん。

表 9-30 dc_clt_chained_accept_notification_s 関数の呼び出し情報 (関数コード : 104)

種類	長さ (バイト)	領域名	位置 (16 進)	内容	
関数入り口 情報	16	4	ntfid	0	dc_clt_chained_accept_notification_s 関数の引数 ntfid 指定値 dc_clt_chained_accept_notification 関数の場合, TP1/Client が使用する領域
		4	inf_len	4	dc_clt_chained_accept_notification_s 関数の引数 inf_len 指 定値
		4	timeout	8	dc_clt_chained_accept_notification_s 関数の引数 timeout 指 定値
		4	flags	C	dc_clt_chained_accept_notification_s 関数の引数 flags 指定値
関数出口 情報	140	64	inf	0	dc_clt_chained_accept_notification_s 関数の引数 inf に返され た値
		4	inf_len	40	dc_clt_chained_accept_notification_s 関数の引数 inf_len に返 された値
		64	hostname	44	dc_clt_chained_accept_notification_s 関数の引数 hostname に返された値
		8	nodeid	84	dc_clt_chained_accept_notification_s 関数の引数 nodeid に 返された値

表 9-31 tpalloc 関数の呼び出し情報 (関数コード : 200)

種類	長さ (バイト)	領域名	位置 (16 進)	内容	
関数入り口 情報	20	8	type	0	tpalloc 関数の引数 type 指定値
		8	subtype	8	tpalloc 関数の引数 subtype 指定値
		4	size	10	tpalloc 関数の引数 size 指定値
関数出口 情報	8	4	return	0	tpalloc 関数のリターン値 (アドレス)

種類	長さ (バイト)		領域名	位置 (16 進)	内容
関数出口 情報	8	4	tperrno	4	tperrno に設定された値

表 9-32 tpfree 関数の呼び出し情報 (関数コード : 201)

種類	長さ (バイト)		領域名	位置 (16 進)	内容
関数入り口 情報	4	4	ptr	0	tpfree 関数の引数 ptr 指定値 (アドレス)

注

関数出口情報はありません。

表 9-33 tpconnect 関数の呼び出し情報 (関数コード : 202)

種類	長さ (バイト)		領域名	位置 (16 進)	内容
関数入り口 情報	100	32	svc	0	tpconnect 関数の引数 svc 指定値
		60	data	20	tpconnect 関数の引数 data 指定値
		4	len	5C	tpconnect 関数の引数 len 指定値
		4	flags	60	tpconnect 関数の引数 flags 指定値
関数出口 情報	4	4	tperrno	0	tperrno に設定された値

表 9-34 tpdison 関数の呼び出し情報 (関数コード : 203)

種類	長さ (バイト)		領域名	位置 (16 進)	内容
関数入り口 情報	4	4	cd	0	tpdiscon 関数の引数 cd 指定値
関数出口 情報	4	4	tperrno	0	tperrno に設定された値

表 9-35 tpSEND 関数の呼び出し情報 (関数コード : 204)

種類	長さ (バイト)		領域名	位置 (16 進)	内容
関数入り口 情報	72	4	cd	0	tpSEND 関数の引数 cd 指定値
		60	data	4	tpSEND 関数の引数 data 指定値
		4	len	40	tpSEND 関数の引数 len 指定値

種類	長さ (バイト)	領域名	位置 (16進)	内容	
関数入り口 情報	72	4	flags	44	tpsend 関数の引数 flags 指定値
関数出口 情報	12	4	revent	0	tpsend 関数の引数 revent に返された値
		4	tpurcode	4	tpurcode に設定された値
		4	tperrno	8	tperrno に設定された値

表 9-36 tprecv 関数の呼び出し情報 (関数コード : 205)

種類	長さ (バイト)	領域名	位置 (16進)	内容	
関数入り口 情報	12	4	cd	0	tprecv 関数の引数 cd 指定値
		4	len	4	tprecv 関数の引数 len 指定値
		4	flags	8	tprecv 関数の引数 flags 指定値
関数出口 情報	76	60	data	0	tprecv 関数の引数 data に返された値
		4	len	3C	tprecv 関数の引数 len に返された値
		4	revent	40	tprecv 関数の引数 revent に返された値
		4	tpurcode	44	tpurcode に設定された値
		4	tperrno	48	tperrno に設定された値

(6) 出力形式 (TP1/Client/P(64)および TP1/Client/W(64)の場合)

表 9-37 dc_rpc_open_s 関数の呼び出し情報 (関数コード : 1)

種類	長さ (バイト)	領域名	位置 (16進)	内容	
関数入り口 情報	16	8	cltid	0	dc_rpc_open_s 関数の引数 cltid 指定値 dc_rpc_open 関数の場合, TP1/Client が使用する領域
		4	flags	8	dc_rpc_open_s 関数の引数 flags 指定値
		4	filler	C	未使用の領域

注

関数出口情報はありませぬ。

表 9-38 dc_rpc_close_s 関数の呼び出し情報 (関数コード : 2)

種類	長さ (バイト)	領域名	位置 (16 進)	内容	
関数入り口 情報	16	8	cltid	0	dc_rpc_close_s 関数の引数 cltid 指定値 dc_rpc_close 関数の場合, TP1/Client が使用する領域
		4	flags	8	dc_rpc_close_s 関数の引数 flags 指定値
		4	filler	C	未使用の領域

注

関数出口情報はありませぬ。

表 9-39 dc_rpc_call_s 関数の呼び出し情報 (関数コード : 3)

種類	長さ (バイト)	領域名	位置 (16 進)	内容	
関数入り口 情報	144	8	cltid	0	dc_rpc_call_s 関数の引数 cltid 指定値 dc_rpc_call 関数の場合, TP1/Client が使用する領域
		32	group	8	dc_rpc_call_s 関数の引数 group 指定値
		32	service	28	dc_rpc_call_s 関数の引数 service 指定値
		60	in	48	dc_rpc_call_s 関数の引数 in 指定値
		4	in_len	84	dc_rpc_call_s 関数の引数 in_len 指定値
		4	out_len	88	dc_rpc_call_s 関数の引数 out_len 指定値
		4	flags	8C	dc_rpc_call_s 関数の引数 flags 指定値
関数出口 情報	64	60	out	0	dc_rpc_call_s 関数の引数 out に返された値
		4	out_len	3C	dc_rpc_call_s 関数の引数 out_len に返された値

表 9-40 dc_clt_cltin_s 関数の呼び出し情報 (関数コード : 4)

種類	長さ (バイト)	領域名	位置 (16 進)	内容	
関数入り口 情報	176	8	hWnd	0	dc_clt_cltin_s 関数の引数 hWnd の指定値 dc_clt_cltin 関数の場合, 0 クリア
		64	defpath	8	dc_clt_cltin_s 関数の引数 defpath の指定値 dc_clt_cltin 関数の場合, 0 クリア
		64	target_host	48	dc_clt_cltin_s 関数の引数 target_host の指定値
		32	logname	88	dc_clt_cltin_s 関数の引数 logname の指定値
		4	flags	A8	dc_clt_cltin_s 関数の引数 flags の指定値

種類	長さ (バイト)	領域名	位置 (16進)	内容	
関数入り口 情報	176	4	filler	AC	未使用の領域
関数出口 情報	72	8	cltid	0	dc_clt_cltin_s 関数の引数 cltid に返された値 dc_clt_cltin 関数の場合、TP1/Client が使用する領域
		64	set_host	8	dc_clt_cltin_s 関数の引数 set_host に返された値

表 9-41 dc_clt_cltout_s 関数の呼び出し情報 (関数コード : 5)

種類	長さ (バイト)	領域名	位置 (16進)	内容	
関数入り口 情報	8	8	cltid	0	dc_clt_cltout_s 関数の引数 cltid 指定値 dc_clt_cltout 関数の場合、TP1/Client が使用する領域
		4	flags	8	dc_clt_cltout_s 関数の引数 flags 指定値
		4	filler	C	未使用の領域

注

関数出口情報はあります。

表 9-42 dc_clt_send_s 関数の呼び出し情報 (関数コード : 6)

種類	長さ (バイト)	領域名	位置 (16進)	内容	
関数入り口 情報	152	8	cltid	0	dc_clt_send_s 関数の引数 cltid 指定値 dc_clt_send 関数の場合、TP1/Client が使用する領域
		64	buff	8	dc_clt_send_s 関数の引数 buff 指定値
		4	sendleng	48	dc_clt_send_s 関数の引数 sendleng 指定値
		64	hostname	4C	dc_clt_send_s 関数の引数 hostname 指定値
		2	portnum	8C	dc_clt_send_s 関数の引数 portnum 指定値
		2	—	8E	未使用の領域
		4	flags	90	dc_clt_send_s 関数の引数 flags 指定値
		4	filler	94	未使用の領域

(凡例)

— : 該当しません。

注

関数出口情報はあります。

表 9-43 dc_clt_receive_s 関数の呼び出し情報 (関数コード : 7)

種類	長さ (バイト)	領域名	位置 (16 進)	内容	
関数入り口 情報	24	8	cltid	0	dc_clt_receive_s 関数の引数 cltid 指定値 dc_clt_receive 関数の場合、TP1/Client が使用する領域
		4	recvleng	8	dc_clt_receive_s 関数の引数 recvleng 指定値
		4	timeout	C	dc_clt_receive_s 関数の引数 timeout 指定値
		4	flags	10	dc_clt_receive_s 関数の引数 flags 指定値
		4	filler	14	未使用の領域
関数出口 情報	64	64	buff	0	dc_clt_receive_s 関数の引数 buff に返された値

表 9-44 dc_trn_begin_s 関数の呼び出し情報 (関数コード : 8)

種類	長さ (バイト)	領域名	位置 (16 進)	内容	
関数入り口 情報	8	8	cltid	0	dc_trn_begin_s 関数の引数 cltid 指定値 dc_trn_begin 関数の場合、TP1/Client が使用する領域
関数出口 情報	32	16	trngid	0	発生したグローバルトランザクションの、トランザクショングローバル識別子
		16	trnbid	10	発生したグローバルトランザクションの、トランザクションブランチ識別子

表 9-45 dc_trn_chained_commit_s 関数の呼び出し情報 (関数コード : 9)

種類	長さ (バイト)	領域名	位置 (16 進)	内容	
関数入り口 情報	40	8	cltid	0	dc_trn_chained_commit_s 関数の引数 cltid 指定値 dc_trn_chained_commit 関数の場合、TP1/Client が使用する領域
		16	trngid	8	現在のグローバルトランザクションの、トランザクショングローバル識別子
		16	trnbid	18	現在のグローバルトランザクションの、トランザクションブランチ識別子
関数出口 情報	32	16	trngid	0	新しく発生したグローバルトランザクションの、トランザクショングローバル識別子
		16	trnbid	10	新しく発生したグローバルトランザクションの、トランザクションブランチ識別子

表 9-46 dc_clt_set_raphost_s 関数の呼び出し情報 (関数コード : 1a)

種類	長さ (バイト)	領域名	位置 (16 進)	内容	
関数入り口 情報	144	8	cltid	0	dc_clt_set_raphost_s 関数の引数 cltid 指定値 dc_clt_set_raphost 関数の場合, TP1/Client が使用する領域
		128	raphost	8	dc_clt_set_raphost_s 関数の引数 raphost 指定値
		4	flags	88	dc_clt_set_raphost_s 関数の引数 flags 指定値
		4	filler	8C	未使用の領域

注

関数出口情報はあります。

表 9-47 dc_clt_get_raphost_s 関数の呼び出し情報 (関数コード : 1b)

種類	長さ (バイト)	領域名	位置 (16 進)	内容	
関数入り口 情報	16	8	cltid	0	dc_clt_get_raphost_s 関数の引数 cltid 指定値 dc_clt_get_raphost 関数の場合, TP1/Client が使用する領域
		4	flags	8	dc_clt_get_raphost_s 関数の引数 flags 指定値
		4	filler	C	未使用の領域
関数出口 情報	128	128	raphost	0	dc_clt_get_raphost_s 関数の引数 raphost に返された値

表 9-48 dc_clt_assem_send_s 関数の呼び出し情報 (関数コード : 1c)

種類	長さ (バイト)	領域名	位置 (16 進)	内容	
関数入り口 情報	152	8	cltid	0	dc_clt_assem_send_s 関数の引数 cltid 指定値 dc_clt_assem_send 関数の場合, TP1/Client が使用する領域
		64	buff	8	dc_clt_assem_send_s 関数の引数 buff 指定値
		4	sendleng	48	dc_clt_assem_send_s 関数の引数 sendleng 指定値
		64	hostname	4C	dc_clt_assem_send_s 関数の引数 hostname 指定値
		2	portnum	8C	dc_clt_assem_send_s 関数の引数 portnum 指定値
		2	—	8E	未使用の領域
		4	timeout	90	dc_clt_assem_send_s 関数の引数 timeout 指定値
		4	flags	94	dc_clt_assem_send_s 関数の引数 flags 指定値

(凡例)

— : 該当しません。

注

関数出口情報はありません。

表 9-49 dc_clt_assem_receive_s 関数の呼び出し情報 (関数コード : 1d)

種類	長さ (バイト)		領域名	位置 (16 進)	内容
関数入り口 情報	24	8	cltid	0	dc_clt_assem_receive_s 関数の引数 cltid 指定値 dc_clt_assem_receive 関数の場合、TP1/Client が使用する領域
		4	recvleng	8	dc_clt_assem_receive_s 関数の引数 recvleng 指定値
		4	timeout	C	dc_clt_assem_receive_s 関数の引数 timeout 指定値
		4	flags	10	dc_clt_assem_receive_s 関数の引数 flags 指定値
		4	filler	14	未使用の領域
関数出口 情報	68	64	buff	0	dc_clt_assem_receive_s 関数の引数 buff に返された値
		4	recvleng	40	dc_clt_assem_receive_s 関数の引数 recvleng に返された値

表 9-50 dc_trn_chained_rollback_s 関数の呼び出し情報 (関数コード : a)

種類	長さ (バイト)		領域名	位置 (16 進)	内容
関数入り口 情報	40	8	cltid	0	dc_trn_chained_rollback_s 関数の引数 cltid 指定値 dc_trn_chained_rollback 関数の場合、TP1/Client が使用する領域
		16	trngid	8	現在のグローバルトランザクションの、トランザクショングローバル識別子
		16	trnbid	18	現在のグローバルトランザクションの、トランザクションブランチ識別子
関数出口 情報	32	16	trngid	0	新しく発生したグローバルトランザクションの、トランザクショングローバル識別子
		16	trnbid	10	新しく発生したグローバルトランザクションの、トランザクションブランチ識別子

表 9-51 dc_trn_unchained_commit_s 関数の呼び出し情報 (関数コード : b)

種類	長さ (バイト)		領域名	位置 (16 進)	内容
関数入り口 情報	40	8	cltid	0	dc_trn_unchained_commit_s 関数の引数 cltid 指定値 dc_trn_unchained_commit 関数の場合、TP1/Client が使用する領域
		16	trngid	8	現在のグローバルトランザクションの、トランザクショングローバル識別子

