HITACHI
Inspire the Next

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/J 編

解説・手引・文法・操作書

3000-3-D59-42

前書き

■ 対象製品

P-2464-73B4 uCosminexus TP1/Client/J 07-53

P-2464-73C4 uCosminexus TP1/Client/J 07-52

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

■ 発行

2022 年 4 月　3000-3-D59-42

■ 著作権

変更内容

変更内容（3000-3-D59-42） uCosminexus TP1/Client/J 07-53，uCosminexus TP1/Client/J 07-52，uCosminexus TP1/Client/J 07-50

追加・変更内容	変更個所
マニュアル訂正の内容を反映した。	－
ノード間負荷バランス機能の説明を変更した。	2.2.5
トレースファイルの出力ファイル名を変更した。	2.11.1，2.11.2，2.11.3，2.11.4，2.11.5
トレースファイルが取得するメッセージを変更した。	2.11.4
次の例外の説明を変更した。 • ErrTimedOutException	4.　TP1/Client/J で使用するクラス クラス TP1Client クラス ErrTimedOutException
次の環境定義の説明を追加した。 • dcnotifyreshost • dcresponsehost	5.1.1，5.2.1，5.2.2

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-D59-41） uCosminexus TP1/Client/J 07-50

追加・変更内容
マニュアル訂正の内容を反映した。
TP1/Client/J が同時に確立できる常設コネクションの説明を変更した。
次の TP1Client クラスのメソッドの説明を変更した。 • rpcOpen • openConnection

変更内容（3000-3-D59-40） uCosminexus TP1/Client/J 07-50

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-D59-30） uCosminexus TP1/Client/J 07-50

追加・変更内容
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。
DCCM3 論理端末に対して RPC を行う場合の負荷分散の説明を変更した。
DCCM3 論理端末に対して RPC を行う場合の注意事項を追加した。
トレースファイルの出力内容の注意事項を追加した。
デバッグトレースの説明を追加した。
次の TP1Client クラスのメソッドの説明を追加した。 • rpcCall • cltAssemSend • cltAssemReceive • setDccltinquiretime
CUP と TP1/Server 間のコネクションで確保される TCP/IP の送受信バッファのサイズを変更できるようにした。 これに伴い，TP1/Client/J 環境定義に，次のオペランドを追加した。 • dccltsendbuffsize • dccltrecvbuffsize
常設コネクションを使用し，かつオートコネクトモードの場合に，CUP 実行プロセスで「問い合わせ間隔最大時間」を監視するようにした。 これに伴い，TP1/Client/J 環境定義に，dcinquiretimecheck オペランドを追加した。
次のオペランドの説明を追加した。 • dcrapdirect • dcrcvport • dccltextend • dcsockopenatrcv • dccltinquiretime
TP1/Client/J 環境定義に関する注意事項を追加した。
次に示すバージョンの変更点を記載した。 • TP1/Client/J 07-03 • TP1/Client/J 07-50
TP1/Client/J が出力するファイルについて追加した。

uCosminexus TP1/Client/J 07-03

追加・変更内容
UAP トレースで取得するメソッドごとの情報に次のメソッドを追加した。

追加・変更内容
• setRpcServicePrio • getRpcServicePrio
CUP からのサービス要求に，優先順位を付加できるようにした（サービス要求のスケジュールプライオリティ設定機能）。 これに伴い，TP1Client クラスに次のメソッドを追加した。 • setRpcServicePrio • getRpcServicePrio