種類	長さ (バイト)	領域名	位置 (16進)	内容	
関数入り口 情報	40	16	trnbid	18	現在のグローバルトランザクションの、トランザクションブランチ識別子

注

関数出口情報はありませぬ。

表 9-52 dc_trn_unchained_rollback_s 関数の呼び出し情報 (関数コード : c)

種類	長さ (バイト)	領域名	位置 (16進)	内容	
関数入り口 情報	40	8	cltid	0	dc_trn_unchained_rollback_s 関数の引数 cltid 指定値 dc_trn_unchained_rollback 関数の場合、TP1/Client が使用する領域
		16	trngid	8	現在のグローバルトランザクションの、トランザクショングローバル識別子
		16	trnbid	18	現在のグローバルトランザクションの、トランザクションブランチ識別子

注

関数出口情報はありませぬ。

表 9-53 dc_trn_info_s 関数の呼び出し情報 (関数コード : d)

種類	長さ (バイト)	領域名	位置 (16進)	内容	
関数入り口 情報	16	8	cltid	0	dc_trn_info_s 関数の引数 cltid 指定値 dc_trn_info 関数の場合、TP1/Client が使用する領域
		8	flags	8	dc_trn_info_s 関数の引数 flags 指定値 (アドレス)

注

関数出口情報はありませぬ。

表 9-54 dc_clt_get_trnid_s 関数の呼び出し情報 (関数コード : e)

種類	長さ (バイト)	領域名	位置 (16進)	内容	
関数入り口 情報	8	8	cltid	0	dc_clt_get_trnid_s 関数の引数 cltid 指定値 dc_clt_get_trnid 関数の場合、TP1/Client が使用する領域
関数出口 情報	32	16	trngid	0	dc_clt_get_trnid_s 関数の引数 trngid に返された値
		16	trnbid	10	dc_clt_get_trnid_s 関数の引数 trnbid に返された値

表 9-55 dc_rpc_get_watch_time_s 関数の呼び出し情報 (関数コード : f)

種類	長さ (バイト)		領域名	位置 (16 進)	内容
関数入り口 情報	8	8	cltid	0	dc_rpc_get_watch_time_s 関数の引数 cltid 指定値 dc_rpc_get_watch_time 関数の場合、TP1/Client が使用する 領域

注

関数出口情報はありません。

表 9-56 dc_rpc_set_watch_time_s 関数の呼び出し情報 (関数コード : 10)

種類	長さ (バイト)		領域名	位置 (16 進)	内容
関数入り口 情報	16	8	cltid	0	dc_rpc_set_watch_time_s 関数の引数 cltid 指定値 dc_rpc_set_watch_time 関数の場合、TP1/Client が使用する 領域
		4	var	8	dc_rpc_set_watch_time_s 関数の引数 var 指定値
		4	filler	C	未使用の領域

注

関数出口情報はありません。

表 9-57 dc_clt_connect_s 関数の呼び出し情報 (関数コード : 13)

種類	長さ (バイト)		領域名	位置 (16 進)	内容
関数入り口 情報	16	8	cltid	0	dc_clt_connect_s 関数の引数 cltid 指定値 dc_clt_connect 関数の場合、TP1/Client が使用する領域
		4	flags	8	dc_clt_connect_s 関数の引数 flags 指定値
		4	filler	C	未使用の領域

注

関数出口情報はありません。

表 9-58 dc_clt_disconnect_s 関数の呼び出し情報 (関数コード : 14)

種類	長さ (バイト)		領域名	位置 (16 進)	内容
関数入り口 情報	16	8	cltid	0	dc_clt_disconnect_s 関数の引数 cltid 指定値 dc_clt_disconnect 関数の場合、TP1/Client が使用する領域
		4	flags	8	dc_clt_disconnect_s 関数の引数 flags 指定値
		4	filler	C	未使用の領域

注

関数出口情報はあります。

表 9-59 dc_clt_receive2_s 関数の呼び出し情報 (関数コード : 17)

種類	長さ (バイト)	領域名	位置 (16 進)	内容	
関数入り口 情報	24	8	cltid	0	dc_clt_receive2_s 関数の引数 cltid 指定値 dc_clt_receive2 関数の場合, TP1/Client が使用する領域
		4	recvleng	8	dc_clt_receive2_s 関数の引数 recvleng 指定値
		4	timeout	C	dc_clt_receive2_s 関数の引数 timeout 指定値
		4	flags	10	dc_clt_receive2_s 関数の引数 flags 指定値
		4	filler	14	未使用の領域
関数出口 情報	68	64	buff	0	dc_clt_receive2_s 関数の引数 buff に返された値
		4	recvleng	40	dc_clt_receive2_s 関数の引数 recvleng に返された値

表 9-60 dc_clt_set_connect_inf_s 関数の呼び出し情報 (関数コード : 18)

種類	長さ (バイト)	領域名	位置 (16 進)	内容	
関数入り口 情報	80	8	cltid	0	dc_clt_set_connect_inf_s 関数の引数 cltid 指定値 dc_clt_set_connect_inf 関数の場合, TP1/Client が使用する 領域
		2	inf_len	8	dc_clt_set_connect_inf_s 関数の引数 inf_len 指定値
		2	—	A	未使用の領域
		64	inf	C	dc_clt_set_connect_inf_s 関数の引数 inf 指定値
		4	flags	4C	dc_clt_set_connect_inf_s 関数の引数 flags 指定値

(凡例)

— : 該当しません。

注

関数出口情報はあります。

表 9-61 dc_rpc_call_to_s 関数の呼び出し情報 (関数コード : 19)

種類	長さ (バイト)	領域名	位置 (16 進)	内容	
関数入り口 情報	240	8	cltid	0	dc_rpc_call_to_s 関数の引数 cltid 指定値 dc_rpc_call_to 関数の場合, TP1/Client が使用する領域
		8	tbl_nid	8	未使用の領域 (0 クリア)

種類	長さ (バイト)	領域名	位置 (16進)	内容	
関数入り口 情報	240	64	tbl_hostnm	10	DCRPC_BINDING_TBL 構造体のメンバ hostnm 指定値
		2	tbl_portno	50	DCRPC_BINDING_TBL 構造体のメンバ portno 指定値
		2	tbl_filler1	52	未使用の領域
		4	tbl_flags	54	未使用の領域 (0x00000002)
		16	tbl_filler2	58	未使用の領域
		32	group	68	dc_rpc_call_to_s 関数の引数 group 指定値
		32	service	88	dc_rpc_call_to_s 関数の引数 service 指定値
		60	in	A8	dc_rpc_call_to_s 関数の引数 in 指定値
		4	in_len	E4	dc_rpc_call_to_s 関数の引数 in_len 指定値
		4	out_len	E8	dc_rpc_call_to_s 関数の引数 out_len 指定値
関数出口 情報	64	60	out	0	dc_rpc_call_s 関数の引数 out に返された値
		4	out_len	3C	dc_rpc_call_s 関数の引数 out_len に返された値

表 9-62 dc_clt_accept_notification_s 関数の呼び出し情報 (関数コード : 100)

種類	長さ (バイト)	領域名	位置 (16進)	内容	
関数入り口 情報	88	8	hWnd	0	dc_clt_accept_notification_s 関数の引数 hWnd 指定値 dc_clt_accept_notification 関数の場合、0 クリア
		64	defpath	8	dc_clt_accept_notification_s 関数の引数 defpath 指定値 dc_clt_accept_notification 関数の場合、0 クリア
		4	inf_len	48	dc_clt_accept_notification_s 関数の引数 inf_len 指定値
		2	port	4C	dc_clt_accept_notification_s 関数の引数 port 指定値
		2	—	4E	未使用の領域
		4	timeout	50	dc_clt_accept_notification_s 関数の引数 timeout 指定値
		4	flags	54	dc_clt_accept_notification_s 関数の引数 flags 指定値
関数出口 情報	140	64	inf	0	dc_clt_accept_notification_s 関数の引数 inf に返された値
		4	inf_len	40	dc_clt_accept_notification_s 関数の引数 inf_len に返された値
		64	hostname	44	dc_clt_accept_notification_s 関数の引数 hostname に返された値
		8	nodeid	84	dc_clt_accept_notification_s 関数の引数 nodeid に返された値

(凡例)

－：該当しません。

表 9-63 dc_clt_cancel_notification_s 関数の呼び出し情報 (関数コード：101)

種類	長さ (バイト)	領域名	位置 (16 進)	内容	
関数入り口 情報	216	8	hWnd	0	dc_clt_cancel_notification_s 関数の引数 hWnd 指定値 dc_clt_cancel_notification 関数の場合、0 クリア
		64	defpath	8	dc_clt_cancel_notification_s 関数の引数 defpath 指定値 dc_clt_cancel_notification 関数の場合、0 クリア
		64	inf	48	dc_clt_cancel_notification_s 関数の引数 inf 指定値
		4	inf_len	88	dc_clt_cancel_notification_s 関数の引数 inf_len 指定値
		2	port	8C	dc_clt_cancel_notification_s 関数の引数 port 指定値
		2	－	8E	未使用の領域
		64	hostname	90	dc_clt_cancel_notification_s 関数の引数 hostname 指定値
		4	flags	D0	dc_clt_cancel_notification_s 関数の引数 flags 指定値
4	filler	D4	未使用の領域		

(凡例)

－：該当しません。

注

関数出口情報はありません。

表 9-64 dc_clt_open_notification_s 関数の呼び出し情報 (関数コード：102)

種類	長さ (バイト)	領域名	位置 (16 進)	内容	
関数入り口 情報	80	8	hWnd	0	dc_clt_open_notification_s 関数の引数 hWnd 指定値 dc_clt_open_notification 関数の場合、0 クリア
		64	defpath	8	dc_clt_open_notification_s 関数の引数 defpath 指定値 dc_clt_open_notification 関数の場合、0 クリア
		2	port	48	dc_clt_open_notification_s 関数の引数 port 指定値
		2	－	4A	未使用の領域
		4	flags	4C	dc_clt_open_notification_s 関数の引数 flags 指定値
関数出口 情報	4	4	ntfid	0	dc_clt_open_notification_s 関数の引数 ntfid に返された値 dc_clt_open_notification 関数の場合、TP1/Client が使用する 領域

(凡例)

－：該当しません。

表 9-65 dc_clt_close_notification_s 関数の呼び出し情報 (関数コード : 103)

種類	長さ (バイト)	領域名	位置 (16 進)	内容	
関数入り口 情報	16	8	ntfid	0	dc_clt_close_notification_s 関数の引数 ntfid 指定値 dc_clt_close_notification 関数の場合, TP1/Client が使用する 領域
		4	flags	8	dc_clt_close_notification_s 関数の引数 flags 指定値
		4	filler	C	未使用の領域

注

関数出口情報はありませぬ。

表 9-66 dc_clt_chained_accept_notification_s 関数の呼び出し情報 (関数コード : 104)

種類	長さ (バイト)	領域名	位置 (16 進)	内容	
関数入り口 情報	24	8	ntfid	0	dc_clt_chained_accept_notification_s 関数の引数 ntfid 指定値 dc_clt_chained_accept_notification 関数の場合, TP1/Client が使用する領域
		4	inf_len	8	dc_clt_chained_accept_notification_s 関数の引数 inf_len 指 定値
		4	timeout	C	dc_clt_chained_accept_notification_s 関数の引数 timeout 指 定値
		4	flags	10	dc_clt_chained_accept_notification_s 関数の引数 flags 指定値
		4	filler	14	未使用の領域
関数出口 情報	140	64	inf	0	dc_clt_chained_accept_notification_s 関数の引数 inf に返され た値
		4	inf_len	40	dc_clt_chained_accept_notification_s 関数の引数 inf_len に返 された値
		64	hostname	44	dc_clt_chained_accept_notification_s 関数の引数 hostname に返された値
		8	nodeid	84	dc_clt_chained_accept_notification_s 関数の引数 nodeid に 返された値

表 9-67 tpallocc 関数の呼び出し情報 (関数コード : 200)

種類	長さ (バイト)	領域名	位置 (16 進)	内容	
関数入り口 情報	20	8	type	0	tpallocc 関数の引数 type 指定値
		8	subtype	8	tpallocc 関数の引数 subtype 指定値

種類	長さ (バイト)		領域名	位置 (16進)	内容
関数入り口 情報	20	4	size	10	tpalloc 関数の引数 size 指定値
関数出口 情報	16	8	return	0	tpalloc 関数のリターン値 (アドレス)
		4	tperrno	8	tperrno に設定された値
		4	filler	C	未使用の領域

表 9-68 tpfree 関数の呼び出し情報 (関数コード : 201)

種類	長さ (バイト)		領域名	位置 (16進)	内容
関数入り口 情報	8	8	ptr	0	tpfree 関数の引数 ptr 指定値 (アドレス)

注

関数出口情報はあります。

表 9-69 tpconnect 関数の呼び出し情報 (関数コード : 202)

種類	長さ (バイト)		領域名	位置 (16進)	内容
関数入り口 情報	100	32	svc	0	tpconnect 関数の引数 svc 指定値
		60	data	20	tpconnect 関数の引数 data 指定値
		4	len	5C	tpconnect 関数の引数 len 指定値
		4	flags	60	tpconnect 関数の引数 flags 指定値
関数出口 情報	4	4	tperrno	0	tperrno に設定された値

表 9-70 tpdison 関数の呼び出し情報 (関数コード : 203)

種類	長さ (バイト)		領域名	位置 (16進)	内容
関数入り口 情報	4	4	cd	0	tpdison 関数の引数 cd 指定値
関数出口 情報	4	4	tperrno	0	tperrno に設定された値

表 9-71 tpsend 関数の呼び出し情報 (関数コード : 204)

種類	長さ (バイト)	領域名	位置 (16 進)	内容	
関数入り口 情報	72	4	cd	0	tpsend 関数の引数 cd 指定値
		60	data	4	tpsend 関数の引数 data 指定値
		4	len	40	tpsend 関数の引数 len 指定値
		4	flags	44	tpsend 関数の引数 flags 指定値
関数出口 情報	12	4	revent	0	tpsend 関数の引数 revent に返された値
		4	tpurcode	4	tpurcode に設定された値
		4	tpermo	8	tpermo に設定された値

表 9-72 tprecv 関数の呼び出し情報 (関数コード : 205)

種類	長さ (バイト)	領域名	位置 (16 進)	内容	
関数入り口 情報	12	4	cd	0	tprecv 関数の引数 cd 指定値
		4	len	4	tprecv 関数の引数 len 指定値
		4	flags	8	tprecv 関数の引数 flags 指定値
関数出口 情報	76	60	data	0	tprecv 関数の引数 data に返された値
		4	len	3C	tprecv 関数の引数 len に返された値
		4	revent	40	tprecv 関数の引数 revent に返された値
		4	tpurcode	44	tpurcode に設定された値
		4	tpermo	48	tpermo に設定された値

10

障害対策

障害が発生した場合の対処方法について説明します。

10.1 通信障害

TP1/Client の通信障害に関する注意事項および対処方法について次に示します。

- 通信障害発生時のエラーメッセージの出力

TP1/Client では、コネクション型の通信プロトコルである TCP を使用し、これをソケットインタフェースで実現しています。そのため、TP1/Client では通信障害をソケットインタフェースのシステムコールエラーとして検知します。システムコールエラーが発生した場合、TP1/Client は障害原因を究明するためのエラーメッセージを出力します。

エラーメッセージは、エラーログファイルに出力されます。エラーログファイルの詳細については、「[2.11.1 エラーログ機能](#)」を参照してください。

- ポートに関する通信障害

TP1/Client では、通信が発生するたびに、コネクションの確立・解放が発生します。このため、複数の CUP で `dc_clt_cltin_s` 関数および `dc_clt_cltout_s` 関数を繰り返したり、`dc_rpc_call_s` 関数を繰り返し行う処理を連続して発行したりすると、OS のポートを一時的に使い切り、通信に失敗することがあります。このような場合、ポートの拡張などを行って運用に耐えられる数値にチューニングしてください。または、ポートが解放され、使用可能状態になるまで時間を置いたあとで、再度通信を行ってください。

- `dc_rpc_call_s` 関数使用時の通信障害

`dc_rpc_call_s` 関数発行時に、通信先のスケジュールサービスが開始処理中、または終了処理中であった場合、ホストを切り替えて処理します。`dc_clt_cltin_s` 関数の引数 `target_host`、またはクライアント環境定義 `DCHOST` に指定された複数のホストのうち、どれかが起動されていない場合、`dc_rpc_call_s` 関数がリターンするまで時間が掛かることがあります。

常設コネクション確立中、またはトランザクションの範囲内で `dc_rpc_call_s` 関数を実行した場合、エラー応答受信時のホストの切り替えは行いません。

10.2 クライアント側の障害

PC の場合、電源のトラブルやオペレーションのミスによって簡単に処理が中止されることがあります。このような場合、RPC の実行要求を受け付け中、または SPP から応答を受信していたとしても、データは保証できません。

10.3 XDM/DCCM3 へ遠隔操作要求時の障害

10.3.1 XDM/DCCM3 へ遠隔操作要求時の DCRPCER_NET_DOWN

XDM/DCCM3 へ遠隔操作要求を行った場合、ネットワーク障害以外でも DCRPCER_NET_DOWN となります。ネットワーク障害以外で DCRPCER_NET_DOWN となる要因を次に示します。

- メッセージフォーマット不正
- トランザクション名称不正
- トランザクションの型と RPC コールの形式組み合わせ不正
- トランザクションの型と仮想端末の型との組み合わせ不正
- トランザクション入力禁止
- トランザクション閉塞
- 機密保護エラー
- 入力キュー障害

10.3.2 XDM/DCCM3 での二重入力エラー検出

TP1/Client で複数の CUP を同じ PC または WS で使用し、XDM/DCCM3 の相手ホスト固定・送受信兼用端末に対して、同期応答型 RPC を行った場合、XDM/DCCM3 のトランザクション種別を問い合わせ応答型にすると、二重入力エラーとなります。このような場合は、トランザクション種別を非問い合わせ応答型にしてください。

11

メッセージ

出力されるメッセージについて説明します。

この章では、各関数の DLL を呼び出すときの C 言語の関数名 (dc_xxx_xxx_s) で説明します。通常オブジェクトライブラリの関数および COBOL 言語を使う場合は、各関数に対応する通常オブジェクトライブラリの関数名 (dc_xxx_xxx) および COBOL 言語の要求文に置き換えて読んでください。

11.1 メッセージの出力形式

出力されるメッセージの形式を次に示します。

```
KFCAn1n2n3n4n5-X YY…………YY
```

KFCAn₁n₂n₃n₄n₅-X

メッセージ ID (半角英数字 11 文字)

YY…………YY

メッセージテキスト

11.2 メッセージの記述形式

ここでは、メッセージの記述形式と、メッセージ ID の記号について説明します。

11.2.1 記述形式

このマニュアルでのメッセージの記述形式を次に示します。

KFCAn1n2n3n4n5-X

メッセージテキスト (Y)

メッセージの意味を説明します。

(S)システムがメッセージを出力したあとにする主な処理を示します。

(O)メッセージ確認時、オペレータがする必要がある処置を示します。

【対策】メッセージ確認時の TP1/Client 管理者の処置を示します。

注

メッセージ中の、オペレータの処置または対策で「保守員に連絡してください。」とは、購入時の契約に基づいて、システム管理者が弊社問い合わせ窓口へ連絡することを示します。

11.2.2 メッセージ ID の記号

メッセージ ID の記号の意味を次に示します。

KFCA

OpenTP1 のメッセージであることを示します。

n₁n₂n₃n₄n₅

メッセージの通し番号を示します。

X

メッセージの種類を示します。メッセージの種類を次の表に示します。

表 11-1 メッセージの種類

種類	意味
E	TP1/Client ライブラリ、運用コマンドが働かない障害が起きたことを示しています。
	定義誤り、運用コマンドのオペランド指定誤りによって、動作できないことを示しています。
W	定義誤り、コマンドのオペランド指定誤りはありましたが、値を仮定して動作を続行することを示しています。
I	上記 E、W に該当しないメッセージで、単純に動作を示しています。

Y

メッセージの出力先種別を示します。一つのメッセージが、複数の出力先種別を持つ場合は、出力される可能性のある種別を '+' でつないであります。メッセージの出力先種別を次の表に示します。

表 11-2 メッセージの出力先種別

種類	出力先
E	標準エラー出力
S	標準出力
R	エラーログファイル

11.3 メッセージ一覧

KFCA02401-E

environment definition error, variable=aa....aa, reason=bb....bb (R)

クライアント環境定義の指定に誤りがあります。

aa....aa : 誤りがあるクライアント環境定義名

bb....bb : 理由

NO VALUE : 指定値がありません。

OUT OF RANGE : 指定できる範囲を超えています。

INVALID CHAR : 指定できない文字があります。

NET ENVIRONMENT : 指定内容がネットワーク環境と一致しません。

UNMATCH LENG : 指定値のけた数が不正です。

(S)実行中の処理を中止する, または省略値解析で処理を続行します。

(O)TP1/Client 管理者に連絡してください。

【対策】 定義の誤りを訂正してください。

KFCA02402-E

memory shortage, size=aa....aa, inf=bb....bb (R + E)

size に示すバイト数のメモリが確保できません。

aa....aa : 確保しようとしたバイト数

bb....bb : 保守情報

(S)実行中の処理を中止します。

(O)TP1/Client 管理者に連絡してください。

【対策】 メモリ所要量を再度見積もってください。メモリ所要量が十分である場合は, エラーログおよび各種トレースを取得し, 保守員に連絡してください。

KFCA02403-E

aa....aa version error, TP1/Client(bb....bb):object(cc....cc) (R)

CUP にリンクされている TP1/Client ライブラリと aa....aa に示す OpenTP1 プログラムは, バージョンが不整合なため, 処理を実行できません。

aa....aa : バージョンの不整合なプログラム名

NAM : TP1/Server の NAM

bb....bb : CUP にリンクされている TP1/Client ライブラリのバージョン番号

cc....cc : OpenTP1 のプログラムのバージョン番号

(S)実行中の処理を中止します。

(O)TP1/Client 管理者に連絡してください。

[対策]

- bb....bb < cc....cc の場合
CUP を再びリンクしてください。
- bb....bb > cc....cc の場合
TP1/Client が正しくインストールされているかどうかを見直してください。

KFCA02404-E

parameter error, func=aa....aa, item=bb....bb (R)

引数の指定に誤りがあるため、関数が実行できません。

aa....aa : エラーが発生した関数名 (_s 付き関数を実行した場合でも, _s 無し関数名が出力される場合があります)

bb....bb : 誤りがあった引数名

(S)実行が要求された関数をエラーリターンします。

(O)TP1/Client 管理者に連絡してください。

[対策] 引数の誤りを訂正してください。

KFCA02406-E

error occurred in TP1/Client, reason=aa....aa, inf=bb....bb (R)

TP1/Client の内部処理でエラーが発生しました。

aa....aa : エラーが発生した原因を表示します。

MEMORY : メモリ不足

RESOURCE : 資源不足

NETWORK : ネットワーク障害

UNEXPECT : 予期しないエラー

NO SUCH SERVICE：サーバに対する要求コード不正

TIMER：タイマ設定失敗

TIME OUT：サーバが開始中、停止中、またはサーバ処理でタイムアウト発生

bb....bb：保守情報

(S)実行中の処理を中止します。

(O)このメッセージが出力されたエラーログを記録し、TP1/Client 管理者に連絡してください。

【対策】 出力されたエラーログから原因を追求して、エラーの要因を取り除いてください。また、ネームサービスのポート番号を指定するオペランドに誤ってスケジュールサービスのポート番号を指定した場合も、reason=UNEXPECT が出力されます。このときはポート番号を見直してください。

KFCA02407-E

```
error occurred in TP1/Client, reason=aa....aa    (R + E)
```

TP1/Client でエラーが発生しました。

aa....aa：エラーが発生した原因として、次のどれかを表示します。

cltin not executed：

dc_clt_cltin 関数が実行されていない。

cltin already executed：

dc_clt_cltin 関数がすでに実行されている。

host name：

ホスト名が誤っている。

TP1/Server not up：

設定したサービスがあるノードで OpenTP1 が起動していない。

(S)実行中の処理を中止します。

(O)このメッセージが出力されたエラーログを記録し、エラーログから原因を追求し、エラーの要因を取り除いたあと、CUP を再び起動してください。

KFCA02410-E

```
TP1/Server replied error, code=aa....aa, IPaddr=bbbbbbbbb, port=cc....cc, inf=dd....dd    (R)
```

TP1/Server からエラー応答を受信しました。

aa....aa：TP1/Server が返したエラーコード

bbbbbbbbb：送信元 IP アドレス（16 進数字）

cc....cc : 送信元ポート番号

dd....dd : 保守情報

(S)次に示すどれかの処理を実行します。

- 実行中の処理を終了します。
- エラー応答を無視して応答メッセージの到着を待ちます。クライアント環境定義の最大応答待ち時間を経過しても応答メッセージが到着しない場合は、DCRPCER_TIMED_OUT でエラーリターンします。
- クライアント環境定義 DCHOSTCHANGE に Y を指定している場合は、窓口となる TP1/Server を切り替え、処理を続行します。

(O)このメッセージが出力されたエラーログを記録し、TP1/Client 管理者に連絡してください。

【対策】 送信元 IP アドレス、および送信元ポート番号から送信元プロセスを求め、TP1/Server 側で何らかの障害が発生しているか調査してください。

KFCA02411-E

```
invalid message received, IPaddr=aaaaaaaa, port=bb....bb, inf=cc....cc (R)
```

解析できないメッセージを受信しました。

aaaaaaaa : 送信元 IP アドレス (16 進数字)

bb....bb : 送信元ポート番号

cc....cc : 保守情報

(S)受信したメッセージを無視して、次のメッセージを待ちます。

(O)TP1/Client 管理者に連絡してください。

【対策】 送信元 IP アドレス、およびポート番号から送信元プロセスを求め、ネットワーク環境を見直してください。前回の OpenTP1 の関数がタイムアウトなどで受け取れなかった応答メッセージを、次の関数が受信した場合も出力します。この場合は無視してください。

KFCA02412-W

```
error-reply received, code=aa....aa, IPaddr=bb....bb,  
port=cc....cc, inf=dd....dd (R)
```

TP1/Server からエラー応答を受信しました。

aa....aa : TP1/Server が返したエラーコード

bb....bb : 送信元 IP アドレス (10 進ドット記法)

cc....cc：送信元ポート番号

dd....dd：保守情報

(S)次に示すどれかの処理を実行します。

- 実行中の処理を終了します。
- キャッシュの情報を更新したあと、再度通信を試みます。
- クライアント環境定義 DCHOSTCHANGE に Y を指定している場合は、窓口となる TP1/Server を切り替え、処理を続行します。

(O)このメッセージが出力されたエラーログを記録して、TP1/Client 管理者に連絡してください。

【対策】 送信元 IP アドレス、および送信元ポート番号から送信元プロセスを求め、TP1/Server 側で何らかの障害が発生しているかを調査してください。

KFCA02419-W

```
error occurred in data compression process, group=aa....aa, service=bb....bb,  
reason=cc....cc (R)
```

データ圧縮内部処理でエラーが発生しました。

aa....aa：要求先サービスグループ名

bb....bb：要求先サービス名

cc....cc：エラーの理由

MEMORY：メモリ不足

NOEFFECT：データ圧縮効果なし

UNEXPECT：予期しないエラー

VERSION：バージョン不一致

(S)データを圧縮しないでサービスを要求します。

(O)「エラーの理由」別に、次のように処置してください。

- 「メモリ不足」の場合
このメッセージの前に出力されたメッセージに従ってください。
- 「データ圧縮効果なし」の場合
圧縮前より圧縮後のデータの方が大きくなってしまうため、同じ CUP でほかにもこのメッセージが出力されていないか確認し、CUP 単位でデータ圧縮機能の使用・不使用を再び検討してください。
- 「予期しないエラー」の場合
TP1/Client 管理者に連絡してください。
- 「バージョン不一致」の場合

サービス要求先の TP1/Server Base がデータ圧縮可能なバージョン (03-03 以降) かどうかを確認してください。

【対策】 エラーの原因を調査し、必要に応じて保守員に連絡してください。

KFCA02420-E

```
protocol error, IPaddr=aaaaaaaa, port=bb....bb, reason=cc....cc (R)
```

TACT/KERNEL プロセス間通信プロトコルのヘッダに誤った情報を検出しました。

aaaaaaaa : 送信元 IP アドレス (16 進数字)

bb....bb : 送信元ポート番号

cc....cc : 理由

SIZE OVER :

TACT ヘッダ部として 12 バイトを超えるメッセージを受信しました。

SIZE SHORT :

TACT ヘッダ部として 12 バイト未満のメッセージを受信しました。

INF2 ERROR :

付加情報 2 の内容が誤っています。

(S)コネクションをクローズして、再び相手からのコネクション確立を待ちます。または、実行中の処理を中断し、関数をエラーリターンします。

(O)TP1/Client 管理者に連絡してください。

【対策】 このメッセージが出力された場合、次のケースが想定できます。

- 不正なメッセージを受信した。
- cc....cc が SIZE SHORT の場合、FIN パケット、または RESET パケットを受信した。
- cc....cc が SIZE SHORT の場合、TACT ヘッダが分割されて TCP バッファに送信された。

送信元 IP アドレス、およびポート番号から送信元プロセスを求め、ネットワーク環境を見直してください。

KFCA02421-E

```
invalid size, received size=aa....aa, size in head=bb....bb, IPaddr=cccccccc, port=dd....dd (R)
```

受信完了通知のデータ長が、TACT/KERNEL プロセス間通信プロトコルのヘッダのデータ長に満たないため、処理を中止します。

aa....aa : 受信データ長

bb....bb : ヘッダ中のデータ長

cccccccc : 送信元 IP アドレス (16 進数字)

dd....dd : 送信元ポート番号

(S)処理を中止します。

(O)TP1/Client 管理者に連絡してください。

【対策】 送信元 IP アドレス, および送信元ポート番号から送信元プロセスを求め, TP1/Server 側で何らかの障害が発生しているかを調査してください。

KFCA02431-E

```
servicegroup list error, file=aa....aa:bb....bb, reason=cc....cc (R)
```