変更内容（3000-3-D59-20） uCosminexus TP1/Client/J 07-02

追加・変更内容
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るようにした（マルチスケジューラ機能）。 これに伴い，TP1/Client/J 環境定義に，次のオペランドを追加した。 • dcscdmulti • dcscdmulticount
従来メインフレームの OLTP での端末一斉起動と同様の運用ができるようにした（サーバからの一方通知受信機能）。 これに伴い，TP1Client クラスに，次のメソッドを追加した。 • acceptNotification • cancelNotification • openNotification • acceptNotificationChained • closeNotification また，次のクラスを追加した。 • ErrAcceptCanceledException • ErrVersionException
次の TP1Client クラスのメソッドから，UAP トレースを取得するようにした。 • setUapTraceMode • setErrorTraceMode • setMethodTraceMode • setDataTraceMode
UAP トレースに，性能解析トレースの識別情報が出力されるようにした。
バージョン 07-02 以降の TP1/Server に対して，リモート API 機能を使用した RPC を行う場合は，スケジューラに送信する RPC 電文中への識別情報（IP アドレスなど）を付加できるようにした。
RPC や TCP/IP 通信機能でのコネクション確立要求時に，送信元ホストを指定できるようにした（送信元ホスト指定機能）。 これに伴い，TP1/Client/J 環境定義に dccltcupsndhost オペランドを追加した。
スケジューラダイレクト機能を使用した RPC の受信ポート，およびネームサービスを使用した RPC の受信ポートを固定できるようにした（受信ポート固定機能）。 これに伴い，TP1/Client/J 環境定義に dccltcuprcvport オペランドを追加した。

追加・変更内容
TP1/Client/J 環境定義のオペランドの一覧を追加した。
rap サーバで代理実行するトランザクションブランチの，開始から終了までの最大実行時間を指定できるようにした。 これに伴い，TP1/Client/J 環境定義に dccltrcmplmttm オペランドを追加した。
TP1/Client/J 環境定義の dcrcvport オペランドについて，注意事項を追加した。
バージョンアップ時の，API，定義，コマンドおよびデフォルト値の変更を記載した。

変更内容（3000-3-D59-10） uCosminexus TP1/Client/J 07-01

追加・変更内容
TP1/Client/J が提供するファイルとファイルを格納するディレクトリについての説明を追加した。
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に，CUP に割り当てられる DCCM3 の論理端末を固定する，端末識別情報設定機能を追加した。 これに伴い，TP1Client クラスに setConnectInformation メソッドを追加した。 また，TP1/Client/J 環境定義に dccltconnectinf オペランドを追加した。
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する，データ圧縮機能を追加した。 これに伴い，TP1/Client/J 環境定義に dccltdatacomp オペランドを追加した。
複数のスレッドから，同一の TP1Client クラスのインスタンスのメソッドを実行する場合の注意事項を追加した。

はじめに

このマニュアルは，次の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。

- P-2464-73B4　uCosminexus TP1/Client/J
- P-2464-73C4　uCosminexus TP1/Client/J

なお，これらの製品は，Java2 Software Development Kit, Standard Edition, Version 5.0 以降に対応する OS でご利用できます。

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。

■ 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

TP1/Client/J の機能概要と特長について説明しています。

第 2 章　機能

TP1/Client/J の機能について説明しています。

第 3 章　プログラムインタフェース

TP1/Client/J を使用するときの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/J で使用するクラス