クライアント環境定義 DCCLTSERVICEGROUPLIST に指定したファイルの内容に誤りがあります。

aa....aa : クライアント環境定義 DCCLTSERVICEGROUPLIST に指定したファイル名

bb....bb : 誤りのあった行番号

cc....cc : 理由

NO VALUE : 指定値がありません。

OUT OF RANGE : 指定できる範囲を超えています。

INVALID CHAR : 指定できない文字があります。

(S)誤りのあった行の内容を無視し, 処理を続行します。

(O)TP1/Client 管理者に連絡してください。

【対策】 誤りのあった行を訂正してください。

KFCA02444-E

```
communication error, func=aa....aa, errno=bb....bb (R)
```

通信障害が発生しました。

aa....aa : 障害が発生したソケットインタフェースの関数名

bb....bb : エラー番号

(S)処理を中止します。

(O)TP1/Client 管理者に連絡してください。

【対策】 関数名, エラー番号を参照して, 原因を調査してください。

KFCA02445-E

```
host resolution failed, func=aa....aa, errno=bb....bb, data=cc....cc (R)
```

ホスト解決に失敗しました。

aa....aa：障害が発生したソケットインタフェースの関数名
gethostbyname または gethostbyaddr が出力されます。

bb....bb：エラー番号

cc....cc：ホスト解決に失敗したホスト名、または IP アドレス（10 進ドット記法）

(S)実行中の処理を中止して関数がエラーリターンするか、またはエラーを無視して処理を続行します。

(O)TP1/Client 管理者に連絡してください。

【対策】 ホスト名または IP アドレスの指定が正しいかどうかを確認してください。ホスト名解決に DNS を使用している場合は、DNS サーバ間のネットワークまたは DNS サーバで障害が起きていないかどうかを確認してください。

KFCA02446-E

```
communication error occurred while aa....aa function was being executed, func=bb....bb,  
errno=cc....cc, c-info=dd....dd:ee....ee, s-info=ff....ff:gg....gg (R)
```

通信障害が発生しました。

aa....aa：実行関数名

bb....bb：障害が発生したソケットインタフェースの関数名

cc....cc：エラー番号

dd....dd：自 IP アドレス（10 進ドット記法）

ee....ee：自ポート

ff....ff：相手 IP アドレス（10 進ドット記法）

gg....gg：相手ポート

(S)処理を中止します。

(O)TP1/Client 管理者に連絡してください。

【対策】 このメッセージに出力された情報とサーバ側の資料を基に、原因を調査してください。

KFCA02447-E

```
timeout occurred while aa....aa function was being executed, sts=bb....bb, time=cc....cc, c-  
info=dd....dd:ee....ee, s-info=ff....ff:gg....gg (R)
```

タイムアウトが発生しました。

aa....aa：実行関数名

bb....bb：タイムアウトが発生したときの待ち状態

CONNECT：コネクションの確立応答待ち

ACCEPT：コネクションの確立要求待ち

RECV(T)：TACT ヘッダの受信待ち

RECV(D)：データの受信待ち

RECV(D2)：データの受信待ち

RECV(D3)：データの受信待ち

RECV(M)：メッセージ情報の受信待ち

RECV(R)：応答専用データの受信待ち

RECVFROM：ブロードキャスト時のデータの受信待ち

cc....cc：クライアント環境定義，または関数の引数に指定されたタイムアウト時間（秒）

dd....dd：自 IP アドレス（10 進ドット記法）

ee....ee：自ポート

ff....ff：相手 IP アドレス（10 進ドット記法）

gg....gg：相手ポート

(S)処理を中断します。

(O)TP1/Client 管理者に連絡してください。

【対策】 適切なタイムアウト値が指定されているかどうかを確認してください。適切な値が指定されていた場合、回線障害または何らかの理由によって、サーバ側の処理が遅延しているおそれがあります。このメッセージに出力された情報とサーバ側の資料を基に、原因を調査してください。

KFCA02449-E

```
communication error, func=aa....aa, errno=bb....bb, remote node addr=cc....cc, remote  
port=dd....dd (R)
```

通信障害が発生しました。

aa....aa：障害が発生したソケットインタフェースの関数名

bb....bb : エラー番号

cc....cc : 送信先 IP アドレス

dd....dd : 送信先ポート番号

(S)処理を中止します。

(O)TP1/Client 管理者に連絡してください。

【対策】 関数名, エラー番号を参照して, 原因を調査してください。

KFCA02450-W

```
The client-ID of a different thread has been specified. func=aa....aa, thread1=bb....bb,  
thread2=cc....cc (R)
```

異なるスレッドのクライアント ID を指定しています。

このメッセージが出力されるタイミングは, dc_clt_cltin_s 関数の実行後, 最初の一度だけです。

aa....aa : クライアント関数名

bb....bb : クライアント ID を取得したスレッドのスレッド ID

cc....cc : クライアント関数を実行したスレッドのスレッド ID

(S)実行中の処理を続行します。

(O)このメッセージが出力されたエラーログを記録して, TP1/Client 管理者に連絡してください。

【対策】 スレッドをわたってクライアント ID を使用しています。正しいクライアント ID を指定して関数を実行するように CUP を修正してください。

KFCA02451-W

```
The function was executed twice.  
func1=aa....aa, thread1=bb....bb, func2=cc....cc, thread2=dd....dd (R)
```

クライアント関数が二重に実行されました。

このメッセージが出力されるタイミングは, dc_clt_cltin_s 関数の実行後, 最初の一度だけです。

aa....aa : 二重実行を検知したクライアント関数名

bb....bb : 二重実行を検知した関数を実行したスレッド ID*

cc....cc : 実行中のクライアント関数名

dd...dd : 実行中の関数を実行したスレッド ID*

注※

スレッド ID の値は TP1/Client/W では 0 になります。

(S)実行中の処理を続行します。

(O)このメッセージが出力されたエラーログを記録して、TP1/Client 管理者に連絡してください。

【対策】 同じクライアント ID を指定したクライアント関数が同時に実行されました。関数を二重に実行しないように CUP を修正してください。

KFCA02460-I

```
usage: aaaaaaa [bb pathname] (E + S)
```

トレース編集出力コマンドの形式です。トレース編集出力コマンドの指定に誤りがあった場合に出力されません。

aaaaaaa : UAP トレース編集出力コマンド

bb : オプション

TP1/Client/W の場合

```
usage : cltdump [-u|-s|-m] [-n] [-f pathname]
```

TP1/Client/P の場合

```
usage : cltdmp32 [/u|/s|/m] [/n] [/f pathname]
```

(S)処理を中止します。

(O)正しい形式で、再びコマンドを実行してください。

KFCA02461-E

```
file(aa....aa)not found (E)
```

トレース編集出力コマンド実行時に指定したファイルが見つかりません。

aa....aa : ユーザが指定したファイル名

(S)処理を中止します。

(O)正しいファイル名を指定し、再びコマンドを実行してください。

KFCA02462-E

```
no file to edit: aa....aa (E)
```

編集出力するトレースファイルが見つかりません。

aa....aa : 編集出力しようとしたトレース種別

UAP : UAP トレース

SOC : ソケットトレース

MDL : モジュールトレース

(S)処理を中止します。

(O)CUP 実行時トレースファイルが作成されなかった場合は、クライアント環境定義の指定に誤りがあるおそれがあります。

次の定義を見直してください。

UAP の場合 : DCTRCUAP

SOC の場合 : DTRCSOC

MDL の場合 : DTRCMDL

TP1/Client/W の場合 : DTRCPATH

指定に誤りがあった場合は、正しく指定し、再び CUP を実行してください。

KFCA02463-E

```
error occurred, file=aa....aa    (E)
```

トレース編集出力コマンド実行時、エラーが発生しました。

aa....aa : エラーが発生したファイル名

(S)実行中の処理を中止します。

(O)TP1/Client 管理者に連絡してください。

[対策] エラーの原因を調査し、必要に応じて保守員に連絡してください。

KFCA02466-I

```
TP1/Server Base ready, hostname=aa....aa, port=bb....bb    (R)
```

窓口となる TP1/Server (ネームサービスの問い合わせ先ホスト) を決定しました。なお、窓口となる TP1/Server は、障害の内容によって切り替わることがあります。

aa....aa : 窓口となる TP1/Server のホスト名

bb....bb : 窓口となる TP1/Server のポート番号 (ネームサービスのポート番号)

KFCA02468-I

```
TP1/Client changed connected TP1/Server Base, hostname=aa....aa, port=bb....bb (R)
```

通信先ホストを切り替えました。

aa....aa : 切り替え後のホスト名

bb....bb : 切り替え後のポート番号

KFCA02470-E

```
the error commitment for transaction(aaaaaaaabbbbbbbb) occurred, return value=cc....cc  
(R)
```

非連鎖モードのコミット，またはロールバックを行わずに dc_rpc_close_s 関数，または dc_clt_cltout_s 関数を実行しようとしたため，TP1/Client が自動的に非連鎖モードのコミットを行ったところ，同期点処理にエラーが発生しました。

aaaaaaa : OpenTP1 システムノード ID (8 文字の文字列)

bbbbbbbb : グローバルトランザクション番号 (8 文字の 16 進数列)

cc....cc : 同期点処理に対するリターン値 (4 けたの負数)

(S)処理を続行します。

(O)TP1/Client 管理者に連絡してください。

[対策]

1. 同期点処理にエラーが発生した場合の処理を行ってください。
2. CUP の処理を非連鎖モードのコミット，またはロールバック要求を行ったあとで，終了処理を行うようにしてください。

KFCA02471-W

```
Current status is outside a transaction. trngid=aa.....aa, trnbid=bb.....bb (R)
```

現在のトランザクションのステータスは，範囲外です。

障害によって，トランザクション実行プロセスとの接続はすでに切断されているため，dc_trn_unchained_commit_s 関数，または dc_trn_unchained_rollback_s 関数は DCCLTER_OLTF_NOT_UP でエラーリターンしました。

aa.....aa : 障害が発生したトランザクショングローバル識別子

bb.....bb : 障害が発生したトランザクションブランチ識別子

(S)実行した関数は DCCLTER_OLTF_NOT_UP でエラーリターンし、ステータスはトランザクションの範囲内から範囲外へ変わります。

(O)TP1/Client 管理者に連絡してください。

【対策】 このメッセージの直前に出力されたメッセージからエラーの原因を追求し、エラーの要因を取り除いてください。原因がタイムアウトの場合、サーバより先にクライアントがタイムアウトしないように、定義や指定値などを見直してください。

再びトランザクションを開始させるためには、dc_trn_begin_s 関数を実行してください。ただし、トランザクション問い合わせ間隔最大時間が満了するまでは、trngid および trnbid に示す前回のトランザクションは処理中のままとなっていますので、ご注意ください。

KFCA02472-E

The function has been issued from an invalid context. func=aa....aa (R)

func に示す関数が、誤ったコンテキストから呼び出されました。

障害によって、トランザクション実行プロセスとの接続はすでに切断されているため、func に示す関数は DCCLTER_OLTF_NOT_UP、または DCRPCER_OLTF_NOT_UP でエラーリターンしました。

aa....aa : 関数名

(S)実行した関数は DCCLTER_OLTF_NOT_UP、または DCRPCER_OLTF_NOT_UP でエラーリターンします。ステータスは、トランザクションの範囲内のままです。

(O)TP1/Client 管理者に連絡してください。

【対策】 このメッセージの直前に出力されたメッセージからエラーの原因を追求し、エラーの要因を取り除いてください。原因がタイムアウトの場合、サーバより先にクライアントがタイムアウトしないように、定義や指定値などを見直してください。

再びトランザクションを開始させるためには、dc_trn_unchained_commit_s 関数、または dc_trn_unchained_rollback_s 関数を実行したあと、dc_trn_begin_s 関数を実行してください。

なお、トランザクションの処理中に障害が発生した場合は、dc_trn_unchained_commit_s 関数、または dc_trn_unchained_rollback_s 関数を実行するように処理を変更することをお勧めします。

KFCA02480-I

The permanent connection was established. Ipaddr=aa....aa,port=bb....bb (R)

常設接続を確立しました。常設接続確立先は、CUP 実行プロセス、rap リスナー、または DCCM3 論理端末のどれかです。

aa....aa : 常設接続確立先の IP アドレス

bb...bb : 常設コネクション確立先のポート番号

KFCA02481-I

```
The permanent connection was cut off.IPaddr=aa....aa, port=bb...bb (R)
```

常設コネクションを切断しました。常設コネクション確立先は、CUP 実行プロセス、rap リスナー、または DCCM3 論理端末のどれかです。

aa...aa : 常設コネクション確立先の IP アドレス

bb...bb : 常設コネクション確立先のポート番号

KFCA02482-E

```
communication error, func=aa....aa, errno=bb....bb,port=cc....cc (R)
```

通信障害が発生しました。

aa...aa : 障害が発生したソケットインタフェースの関数名

bb...bb : エラー番号

cc....cc : 使用中ポート番号

(S)処理を中断します。

(O)TP1/Client 管理者に連絡してください。

[対策] 関数名, エラー番号を参照して, 原因を調査してください。

KFCA02485-E

```
invalid message received while aa....aa function was being executed. expected  
message=bb...bb, received message=cc....cc, c-info=dd....dd:ee....ee, s-info=ff....ff:gg....gg,  
action=hh....hh (R)
```

メッセージの組み立て機能, またはメッセージの送達確認機能を使用しているときに, 不正な値が設定されたメッセージを受信しました。

aa...aa : 実行関数名

bb...bb : 期待したメッセージ (16 進数列) (出力情報がない場合, ****を出力します)

cc....cc : 受信したメッセージ (16 進数列)

dd....dd : 自 IP アドレス (10 進ドット記法)

ee....ee : 自ポート

ff...ff : 相手 IP アドレス (10 進ドット記法)

gg...gg : 相手ポート

hh...hh : 処理

RETRY : 処理を続行します。受信処理をリトライします。

STOP : 処理を中止します。関数はエラーリターンします。

(O)TP1/Client 管理者に連絡してください。

[対策]

hh...hh の値に応じて、次のように対策してください。

RETRY の場合

メッセージ ID が不正なメッセージを受信したため、受信処理をリトライしました。そのため、正しいメッセージを受信し、関数は正常終了している可能性があります。正常終了しなかった場合、このメッセージの次に出力されるメッセージを参照してください。

STOP の場合

メッセージ長、またはセグメント情報が不正なメッセージを受信したため、処理を中止しました。相手システムの設定を見直してください。メッセージが衝突 (dc_clt_assem_send_s 関数が DCCLTER_COLLISION_MESSAGE でエラーリターン) している場合は、必要に応じてリトライしてください。

KFCA02486-E

```
receive buffer overflowed. argument length=aa....aa, received length=bb....bb, received  
message=cc....cc, c-info=dd....dd:ee....ee, s-info=ff....ff:gg....gg (R)
```

メッセージの組み立て機能、またはメッセージの送達確認機能の使用時に、CUP が用意した領域が小さいため、相手システムからのメッセージを受信できません。

aa...aa : dc_clt_assem_receive_s 関数の引数 recvleng に指定した値 (10 進数字)

bb...bb : 受信するメッセージ長 (10 進数字)

cc...cc : 受信したメッセージ (16 進数列)

dd...dd : 自 IP アドレス (10 進ドット記法)

ee...ee : 自ポート

ff...ff : 相手 IP アドレス (10 進ドット記法)

gg...gg : 相手ポート

(S)処理を中断します。dc_clt_assem_receive_s 関数は、DCCLTER_INF_TOO_BIG でエラーリターンします。相手システムとの接続は解放されます。

(O)このメッセージが出力されたエラーログを記録し、TP1/Client 管理者に連絡してください。

【対策】 受信したメッセージ情報には、関数の引数に指定したメッセージ長より長いメッセージ長が設定されています。dc_clt_assem_receive_s 関数の引数 recvleng に指定した値が適切であるかを見直してください。または、相手システムの UAP を見直してください。

付録

付録 A コード変換の仕様

ここでは、文字コード変換機能を使用した場合の、コード変換仕様について説明します。文字コード変換機能は、TP1/Client/P でだけ使用できます。

TP1/Client/P で支援するコード体系を次の表に示します。

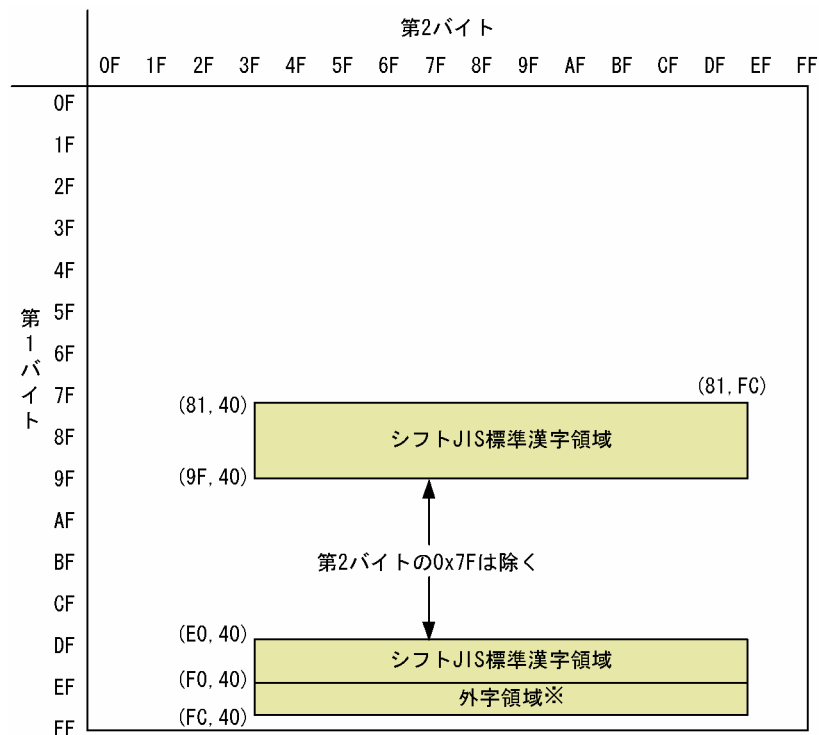
表 A-1 コード体系

文字の種類	文字コード	文字コード
半角文字	JIS コード	EBCDIK コード EBCDIC コード
全角文字	シフト JIS コード	KEIS コード

付録 A.1 TP1/Client/P でサポートするコード範囲

TP1/Client/P でサポートするコード範囲を示します。

(1) シフト JIS コードのサポート範囲



(凡例)

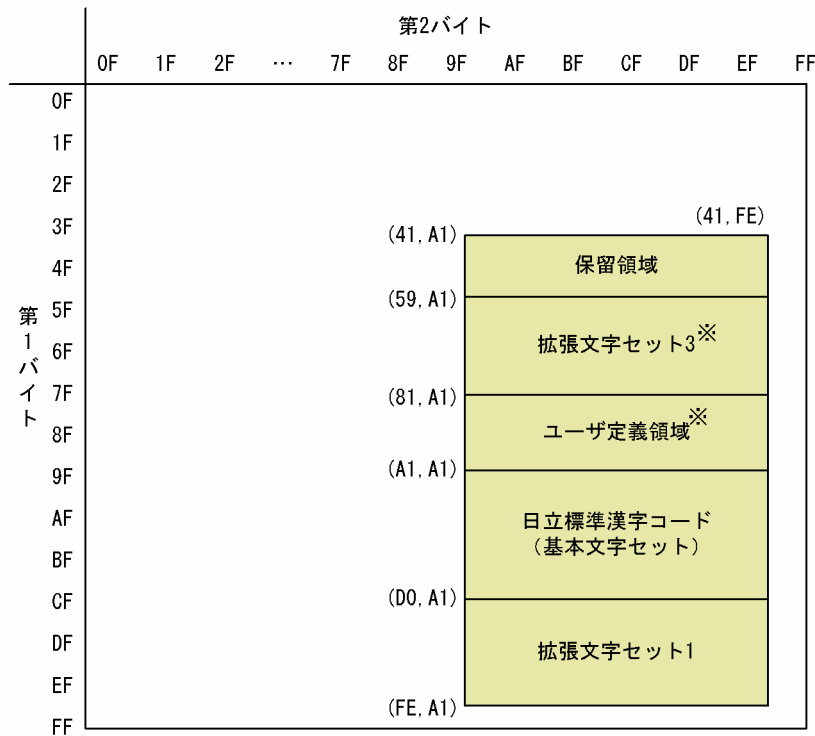
: TP1/Client/Pのサポートする範囲

注※

シフト JIS 外字コードとして変換します。外字コードの変換の詳細については、マニュアル「CommuniNet Version 3」を参照してください。

また、dc_clt_code_convert 関数の場合は、スペースに変換します。

(2) KEIS コードのサポート範囲



(凡例)

… : 記述が省略されていることを示します。

: TP1/Client/Pのサポートする範囲

注※

シフト JIS 外字コードに変換します。外字コードの変換の詳細については、マニュアル「CommuniNet Version 3」を参照してください。

また、dc_clt_code_convert 関数の場合は、スペースに変換します。

付録 A.2 シフト JIS コードと KEIS コードの変換

シフト JIS コードと KEIS コードとの変換の仕様を次の表に示します。

表 A-2 シフト JIS コード／KEIS コード変換仕様

シフト JIS コード範囲	KEIS コード範囲	変換仕様	
		シフト JIS コード→KEIS コード	KEIS コード→シフト JIS コード
第 1 バイト 81~9F E0~EF 第 2 バイト 40~7E 80~FC	第 1 バイト A1~FE 第 2 バイト A1~FE	KEIS コードの基本文字セット，拡張文字 セット 1 にマッピングします。	シフト JIS 標準漢字コードの領域にマッピ ングします。
第 1 バイト F0~FC 第 2 バイト 40~7E 80~FC	第 1 バイト 41~A0 第 2 バイト A1~FE	CommuniNet のコードマッピングユティ リティを使用し，KEIS コードの外字領域に マッピングします。 詳細については，マニュアル 「CommuniNet Version 3」を参照してく ださい。 また，dc_clt_code_convert 関数の場合は， スペースに変換します。	CommuniNet のコードマッピングユティ リティを使用し，シフト JIS コードの外字 領域にマッピングします。 詳細については，マニュアル 「CommuniNet Version 3」を参照してく ださい。 また，dc_clt_code_convert 関数の場合は， スペースに変換します。

シフト JIS コードから'83 版 KEIS コードへの変換を次の表に示します。

表 A-3 シフト JIS コードから'83 版 KEIS コードへの変換

L/H	81	82	...	9E	9F	E0	E1	...	EE	EF	F0...FC
00 ... 3F	※1										
40	A1A1	A3A1	...	DBA1	DDA1	DFA1	E1A1	...	FBA1	FDA1	※2
41	A1A2	A3A2	...	DBA2	DDA2	DFA2	E1A2	...	FBA2	FDA2	
...	
7E	A1DF	A3DF	...	DBDF	DDDF	DFDF	E1DF	...	FBDF	FDDF	
7F	※1										
80	A1E0	A3E0	...	DBE0	DDE0	DFE0	E1E0	...	FBE0	FDE0	※2
81	A1E1	A3E1	...	DBE1	DDE1	DFE1	E1E1	...	FBE1	FED1	
...	
9D	A1FD	A3FD	...	DBFD	DDFD	DFFD	E1FD	...	FBFD	FDFD	
9E	A1FE	A3FE	...	DBFE	DDFE	DFFE	E1FE	...	FBFE	FDFE	
9F	A2A1	A4A1	...	DCA1	DEA1	E0A1	E2A1	...	FBA1	FEA1	
A0	A2A2	A4A2	...	DCA2	DEA2	E0A2	E2A2	...	FBA2	FEA2	

L/H	81	82	...	9E	9F	E0	E1	...	EE	EF	F0...FC
...	※2
FB	A2FD	A4FD	...	DCFD	DEFD	E0FD	E2FD	...	FBFD	FEFD	
FC	A2FE	A4FE	...	DCFE	DEFE	E0FE	E2FE	...	FBFE	FEFE	
FD	※1										
...											

(凡例)

…：記述が省略されていることを示します。

注※1

未定義の漢字コードと判断し、dc_clt_code_convert 関数および dc_clt_code_convert_exec 関数の引数 flags の指定によって、スペースに変換、または関数がエラーリターンします。

注※2

シフト JIS 外字コードに変換します。

dc_clt_code_convert 関数の場合は、スペースに変換します。

'83 版 KEIS コードからシフト JIS コードへの変換を次の表に示します。

表 A-4 '83 版 KEIS コードからシフト JIS コードへの変換

L/H	00...40	41...A0	A1	A2	...	FD	FE	FF
00	※1	※3						
...								
A0								
A1	※2	※3	8140	819F	...	EF40	EF9F	※3
A2			8141	81A0	...	EF41	EFA0	
A3			8142	81A1	...	EF42	EFA1	
A4			8143	81A2	...	EF43	EFA2	
...			
DF			817E	81DD	...	EF7E	EFDD	
E0			8180	81DE	...	EF80	EFDE	
E1			8181	81DF	...	EF81	EFDF	
...			
FE			819E	81FC	...	EF91	EFFC	
FF	※3							

(凡例)

…：記述が省略されていることを示します。

注※1

1 バイトコード (EBCDIK コード, EBCDIC コード) の制御コードとして変換します。

注※2

シフト JIS 外字コードに変換します。

また, dc_clt_code_convert 関数の場合は, スペースに変換します。

注※3

未定義の漢字コードと判断し, dc_clt_code_convert 関数および dc_clt_code_convert_exec 関数の引数 flags の指定によって, スペースに変換, または関数がエラーリターンします。

シフト JIS コード, '78 版 KEIS コード, および'83 版 KEIS コードの対応を次の表に示します。

表 A-5 シフト JIS コード/'78 版 KEIS コード/'83 版 KEIS コードの対応 (1)

項番	文字	シフトJISコード	'78版KEISコード	'83版KEISコード
1	鯆	E9CB	B0B3	F2CD
2	鯆	88B1	F2CD	B0B3
3	鶯	E9F2	B2A9	F2F4
4	鶯	89A7	F2F4	B2A9
5	蠣	E579	B3C2	E9DA
6	蛎	8A61	E9DA	B3C2
7	攪	9D98	B3C9	D9F8
8	攪	8A68	D9F8	B3C9
9	竈	E27D	B3F6	E3DE
10	竈	8A96	E3DE	B3F6
11	灌	9FF3	B4C3	DEF5
12	灌	8AC1	DEF5	B4C3
13	諫	E67C	B4D2	EBDD
14	諫	8AD0	EBDD	B4D2
15	頸	E8F2	B7DB	F0F4
16	頸	8C7A	F0F4	B7DB
17	礪	E1E6	B9DC	E2E8
18	礪	8D7B	E2E8	B9DC
19	藥	E541	BCC9	E9A2
20	蕊	8EC7	E9A2	BCC9
21	靱	E8D5	BFD9	F0D7
22	靱	9078	F0D7	BFD9
23	賤	E6CB	C1A8	ECCD
24	賤	9147	ECCD	C1A8
25	壺	9AE2	C4DB	D4E4
26	壺	92D9	D4E4	C4DB
27	礪	E1E8	C5D7	E2EA
28	礪	9376	E2EA	C5D7
29	檣	9E8D	C5EE	DBED
30	檣	938E	DBED	C5EE
31	濤	9FB7	C5F3	DEB9
32	濤	9393	DEB9	C5F3
33	邇	E78E	C6F6	EDEE
34	迓	93F4	EDEE	C6F6
35	蠅	E5A2	C7E8	EAA4
36	蠅	9488	EAA4	C7E8
37	檜	9E77	C9B0	DBD8
38	桧	954F	DBD8	C9B0
39	儘	98D4	CBF9	D0D6
40	俛	9699	D0D6	CBF9
41	藪	E54D	CCF9	E9AE
42	藪	96F7	E9AE	CCF9
43	籠	E2C4	CFB6	E4C6
44	籠	9855	E4C6	CFB6
45	堯*	EA9F	5CC3	F4A1
46	遙*	EAA1	6BA2	F4A3
47	瑤*	EAA2	65A7	F4A4

表 A-6 シフト JIS コード/'78 版 KEIS コード/'83 版 KEIS コードの対応 (2)

項番	文字	シフトJISコード	'78版KEISコード	'83版KEISコード
48	楨※	EAA0	61FC	F4A2
49	コ	81BD	A2BE	A2BF
50	ク	81BC	A2BF	A2BE
51	カ	84A1	AFA1	A8A3
52	カ	84AC	AFA2	A8AE
			AFA3	
53	カ	84A2	AFA4	A8A4
54	カ	84AD	AFA5	A8AF
			AFA6	
55	カ	84A4	AFA7	A8A6
56	カ	84AF	AFA8	A8B1
			AFA9	
57	カ	84A3	AFAA	A8A5
58	カ	84AE	AFAB	A8B0
			AFAC	
59	カ	84A5	AFAD	A8A7
60	カ	84B0	AFAE	A8B2
			AFAF	
61	カ	84A7	AFB0	A8A9
62	カ	84B2	AFB1	A8B4
			AFB2	
63	カ	84A6	AFB3	A8A8
64	カ	84B1	AFB4	A8B3
			AFB5	
65	カ	84A8	AFB6	A8AA
66	カ	84B3	AFB7	A8B5
			AFB8	
67	カ	84A9	AFB9	A8AB
68	カ	84B4	AFBA	A8B6
			AFBB	
69	カ	849F	AFBC	A8A1
70	カ	84AA	AFBD	A8AC
			AFBE	
71	カ	84A0	AFBF	A8A2
72	カ	84AB	AFC0	A8AD
			AFC1	
			AFC2	
			AFC3	
			AFC4	

注※

シフトJISコードを'78版KEISコードに変換する場合、KEISコードの外字領域（拡張セット3の領域）に変換します。'78版KEISコードをシフトJISコードに変換する場合、スペースに変換します。ただし、外字マッピングテーブルに登録されている場合は、外字マッピングテーブルに従って変換します。外字変換の詳細については、マニュアル「CommuniNet Version 3」を参照してください。