TP1/Client/J で使用するクラスについて説明しています。

第 5 章　定義

TP1/Client/J 環境定義について説明しています。

第 6 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。

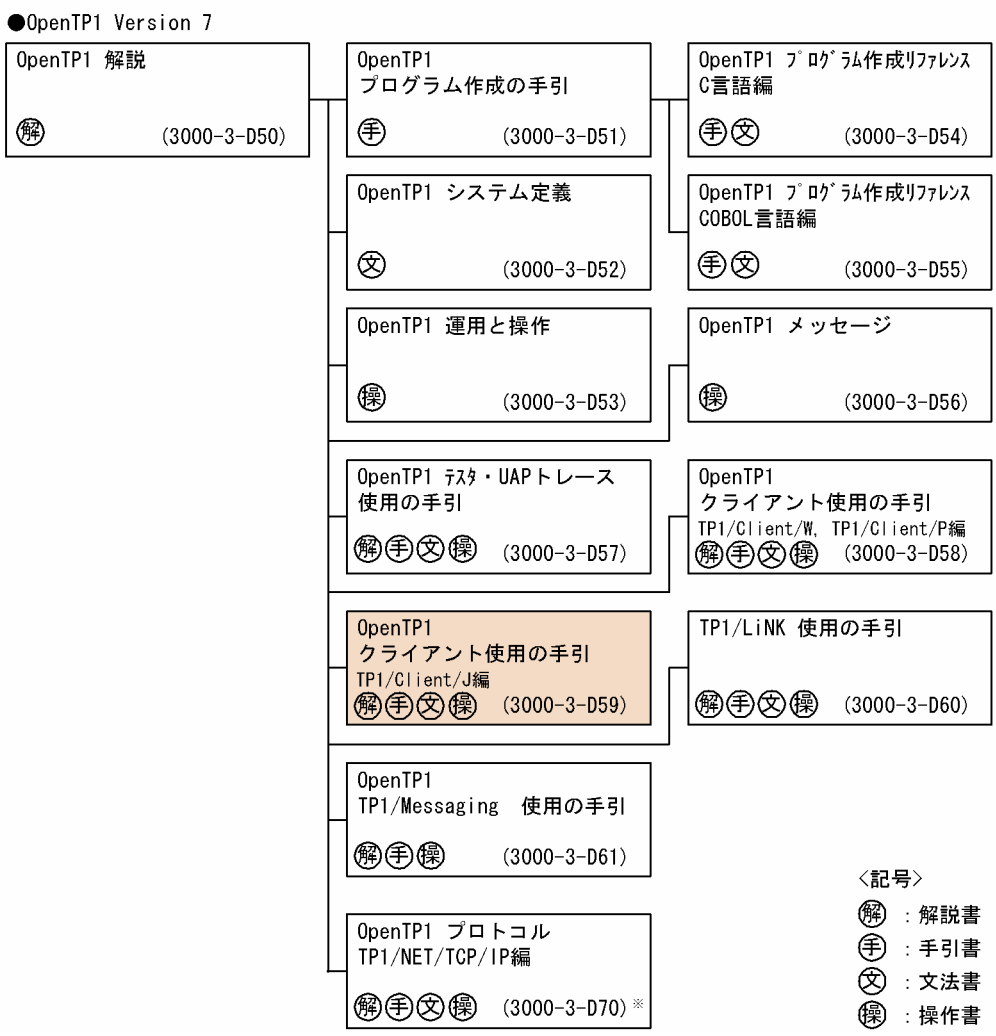
付録 A　バージョンアップ時の変更点

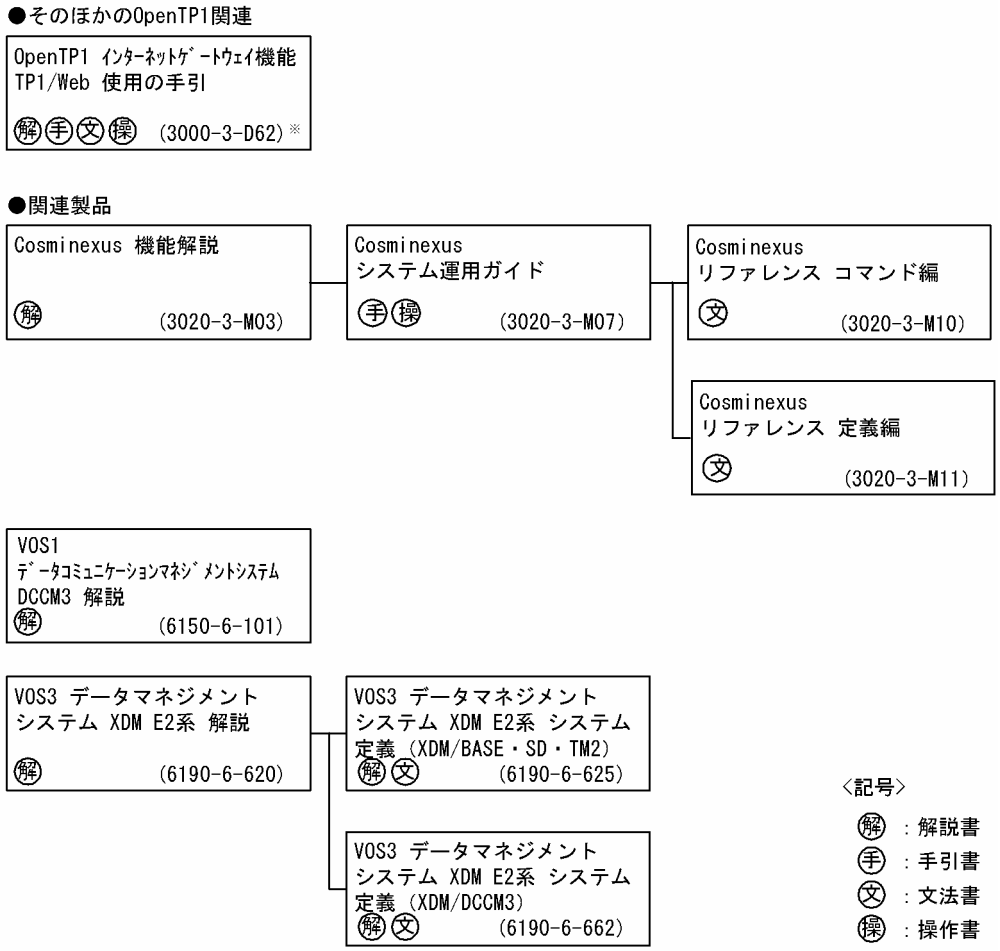
各バージョンでの API，定義およびコマンドの変更点について説明しています。

付録 B　TP1/Client/J が出力するファイル一覧

TP1/Client/J が出力するファイルについて説明しています。

■ 関連マニュアル

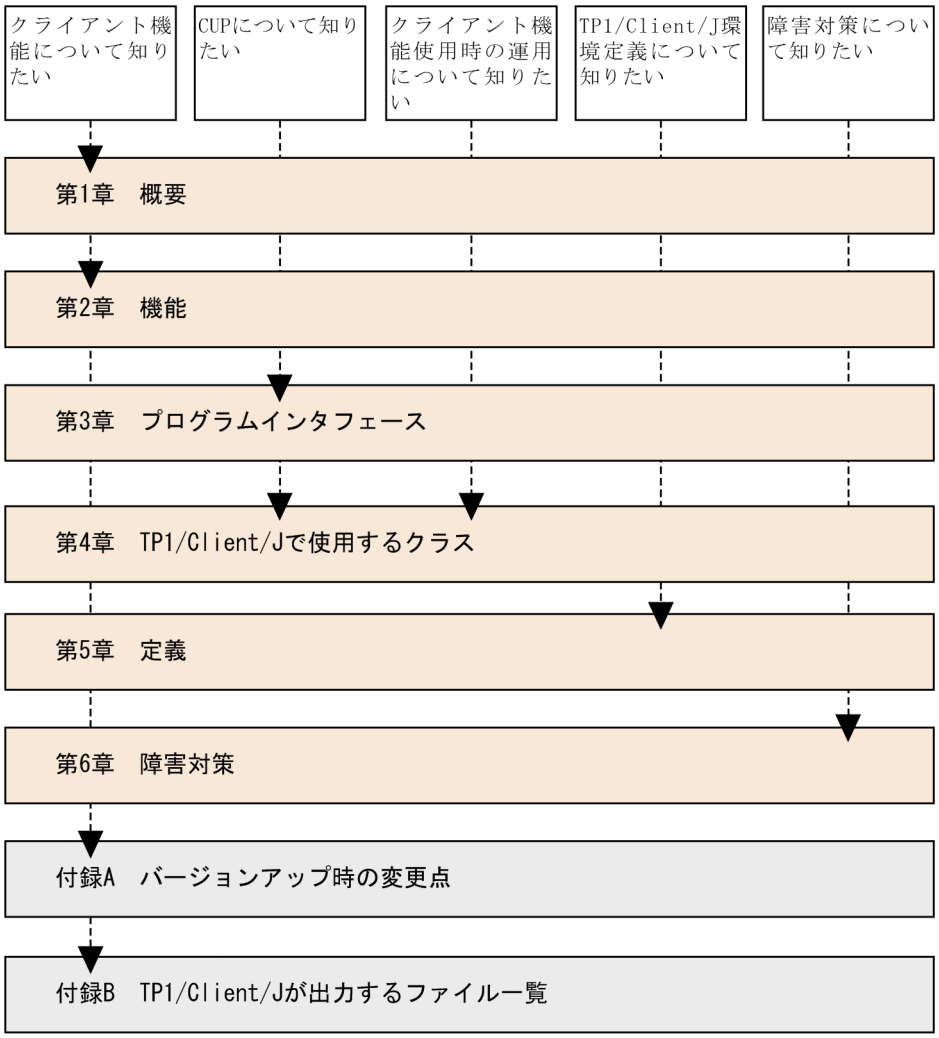




注※　このマニュアルおよびこのマニュアルが対象とする製品の発行時期についてはご確認ください。

■ 読書手順

このマニュアルは，利用目的に合わせて直接章を選択して読むことができます。利用目的別に，次の流れに従って，▼の部分をお読みいただくことをお勧めします。

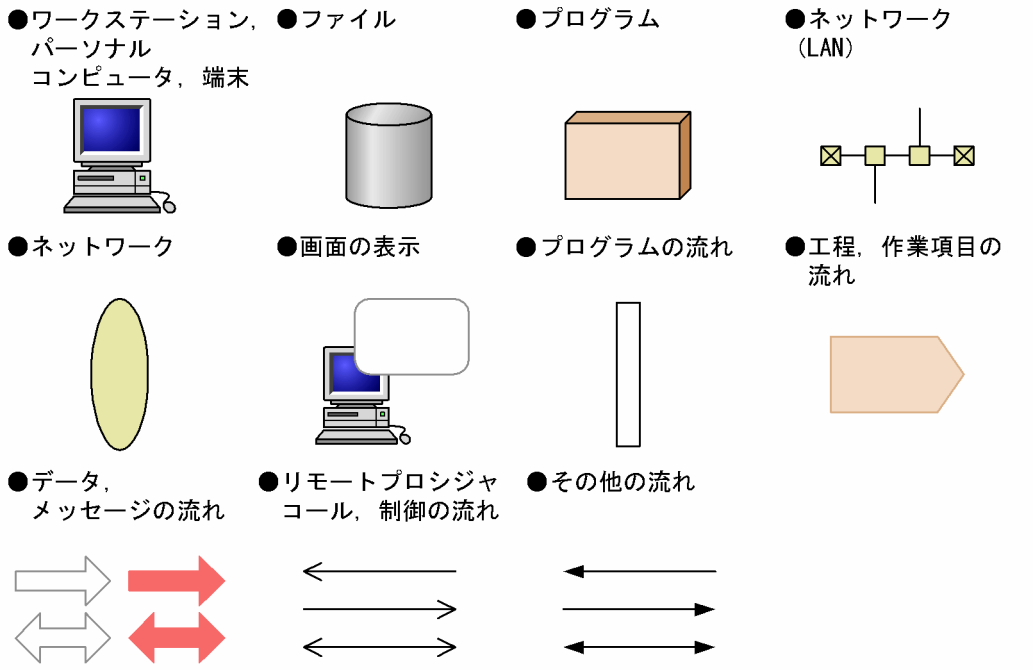


（凡例）

：　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。



■ このマニュアルでの表記