JIS コードから EBCDIK コードへの変換表を次の二つの表に示します。

表 A-7 JIS コードから EBCDIK コードへの変換表 (1)

L/H	0	1	2	3	4	5	6	7
0	NUL (00)	DLE (10)	SP (40)	0 (F0)	@ (7C)	P (D7)	' (79)	p (76)
1	SOH (01)	DC1 (11)	! (4F)	1 (F1)	A (C1)	Q (D8)	a (59)	q (77)
2	STX (02)	DC2 (12)	" (7F)	2 (F2)	B (C2)	R (D9)	b (62)	r (78)
3	ETX (03)	DC3 (13)	# (7B)	3 (F3)	C (C3)	S (E2)	c (63)	s (80)
4	EOT (37)	DC4 (3C)	\$ (E0)	4 (F4)	D (C4)	T (E3)	d (64)	t (8B)
5	ENQ (2D)	NAK (3D)	% (6C)	5 (F5)	E (C5)	U (E4)	e (65)	u (9B)
6	ACK (2E)	SYN (32)	& (50)	6 (F6)	F (C6)	V (E5)	f (66)	v (9C)
7	BEL (2F)	ETB (26)	' (7D)	7 (F7)	G (C7)	W (E6)	g (67)	w (A0)
8	BS (16)	CAN (18)	((4D)	8 (F8)	H (C8)	X (E7)	h (68)	x (AB)
9	HT (05)	EM (19)) (5D)	9 (F9)	I (C9)	Y (E8)	i (69)	y (B0)
A	NL (15)	SUB (3F)	* (5C)	: (7A)	J (D1)	Z (E9)	j (70)	z (B1)
B	VT (0B)	ESC (27)	+ (4E)	: (5E)	K (D2)	[(4A)	k (71)	{ (C0)
C	FF (0C)	FS (1C)	. (6B)	< (4C)	L (D3)	¥ (5B)	l (72)	 (6A)
D	CR (0D)	GS (1D)	- (60)	= (7E)	M (D4)] (5A)	m (73)	} (D0)
E	SO (0E)	RS (1E)	. (4B)	> (6E)	N (D5)	^ (5F)	n (74)	(A1)
F	SI (0F)	US (1F)	/ (61)	? (6F)	0 (D6)	- (6D)	o (75)	DEL (07)

表 A-8 JIS コードから EBCDIK コードへの変換表 (2)

L/H	8	9	A	B	C	D	E	F
0	(空白) (20)	—	(空白) (57)	— (58)	々 (91)	ミ (A5)	—	—
1	—	—	。 (41)	ア (81)	チ (92)	ム (A6)	—	—
2	—	—	「 (42)	イ (82)	ツ (93)	メ (A7)	—	—
3	—	—	」 (43)	ウ (83)	テ (94)	モ (A8)	—	—
4	—	—	、 (44)	エ (84)	ト (95)	ヤ (A9)	—	—
5	—	—	・ (45)	オ (85)	ナ (96)	ユ (AA)	—	—
6	—	—	ヲ (46)	カ (86)	ニ (97)	ヨ (AC)	—	—
7	—	—	ァ (47)	キ (87)	ヌ (98)	ラ (AD)	—	—
8	—	—	ィ (48)	ク (88)	ネ (99)	リ (AE)	—	—
9	—	—	ゥ (49)	ケ (89)	ノ (9A)	ル (AF)	—	—
A	—	—	ェ (51)	コ (8A)	ハ (9D)	レ (BA)	—	—
B	—	—	ォ (52)	サ (8C)	ヒ (9E)	ロ (BB)	—	—
C	—	—	ャ (53)	シ (8D)	フ (9F)	ワ (BC)	—	—
D	—	—	ュ (54)	ス (8E)	ヘ (A2)	ン (BD)	—	(空白) (FD)
E	—	—	ョ (55)	セ (8F)	ホ (A3)	ベ (BE)	—	(空白) (FE)
F	—	—	ッ (56)	ソ (90)	マ (A4)	ブ (BF)	—	(空白) (FF)

(凡例)

— : シフトJISコードの先頭バイトと判断して変換します。

EBCDIK コードから JIS コードへの変換表を次の二つの表に示します。

表 A-9 EBCDIK コードから JIS コードへの変換表 (1)

L/H	0	1	2	3	4	5	6	7
0	NUL (00)	DEL (10)	(空白) (80)	(空白) (90)	SP (20)	& (26)	- (2D)	j (6A)
1	SOH (01)	DC1 (11)	(空白) (81)	(空白) (91)	。 (A1)	ヱ (AA)	/ (2F)	k (6B)
2	STX (02)	DC2 (12)	(空白) (82)	SYN (16)	Γ (A2)	ォ (AB)	b (62)	l (6C)
3	ETX (03)	DC3 (13)	(空白) (83)	(空白) (93)	∟ (A3)	ャ (AC)	c (63)	m (6D)
4	(空白) (9C)	(空白) (9D)	(空白) (84)	(空白) (94)	, (A4)	ユ (AD)	d (64)	n (6E)
5	HT (09)	NL (0A)	(空白) (85)	(空白) (95)	・ (A5)	ヨ (AE)	e (65)	o (6F)
6	(空白) (86)	BS (08)	ETB (17)	(空白) (96)	ヲ (A6)	ヅ (AF)	f (66)	p (70)
7	BEL (7F)	(空白) (87)	ESC (1B)	EOT (04)	ヲ (A7)	(空白) (A0)	g (67)	q (71)
8	(空白) (97)	CAN (18)	(空白) (88)	(空白) (98)	イ (A8)	- (B0)	h (68)	r (72)
9	(空白) (8D)	EM (19)	(空白) (89)	(空白) (99)	ウ (A9)	a (61)	i (69)	' (60)
A	(空白) (8E) ※	(空白) (92)	(空白) (8A)	(空白) (9A)	[(5B)] (5D)	 (7C)	: (3A)
B	VT (0B)	(空白) (8F)	(空白) (8B)	(空白) (9B)	・ (2E)	¥ (5C)	, (2C)	# (23)
C	FF (0C)	FS (1C)	(空白) (8C)	DC4 (14)	< (3C)	* (2A)	% (25)	@ (40)
D	CR (0D)	GS (1D)	ENQ (05)	NAK (15)	((28)) (29)	- (5F)	' (27)
E	SO (0E)	RS (1E)	ACK (06)	(空白) (9E)	+ (2B)	; (3B)	> (3E)	= (3D)
F	SI (0F)	US (1F)	BEL (07)	SUB (1A)	! (21)	^ (5E)	? (3F)	" (22)

注※

切り替えコード (0x0A41, 0x0A42) の先頭コードバイトと判断して変換します。

表 A-10 EBCDIK コードから JIS コードへの変換表 (2)

L/H	8	9	A	B	C	D	E	F
0	s (73)	ソ (BF)	w (77)	y (79)	{ (7B)	} (7D)	\$ (24)	0 (30)
1	ア (B1)	タ (C0)	ウ (7E)	Z (7A)	A (41)	J (4A)	(空白) (9F)	1 (31)
2	イ (B2)	チ (C1)	エ (CD)	(空白) (E0)	B (42)	K (4B)	S (53)	2 (32)
3	ウ (B3)	ツ (C2)	ホ (CE)	(空白) (E1)	C (43)	L (4C)	T (54)	3 (33)
4	エ (B4)	テ (C3)	マ (CF)	(空白) (E2)	D (44)	M (4D)	U (55)	4 (34)
5	オ (B5)	ト (C4)	ミ (D0)	(空白) (E3)	E (45)	N (4E)	V (56)	5 (35)
6	カ (B6)	ナ (C5)	ム (D1)	(空白) (E4)	F (46)	O (4F)	W (57)	6 (36)
7	キ (B7)	ニ (C6)	メ (D2)	(空白) (E5)	G (47)	P (50)	X (58)	7 (37)
8	ク (B8)	ヌ (C7)	モ (D3)	(空白) (E6)	H (48)	Q (51)	Y (59)	8 (38)
9	ケ (B9)	ネ (C8)	ヤ (D4)	(空白) (E7)	I (49)	R (52)	Z (5A)	9 (39)
A	コ (BA)	ノ (C9)	ユ (D5)	レ (DA)	(空白) (E8)	(空白) (EE)	(空白) (F4)	(空白) (FA)
B	ク (74)	ウ (75)	エ (78)	カ (DB)	(空白) (E9)	(空白) (EF)	(空白) (F5)	(空白) (FB)
C	サ (BB)	ヴ (76)	ヨ (D6)	リ (DC)	(空白) (EA)	(空白) (F0)	(空白) (F6)	(空白) (FC)
D	シ (BC)	ハ (CA)	ラ (D7)	ン (DD)	(空白) (EB)	(空白) (F1)	(空白) (F7)	(空白) (FD)
E	ス (BD)	ヒ (CB)	リ (D8)	ン (DE)	(空白) (EC)	(空白) (F2)	(空白) (F8)	(空白) (FE)
F	セ (BE)	フ (CC)	ル (D9)	ン (DF)	(空白) (ED)	(空白) (F3)	(空白) (F9)	E0 (FF)

JIS コードから EBCDIC コードへの変換表を次の二つの表に示します。

表 A-11 JIS コードから EBCDIC コードへの変換表 (1)

L/H	0	1	2	3	4	5	6	7
0	NUL (00)	DLE (10)	SP (40)	0 (F0)	@ (7C)	P (D7)	' (79)	p (97)
1	SOH (01)	DC1 (11)	! (4F)	1 (F1)	A (C1)	Q (D8)	a (81)	q (98)
2	STX (02)	DC2 (12)	" (7F)	2 (F2)	B (C2)	R (D9)	b (82)	r (99)
3	ETX (03)	DC3 (13)	# (7B)	3 (F3)	C (C3)	S (E2)	c (83)	s (A2)
4	EOT (37)	DC4 (3C)	\$ (E0)	4 (F4)	D (C4)	T (E3)	d (84)	t (A3)
5	ENQ (2D)	NAK (3D)	% (6C)	5 (F5)	E (C5)	U (E4)	e (85)	u (A4)
6	ACK (2E)	SYN (32)	& (50)	6 (F6)	F (C6)	V (E5)	f (86)	v (A5)
7	BEL (2F)	ETB (26)	' (7D)	7 (F7)	G (C7)	W (E6)	g (87)	w (A6)
8	BS (16)	CAN (18)	((4D)	8 (F8)	H (C8)	X (E7)	h (88)	x (A7)
9	HT (05)	EM (19)) (5D)	9 (F9)	I (C9)	Y (E8)	i (89)	y (A8)
A	NL (15)	SUB (3F)	* (5C)	: (7A)	J (D1)	Z (E9)	j (91)	z (A9)
B	VT (0B)	ESC (27)	+ (4E)	; (5E)	K (D2)	[(4A)	k (92)	{ (C0)
C	FF (0C)	FS (1C)	. (6B)	< (4C)	L (D3)	¥ (5B)	l (93)	 (6A)
D	CR (0D)	GS (1D)	- (60)	= (7E)	M (D4)] (5A)	m (94)	} (D0)
E	SO (0E)	RS (1E)	. (4B)	> (6E)	N (D5)	^ (5F)	n (95)	(A1)
F	SI (0F)	US (1F)	/ (61)	? (6F)	O (D6)	_ (6D)	o (96)	DEL (07)

表 A-12 JIS コードから EBCDIC コードへの変換表 (2)

L/H	8	9	A	B	C	D	E	F
0	(20)	—	(57)	(40)	(40)	(40)	—	—
1	—	—	(40)	(40)	(40)	(40)	—	—
2	—	—	(40)	(40)	(40)	(40)	—	—
3	—	—	(40)	(40)	(40)	(40)	—	—
4	—	—	(40)	(40)	(40)	(40)	—	—
5	—	—	(40)	(40)	(40)	(40)	—	—
6	—	—	(40)	(40)	(40)	(40)	—	—
7	—	—	(40)	(40)	(40)	(40)	—	—
8	—	—	(40)	(40)	(40)	(40)	—	—
9	—	—	(40)	(40)	(40)	(40)	—	—
A	—	—	(40)	(40)	(40)	(40)	—	—
B	—	—	(40)	(40)	(40)	(40)	—	—
C	—	—	(40)	(40)	(40)	(40)	—	—
D	—	—	(40)	(40)	(40)	(40)	—	(FD)
E	—	—	(40)	(40)	(40)	(40)	—	(FE)
F	—	—	(40)	(40)	(40)	(40)	—	(FF)

(凡例)

— : シフトJISコードの先頭バイトと判断して変換します。

EBCDIC コードから JIS コードへの変換表を次の二つの表に示します。

表 A-13 EBCDIC コードから JIS コードへの変換表 (1)

L/H	0	1	2	3	4	5	6	7
0	NUL (00)	DEL (10)	(空白) (80)	(空白) (90)	SP (20)	& (26)	- (2D)	(空白) (BA)
1	SOH (01)	DC1 (11)	(空白) (81)	(空白) (91)	' (A0)	(空白) (A9)	/ (2F)	(空白) (BB)
2	STX (02)	DC2 (12)	(空白) (82)	SYN (16)	(空白) (A1)	(空白) (AA)	(空白) (B2)	(空白) (BC)
3	ETX (03)	DC3 (13)	(空白) (83)	(空白) (93)	(空白) (A2)	(空白) (AB)	(空白) (B3)	(空白) (BD)
4	(空白) (9C)	(空白) (9D)	(空白) (84)	(空白) (94)	(空白) (A3)	(空白) (AC)	(空白) (B4)	(空白) (BE)
5	HT (09)	NL (0A)	(空白) (85)	(空白) (95)	(空白) (A4)	(空白) (AD)	(空白) (B5)	(空白) (BF)
6	(空白) (86)	BS (08)	ETB (17)	(空白) (96)	(空白) (A5)	(空白) (AE)	(空白) (B6)	(空白) (C0)
7	DEL (7F)	(空白) (87)	ESC (1B)	EOT (04)	(空白) (A6)	(空白) (AF)	(空白) (B7)	(空白) (C1)
8	(空白) (97)	CAN (18)	(空白) (88)	(空白) (98)	(空白) (A7)	(空白) (B0)	(空白) (B8)	(空白) (C2)
9	(空白) (8D)	EM (19)	(空白) (89)	(空白) (99)	(空白) (A8)	(空白) (B1)	(空白) (B9)	' (60)
A	(空白) (8E) ※	(空白) (92)	(空白) (8A)	(空白) (9A)	[(5B)] (5D)	 (7C)	: (3A)
B	VT (0B)	(空白) (8F)	(空白) (8B)	(空白) (9B)	. (2E)	¥ (5C)	. (2C)	# (23)
C	FF (0C)	FS (1C)	(空白) (8C)	DC4 (14)	< (3C)	* (2A)	% (25)	@ (40)
D	CR (0D)	GS (1D)	ENQ (05)	NAK (15)	((28)) (29)	- (5F)	' (27)
E	SO (0E)	RS (1E)	ACK (06)	(空白) (9E)	+ (2B)	; (3B)	> (3E)	= (3D)
F	SI (0F)	US (1F)	BEL (07)	SUB (1A)	! (21)	^ (5E)	? (3F)	" (22)