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	
uCosminexus Application Server Enterprise	Cosminexus Application Server	
uCosminexus Application Server Standard		
VOS1 DCCM3	DCCM3	
VOS3 XDM/DCCM3		
Java2 Software Development Kit, Standard Edition	JDK	
Java Platform, Standard Edition Development Kit		
uCosminexus TP1/Client/J	TP1/Client/J	
uCosminexus TP1/NET/TCP/IP	TP1/NET/TCP/IP	
uCosminexus TP1/NET/TCP/IP(64)		
uCosminexus TP1/LiNK	TP1/LiNK	TP1/Server
uCosminexus TP1/Server Base	TP1/Server Base	
uCosminexus TP1/Server Base(64)		
uCosminexus TP1/Web	TP1/Web	

製品名称	略称		
uCosminexus TP1/Web(64)	TP1/Web		
AIX 6.1	AIX		UNIX
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)			
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 (32-bit x86)			
Red Hat Enterprise Linux Server 7 (64-bit x86_64)			
Solaris 10	Solaris		
Oracle Solaris 11			
Itanium Processor Family	IPF		
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 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			

■ 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	Application Programming Interface
CGI	Common Gateway Interface
CUP	Client User Program
DCE	Distributed Computing Environment
EJB	Enterprise Java Beans
FD	Floppy Disk
HTTP	Hyper Text Transfer Protocol
J2EE	Java 2 Enterprise Edition
JDK	Java Development Kit
JSP	JavaServer Pages