注※

切り替えコード (0x0A41, 0x0A42) の先頭コードバイトと判断して変換します。

表 A-14 EBCDIC コードから JIS コードへの変換表 (2)

L/H	8	9	A	B	C	D	E	F
0	(空白) (C3)	(空白) (CA)	(空白) (D1)	(空白) (D8)	{ (7B)	} (7D)	\$ (24)	0 (30)
1	a (61)	j (6A)	(空白) (7E)	(空白) (D9)	A (41)	J (4A)	(空白) (20)	1 (31)
2	b (62)	k (6B)	s (73)	(空白) (DA)	B (42)	K (4B)	S (53)	2 (32)
3	c (63)	l (6C)	t (74)	(空白) (DB)	C (43)	L (4C)	T (54)	3 (33)
4	d (64)	m (6D)	u (75)	(空白) (DC)	D (44)	M (4D)	U (55)	4 (34)
5	e (65)	n (6E)	v (76)	(空白) (DD)	E (45)	N (4E)	V (56)	5 (35)
6	f (66)	o (6F)	w (77)	(空白) (DE)	F (46)	O (4F)	W (57)	6 (36)
7	g (67)	p (70)	x (78)	(空白) (DF)	G (47)	P (50)	X (58)	7 (37)
8	h (68)	q (71)	y (79)	(空白) (E0)	H (48)	Q (51)	Y (59)	8 (38)
9	i (69)	r (72)	z (7A)	(空白) (E1)	I (49)	R (52)	Z (5A)	9 (39)
A	(空白) (C4)	(空白) (CB)	(空白) (D2)	(空白) (E2)	(空白) (E8)	(空白) (EE)	(空白) (F4)	(空白) (FA)
B	(空白) (C5)	(空白) (CC)	(空白) (D3)	(空白) (E3)	(空白) (E9)	(空白) (EF)	(空白) (F5)	(空白) (FB)
C	(空白) (C6)	(空白) (CD)	(空白) (D4)	(空白) (E4)	(空白) (EA)	(空白) (F0)	(空白) (F6)	(空白) (FC)
D	(空白) (C7)	(空白) (CE)	(空白) (D5)	(空白) (E5)	(空白) (EB)	(空白) (F1)	(空白) (F7)	(空白) (FD)
E	(空白) (C8)	(空白) (CF)	(空白) (D6)	(空白) (E6)	(空白) (EC)	(空白) (F2)	(空白) (F8)	(空白) (FE)
F	(空白) (C9)	(空白) (D0)	(空白) (D7)	(空白) (E7)	(空白) (ED)	(空白) (F3)	(空白) (F9)	E0 (FF)

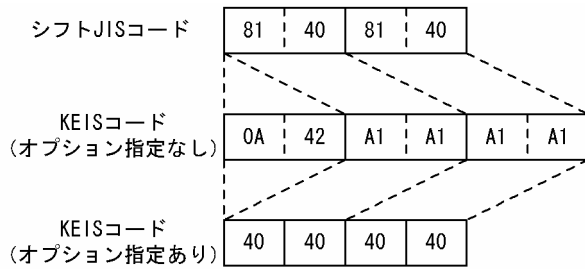
付録 A.3 コード変換例

dc_clt_code_convert 関数または dc_clt_codeconv_exec 関数の引数 flags の指定によって、変換結果が異なる文字を引数 flags の指定ごとに次に示します。

(1) DCCLT_CNV_SPCHAN を指定した場合

dc_clt_code_convert 関数または dc_clt_codeconv_exec 関数の引数 flags に DCCLT_CNV_SPCHAN を指定した場合、全角スペースを半角スペース 2 個に変換します。DCCLT_CNV_SPCHAN を指定した場合の文字コードの対応を次の図に示します。

図 A-1 文字コードの対応 (DCCLT_CNV_SPCHAN を指定した場合)



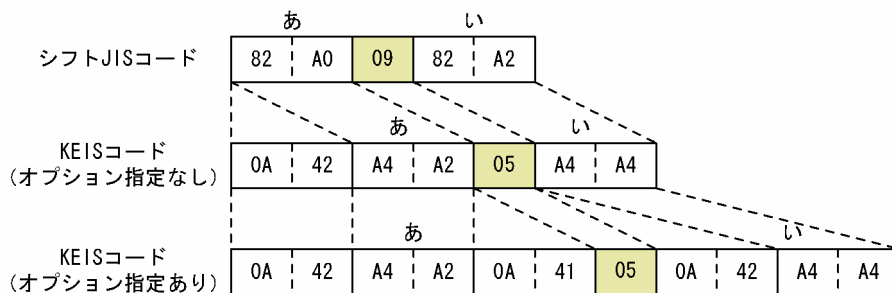
コードの説明

- 8140 : シフトJISの全角スペース
- 0A42 : 全角開始コード
- A1A1 : KEISコードの全角スペース
- 40 : EBCDIKコード/EBCDICコードの半角スペース

(2) DCCLT_CNV_TAB を指定した場合

dc_clt_code_convert 関数または dc_clt_codeconv_exec 関数の引数 flags に DCCLT_CNV_TAB を指定した場合、タブコードを半角コードに変換します。タブコードの直前または直後のデータが全角コードのときは、シフトコードが付きます。DCCLT_CNV_TAB を指定した場合の文字コードの対応を次の図に示します。

図 A-2 文字コードの対応 (DCCLT_CNV_TAB を指定した場合)



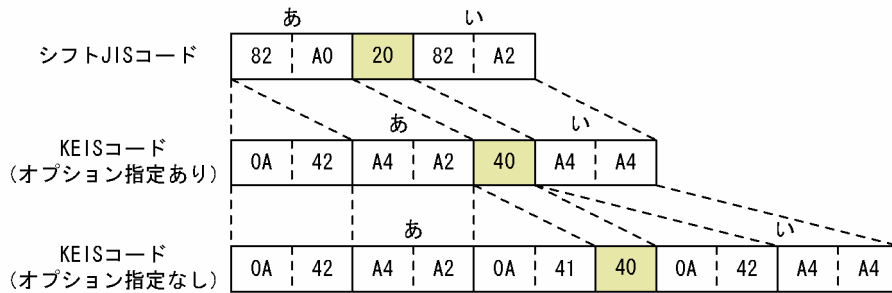
コードの説明

- 09 : JISコードのタブコード
- 0A42 : 全角開始コード
- 05 : EBCDIKコード/EBCDICコードのタブコード
- 0A41 : 半角開始コード

(3) DCCLT_CNV_CNTL を指定した場合

dc_clt_code_convert 関数または dc_clt_codeconv_exec 関数の引数 flags に DCCLT_CNV_CNTL を指定した場合、制御コードを半角コードに変換します。制御コードの直前または直後のデータが全角コードのときは、シフトコードが付きます。DCCLT_CNV_CNTL を指定した場合の文字コードの対応を次の図に示します。

図 A-3 文字コードの対応 (DCCLT_CNV_CNTRL を指定した場合)



コードの説明

- 20 : JISコードの半角スペース
- 0A42 : 全角開始コード
- 40 : EBCDIKコード/EBCDICコードの半角スペース
- 0A41 : 半角開始コード

付録 A.4 コード変換をするときの注意事項

- 入力データの先頭が KEIS コードで始まる場合は、全角開始コード (0x0A42) を付けてください。
- 外字マッピングテーブルが存在しない場合は、外字コードはスペースに変換します。外字マッピングテーブルの詳細については、マニュアル「CommuniNet Version 3」を参照してください。
- EBCDIK コード、EBCDIC コード、または KEIS コードから、JIS コードまたはシフト JIS コードへの変換中に、変換表に存在しないコードを検出した場合は、次のように変換します。
 - dc_clt_code_convert 関数または dc_clt_codeconv_exec 関数の引数 flags に DCCLT_CNV_INVSPC を指定した場合、0x0A 直後の 1 バイトが 0x41 または 0x42 以外ときは、0x0A を該当するコードに変換します。
 - 全角モードでの変換中に 0x00~0x40 の半角コードを検出した場合、該当する 1 バイトコードに変換します。

付録 B UNIX の CUP 実行プロセスでユーザが同時に使用できるファイル記述子の最大数

CUP 実行プロセスでユーザが同時に使用できるファイル記述子の最大数（ソケット用のファイル記述子も含む）は、次の計算式で求められます。

次の計算式を超えるファイル記述子は同時に使用できません。使用した場合、CUP が異常終了するおそれがあります。

$$A \leq B - (C \times 10^{※1})$$

A：CUP 実行プロセスでユーザが同時に使用可能なファイル記述子の最大数

B：CUP 実行プロセスで使用可能なファイル記述子の最大数

適用 OS が Linux の場合（07-51 以前）：1024

適用 OS が Linux の場合（07-52 以降）：カーネル設定値^{※2}

適用 OS が AIX の場合：65534

C：TP1/Client の関数を発行するスレッドの同時実行数

注※1 TP1/Client/W が同時に使用するファイル記述子の最大数です。

注※2 カーネル設定値の詳細については、OS のマニュアルを参照してください。

付録 C バージョンアップ時の変更点

各バージョンでの変更点を次に示す分類ごとに示します。

- 関数，定義およびコマンドの追加と削除
- 動作の変更
- 関数，定義およびコマンドのデフォルト値の変更

付録 C.1 07-02 での変更点

TP1/Client/W 07-02，TP1/Client/P 07-02 での関数，定義およびコマンドの追加と削除を次の表に示します。

表 C-1 TP1/Client/W 07-02，TP1/Client/P 07-02 での関数，定義およびコマンドの追加と削除

種別	分類	内容
追加	関数	dc_clt_assem_send_s
		dc_clt_assem_receive_s
		CBLDCCLS('STCONINF')
		CBLDCCLS('ASMSEND')
		CBLDCCLS('ASMRECV')
	定義	クライアント環境定義 DCCLTDELIVERYCHECK
		クライアント環境定義 DCCLTPRFINFOSEND
		クライアント環境定義 DCCLTCUPSNDHOST
	コマンド	なし
	削除	なし

TP1/Client/W 07-02，TP1/Client/P 07-02 での動作の変更点を次の表に示します。

表 C-2 TP1/Client/W 07-02，TP1/Client/P 07-02 での動作の変更点

分類	内容
関数	なし
コマンド	なし
その他	メッセージの組み立て機能を追加
	メッセージの送達確認機能を追加

分類	内容
その他	CUP の送信元ホストを指定できる機能を追加

TP1/Client/W 07-02, TP1/Client/P 07-02 でのデフォルト値の変更点を次の表に示します。

表 C-3 TP1/Client/W 07-02, TP1/Client/P 07-02 でのデフォルト値の変更点

分類	内容
関数	なし
定義	クライアント環境定義 DCSCDMULTICOUNT のデフォルト値を《1》に、指定範囲を((1~4096))に変更
コマンド	なし

付録 C.2 07-01 での変更点

TP1/Client/W 07-01, TP1/Client/P 07-01 での関数, 定義およびコマンドの追加と削除を次の表に示します。

表 C-4 TP1/Client/W 07-01, TP1/Client/P 07-01 での関数, 定義およびコマンドの追加と削除

種別	分類	内容
追加	関数	なし
	定義	クライアント環境定義 DCCLTRECVBUFFSIZE
		クライアント環境定義 DCCLTSENDBUFFSIZE
		クライアント環境定義 DCCLTTCPNODELAY
コマンド	なし	
削除	なし	

TP1/Client/W 07-01, TP1/Client/P 07-01 での動作の変更はありません。

TP1/Client/W 07-01, TP1/Client/P 07-01 での関数, 定義およびコマンドのデフォルト値の変更はありません。

付録 C.3 07-00 での変更点

TP1/Client/W 07-00, TP1/Client/P 07-00 での関数, 定義およびコマンドの追加と削除を次の表に示します。

表 C-5 TP1/Client/W 07-00, TP1/Client/P 07-00 での関数, 定義およびコマンドの追加と削除

種別	分類	内容
追加	関数	CBLDCCLS('EXCLTIN ')
		CBLDCCLS('EXSEND ')
		CBLDCCLS('EXNACPT ')
		CBLDCCLS('EXNCANCL')
		CBLDCCLS('EXNCACPT')
	定義	クライアント環境定義 DCCLTRPCMAXMSGSIZE
	コマンド	なし
削除	なし	

TP1/Client/W 07-00, TP1/Client/P 07-00 での動作の変更点を次の表に示します。

表 C-6 TP1/Client/W 07-00, TP1/Client/P 07-00 での動作の変更点

分類	内容
関数	<p>次に示す関数の引数に指定可能, または返却可能なホスト名長 (63 文字) を, クライアント環境定義 DCCLTOPTION の指定によって 255 文字に拡大</p> <ul style="list-style-type: none"> • dc_clt_cltin_s • DCRPC_DIRECT_SCHEDULE • dc_clt_set_raphost_s • dc_clt_get_raphost_s • dc_clt_send_s • dc_clt_accept_notification_s • dc_clt_cancel_notification_s • dc_clt_chained_accept_notification_s • CBLDCCLS('STRAPHST') • CBLDCCLS('GTRAPHST') <p>次に示す関数の引数に指定可能, または返却可能なユーザデータ長 (1 メガバイト) をクライアント環境定義 DCCLTRPCMAXMSGSIZE に指定した値に拡大</p> <ul style="list-style-type: none"> • dc_rpc_call_s • dc_rpc_call_to_s • dc_clt_accept_notification_s • dc_clt_cancel_notification_s • dc_clt_chained_accept_notification_s • CBLDCRPS('CALL ') • CBLDCCLS('NOTIFY ') • CBLDCCLS('EXNACPT ') • CBLDCCLS('CANCEL ')

分類	内容
関数	<ul style="list-style-type: none"> • CBLDCCLS('EXNCANCL') • CBLDCCLS('A-NOTIFY') • CBLDCCLS('EXNCACPT')
	<p>次に示す関数で、引数 hostname に NULL を指定すると、ホスト名を格納しないように変更</p> <ul style="list-style-type: none"> • dc_clt_accept_notification_s • dc_clt_chained_accept_notification_s
定義	クライアント環境定義 DCCLTOPTION に、ホスト名長を拡張する指定 (00000008) を追加
	<p>次に示すクライアント環境定義に指定可能なホスト名長 (63 文字) を、クライアント環境定義 DCCLTOPTION の指定によって 255 文字に拡大</p> <ul style="list-style-type: none"> • DCHOST • DCCLTRAPHOST • DCCLTDCCMHOST • DCSNDHOST • DCCLTSERVICEGROUPLIST
コマンド	なし
その他	ホスト名拡張機能を追加

TP1/Client/W 07-00, TP1/Client/P 07-00 での関数、定義およびコマンドのデフォルト値の変更はありません。

索引

C

CBLDCCLS('ASMRECV ') 352
CBLDCCLS('ASMSEND ') 348
CBLDCCLS('A-NOTIFY') 375
CBLDCCLS('CANCEL ') 364
CBLDCCLS('CLTIN ') 284
CBLDCCLS('CLTOUT ') 292
CBLDCCLS('CONNECT ') 309
CBLDCCLS('C-NOTIFY') 374
CBLDCCLS('DISCNCT ') 312
CBLDCCLS('EXCLTIN ') 288
CBLDCCLS('EXNACPT ') 360
CBLDCCLS('EXNCACPT') 378
CBLDCCLS('EXNCANCL') 367
CBLDCCLS('EXSEND ') 339
CBLDCCLS('GETTRNID') 334
CBLDCCLS('GTRAPHST') 316
CBLDCCLS('NOTIFY ') 356
CBLDCCLS('O-NOTIFY') 371
CBLDCCLS('RECEIVE2') 345
CBLDCCLS('RECEIVE ') 342
CBLDCCLS('SEND ') 336
CBLDCCLS('STCONINF') 318
CBLDCCLS('STRAPHST') 314
CBLDCRPS('CALL ') 298
CBLDCRPS('CLOSE ') 296
CBLDCRPS('GETWATCH') 306
CBLDCRPS('OPEN ') 294
CBLDCRPS('SETWATCH') 305
CBLDCTRS('BEGIN ') 321
CBLDCTRS('C-COMMIT') 323
CBLDCTRS('C-ROLL ') 326
CBLDCTRS('INFO ') 332
CBLDCTRS('U-COMMIT') 328
CBLDCTRS('U-ROLL ') 330
CBLDCUTL('CNVCLS ') 386
CBLDCUTL('CNVEXEC ') 388

CBLDCUTL('CNVOPN ') 384
CBLDCUTL('CODECNV ') 381
cltdump 436
COBOL 言語用テンプレート 269
CUP 35
CUP 実行プロセス 42
CUP と SPP の作成 143, 271
CUP のコーディング例 143
CUP の受信で使用するポート番号 417
CUP の送信元ホスト 417
CUP を COBOL 言語で作成した場合のコーディング例 271
C 言語の関数の引数に指定できるホスト名長およびホスト名格納領域長 126

D

dc_clt_accept_notification_s 217
dc_clt_assem_receive_s 213
dc_clt_assem_send_s 209
dc_clt_cancel_notification_s 221
dc_clt_chained_accept_notification_s 229
dc_clt_close_notification_s 228
dc_clt_cltin_s 152
dc_clt_cltout_s 157
dc_clt_code_convert 247
dc_clt_codeconv_close 253
dc_clt_codeconv_exec 254
dc_clt_codeconv_open 251
dc_clt_connect_s 177
dc_clt_disconnect_s 179
dc_clt_get_raphost_s 183
dc_clt_get_trnid_s 198
dc_clt_open_notification_s 225
dc_clt_receive_s 204
dc_clt_receive2_s 207
dc_clt_send_s 202
dc_clt_set_connect_inf_s 185

dc_clt_set_raphost_s 181
 dc_rpc_call_s 160
 dc_rpc_call_to_s 167
 dc_rpc_close_s 159
 dc_rpc_get_watch_time_s 174
 dc_rpc_open_s 158
 dc_rpc_set_watch_time_s 173
 dc_trn_begin_s 188
 dc_trn_chained_commit_s 190
 dc_trn_chained_rollback_s 192
 dc_trn_info_s 200
 dc_trn_unchained_commit_s 194
 dc_trn_unchained_rollback_s 196
 DCCACHE 411
 DCCLTAUTHENT 424
 DCCLTBACKLOGCOUNT 428
 DCCLTCACHETIM 412
 DCCLTCONNECTINF 424
 DCCLTCONNECTRETRY 413
 DCCLTCONNECTTIMEOUT 408
 DCCLTCUPRCVPORT 417
 DCCLTCUPSNDHOST 417
 DCCLTDATACOMP 414
 DCCLTDCCMHOST 415
 DCCLTDCCMPORT 416
 DCCLTDELAY 416
 DCCLTDELIVERYCHECK 411
 DCCLTINQUIRETIME 415
 DCCLTLOADBALANCE 412
 DCCLTNAMEXTEND 427
 DCCLTNOSERVER 426
 DCCLTONLYTHISNODE 426
 DCCLTOPTION 427
 DCCLTPORT 415
 DCCLTPRFINFOSEND 430
 DCCLTRAPAUTOCONNECT 419
 DCCLTRAPHOST 417
 DCCLTRECVBUFFSIZE 431
 DCCLTRPCMAXMSGSIZE 430
 DCCLTSENDBUFSIZE 431
 DCCLTSERVICEGROUPLIST 412
 DCCLTTCPNODELAY 432
 DCCLTTRCPUTM 409
 DCCLTTREXPSP 409
 DCCLTTREXPTM 408
 DCCLTTRLIMITTIME 421
 DCCLTTROPTIITEM 420
 DCCLTTRRBINFO 421
 DCCLTTRRBRCV 422
 DCCLTTRRECOVERYTYPE 423
 DCCLTTRSTATISITEM 419
 DCCLTTRWATCHTIME 421
 DCCLTTRWATTM 409
 DCCLTUTTRCMT 409
 DCCLTXATMI 416
 DCCM3 論理端末に端末識別情報を通知する 44
 DCCM3 論理端末のポート番号 416
 DCCM3 論理端末のホスト名 415
 DCCM3 論理端末へ RPC を行う場合の負荷分散 54
 DCEXTENDFUNCTION 414
 DCHOST 407
 DHOSTCHANGE 426
 DHOSTSELECT 425
 DCMAXDNSNAME 126
 DCNAMPORT 407
 DCRCVPORT 410
 DCRPC_BINDING_TBL 構造体の作成 175
 DCRPC_DIRECT_SCHEDULE 175
 DCSCDDIRECT 413
 DCSCDLOADPRIORITY 426
 DCSCDMULTI 425
 DCSCDMULTICOUNT 425
 DCSCDPORT 414
 dcselint 432
 DCSNDHOST 410
 DCSNDPORT 410
 DC SOCKOPENATRCV 410
 DCSYSWATCHTIM 424

DCTRCERR 428
DTRCMDL 429
DTRCPATH 428
DTRCSOC 429
DTRCSOCSIZE 429
DTRCUAP 429
DCUTOKEY 411
dcvclt.h 126
DCWATCHTIM 408
DCWATCHTIMINHERIT 416
DCWATCHTIMRPCINHERIT 424

E

EBCDIC コードから JIS コードへの変換表 (1) 504
EBCDIC コードから JIS コードへの変換表 (2) 505
EBCDIK コードから JIS コードへの変換表 (1) 501
EBCDIK コードから JIS コードへの変換表 (2) 502

J

JIS コードから EBCDIC コードへの変換表 (1) 503
JIS コードから EBCDIC コードへの変換表 (2) 503
JIS コードから EBCDIK コードへの変換表 (1) 499
JIS コードから EBCDIK コードへの変換表 (2) 500

M

MAXHOSTNAME 126

O

OpenTP1 以外のサーバへの RPC 52
OpenTP1 制御の最大応答待ち時間 424

R

rap リスナー, または DCCM3 の論理端末 417
RPC 46
RPC サービスの機能拡張レベル 414
RPC 送受信電文の最大長 430

S

SPP 35

SPP のコーディング例 (サービス関数 DAM アクセス) 146

SPP のコーディング例 (メイン関数 DAM アクセス) 145

T

TCP/IP 通信機能 81, 202, 336
TCP/IP の受信バッファサイズ 431
TCP/IP の送信バッファサイズ 431
TP1/Client 35
TP1/Client/P 35
TP1/Client/P の注意事項 433
TP1/Client/W 35
TP1/Client/W の注意事項 432
tpalloc 232
tpconnect 234
tpdiscon 237
tpfree 233
tprecv 242
tpsend 239

U

UAP トレース 121
UAP トレース機能 121
UAP トレースのファイルサイズ 429
UAP の開始 158, 294
UAP の終了 159, 296
UNIX 環境の場合の翻訳と結合 138, 265

W

Windows 環境の場合の翻訳と結合 139, 266
Windows の C ランタイムライブラリのインストール 393

X

X_OCTET 113
XATMI インタフェース機能 107, 232

い

一時的に格納したサービス情報の有効時間 412

一方通知受信 229, 375
一方通知受信 (ホスト名長の拡張時) 378
一方通知受信の開始 225, 371
一方通知受信の終了 228, 374
一方通知待ち状態のキャンセル 221, 364
一方通知待ち状態のキャンセル (ホスト名長の拡張時) 367
一方通知メッセージの受信 217, 356
一方通知メッセージの受信 (ホスト名長の拡張時) 360
イベントの受信 112

う

運用コマンド 434

え

エラーログ 121
エラーログ機能 121
エラーログのサイズ 428
遠隔サービスの要求 160, 298

お

オリジネータ 108
オンラインテスト機能 120

か

会話型サービス 108
会話型サービスからのメッセージの受信 242
会話型サービスで使用できる通信データの型 113
会話型サービスとの接続の確立 234
会話型サービスとの接続の切断 237
会話型サービスの時間監視 111
会話型サービスの通信 107
会話型サービスへのメッセージの送信 239
型付きバッファの解放 233
型付きバッファの割り当て 232
環境設定 392
関数インタフェース 133
関数の一覧 133
関数の記述形式 136

<

組み立てメッセージの受信 213, 352
組み立てメッセージの送信 209, 348
クライアント拡張サービス 42
クライアント拡張サービスのポート番号 415
クライアント側の障害 468
クライアント環境定義 401
クライアント環境定義の一覧 395
クライアント環境定義のオペランドで指定できる文字数 128
クライアントの機能拡張オプション 427
クライアントユーザの認証解除 157, 292
クライアントユーザの認証要求 152, 284
クライアントユーザの認証要求 (ホスト名長の拡張時) 288

け

現在のトランザクションに関する識別子の取得 78, 198, 334
現在のトランザクションに関する情報の報告 78, 200, 332

こ

コネクション確立最大監視時間 408
コネクション確立の再試行回数 413
コネクション確立要求を格納するキューの数 428
コネクションの確立 108
コネクションの強制切断 110
コネクションの切断 109
コミット 71

さ

サーバ UAP から戻ってくる値 300
サーバ UAP に渡す値 299
サーバからの一方通知受信機能 103, 217, 356
サービス応答待ち時間の更新 173, 305
サービス応答待ち時間の参照 174, 306
サービスグループと RPC 受け付け窓口の対応を定義したファイル名 412
サービス情報を一時的に格納する領域数 411

最大応答待ち時間 408
最大応答待ち時間のタイムアウト 111
最大通信遅延時間 416
サブオーディネータ 108

し

時間監視 48, 52
指定する項目の説明で使う記号 264
シフト JIS コード/'78 版 KEIS コード/'83 版 KEIS
コードの対応 (1) 497
シフト JIS コード/'78 版 KEIS コード/'83 版 KEIS
コードの対応 (2) 498
受信チェック間隔時間 432
受信で使用する CUP のポート番号 410
受信ポート固定機能 130
障害対策 466
障害発生時のトランザクションの同期点を検証する
方法 79
常設コネクション 42, 177, 309
常設コネクション確立要求先の指定 181, 314
常設コネクション確立要求先の取得 183, 316
常設コネクション問い合わせ間隔最大時間 415
常設コネクション問い合わせ間隔最大時間のタイムア
ウト 112
常設コネクションの解放 179, 312
常設コネクションの確立 177, 309
常設コネクションの確立・解放 42

す

スケジュール機能 48
スケジュールサービスのポート番号 414

せ

性能検証用トレース 123
接続先のノード名 410
接続先のポート番号 410

そ

送信元ホスト指定機能 129
ソケット受信型サーバ 48

ソケットトレース機能 122
ソケットトレースのデータサイズ 429
ソケットトレースのファイルサイズ 429

た

端末固定割り当て機能 44
端末識別情報 44, 424
端末識別情報の設定 185, 318
端末識別情報の通知 44

つ

通信先を指定した遠隔サービスの要求 167
通信障害 467

て

定義の概要 395
定義の規則 399
データ圧縮機能 64
データの受信 109
データの送信 109
テストユーザ ID 411

と

同期応答型 RPC 46
同期応答型 RPC タイムアウト時のサーバ負荷軽減 68
同期応答型 RPC と同期点の関係 75
同期点取得 71
統計情報項目 419
トラブルシュート機能 121
トランザクション最適化項目 420
トランザクション制御 70, 188, 321
トランザクション問い合わせ間隔最大時間 409
トランザクション同期点処理時の最大通信待ち時間
421
トランザクションの開始 188, 321
トランザクションの生成 110
トランザクションブランチ CPU 監視時間 409
トランザクションブランチ限界経過時間 408
トランザクションブランチ限界経過時間のタイムア
ウト 112

トランザクションブランチ最大実行可能時間 421

トレースの編集出力 436

トレースファイル作成ディレクトリ 428

ね

ネームサービスのポート番号 407

ネームサービスを使用した RPC 55

の

ノード間負荷バランス機能 49

ひ

非応答型 RPC 47

非応答型 RPC と同期点の関係 76

引数として指定する記号の説明 137

引数に指定する値の説明で使う記号 136

引数の参照 162

引数の設定 161

ヒューリスティック発生時の処置 74

非連鎖モードのコミット 72, 194, 328

非連鎖モードのロールバック 74, 196, 330

ふ

ファイアウォール 65

ファイル記述子の最大数 508

ほ

ホスト名拡張機能 126

ホスト名拡張機能使用時の COBOL-UAP 作成用プログラム 127

翻訳と結合 138, 265

ま

マスタスケジューラデーモン 56

窓口となる TP1/Server 36, 407

窓口となる TP1/Server の切り替え機能 59

窓口となる TP1/Server を優先して負荷分散する
63, 426

窓口となる TP1/Server をランダムに選択する 62,
425

マルチスケジューラ機能 56

マルチスケジューラ機能を使用した RPC 56

マルチスケジューラデーモン 56

マルチスケジューラデーモンのプロセス数 425

マルチスレッド機能 117

マルチスレッド対応のユーザアプリケーションプログラムの作成 148, 275

マルチスレッドに対応した CUP の処理の概要 117

マルチスレッドに対応しない関数の実行 118

め

メッセージ 470

メッセージ受信時の注意事項 101

メッセージ送信時の注意事項 101

メッセージの一方受信 82

メッセージの一方送信 81

メッセージの組み立て機能と送達確認機能 87

メッセージの受信 204, 342

メッセージの受信 (障害時メッセージ受信) 207, 345

メッセージの送受信 85

メッセージの送信 202, 336

メッセージの送信 (ホスト名長の拡張時) 339

も

文字コード変換 247, 381

文字コード変換機能 114

文字コード変換機能 (コードマッピングテーブルを使用しない場合) 247, 381

文字コード変換機能 (コードマッピングテーブルを使用する場合) 251, 384

文字コード変換の開始 251, 384

文字コード変換の実行 254, 388

文字コード変換の終了 253, 386

モジュールトレース機能 122

モジュールトレースのファイルサイズ 429

ゆ

ユーザアプリケーションプログラムの作成例 143, 271

ユーザ認証機能 40, 152, 284

ユースケースごとの設定方法とポートの割り当て 94

り

リモート API 機能 65

リモートプロシジャコール 158, 294

れ

連鎖 RPC 47

連鎖モードのコミット 72, 190, 323

連鎖モードのロールバック 74, 192, 326

ろ

ロールバック 73