英略語	英字での表記
MHP	Message Handling Program
OLTP	Online Transaction Processing
OS	Operating System
PC	Personal Computer
PRF	PeRFormance
RAP	Remote Application Programming Interface
RPC	Remote Procedure Call
SPP	Service Providing Program
TCP/IP	Transmission Control Protocol / Internet Protocol
TMS-4V/SP	Transaction Management System-4V/System Product
UAP	User Application Program
VM	Virtual Machine
WS	Workstation
WWW	World Wide Web

■ 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，$1{,}024^2$ バイト，$1{,}024^3$ バイト，$1{,}024^4$ バイトです。

目次

図目次

表目次

1 概要

OpenTP1 のクライアント機能の概要および特長について説明します。

1.1　クライアント機能の特長

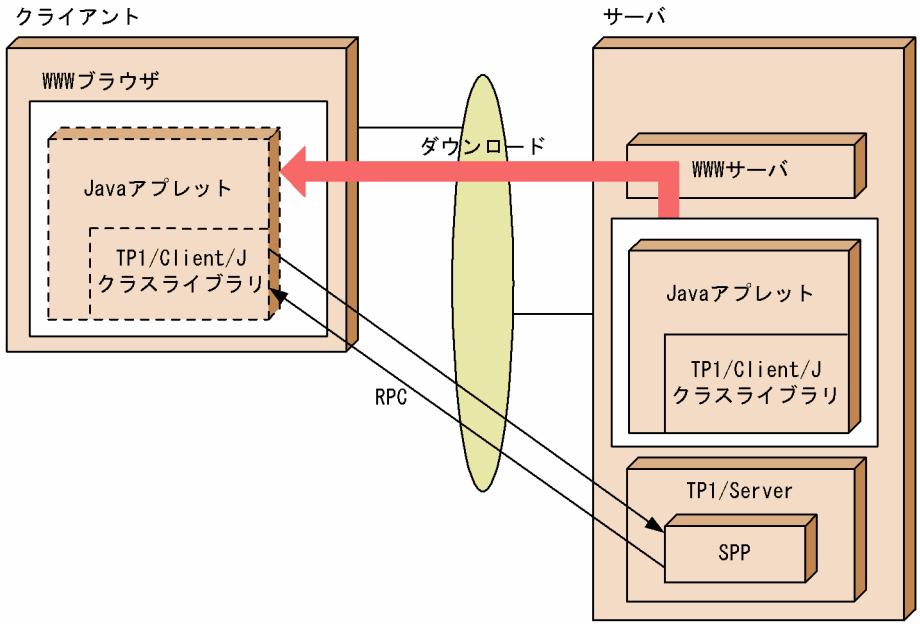
TP1/Client/J を使用すると，WWW ブラウザで動作する Java アプレット，Java アプリケーション，およびアプリケーションサーバで動作する Java サーブレットから，OpenTP1 のサーバ UAP へリモートプロシジャコールでサービスを要求できます。Java アプレット，Java アプリケーション，および Java サーブレットで作成したサービスを要求するプログラムを CUP といいます。CUP がサービスを要求できるサーバ UAP は，OpenTP1 のサービス提供プログラム（SPP）です。

TP1/Client/J 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，イントラネットおよびインターネット上の業務システムに分散 OLTP 環境を適用できます。また，CUP を Java アプレット，Java アプリケーション，および Java サーブレットで作成できるので，一つの CUP で幅広いプラットフォームをサポートできます。プラットフォームに合わせて複数の CUP を作成する必要はありません。

なお，このマニュアルでは，TP1/Server Base と TP1/LiNK の総称として，TP1/Server と記載します。TP1/Client/J からの要求を処理できる TP1/Server Base のバージョンは 03-05 以降となります。それ以前のバージョンでは動作しません。

CUP を Java アプレット，Java アプリケーション，および Java サーブレットで作成した場合の TP1/Server と TP1/Client/J の関係を，次の図に示します。

図 1-1　TP1/Server と TP1/Client/J の関係（Java アプレットの場合）



Java アプレットで CUP を作成した場合，CUP を WWW サーバで一括管理できます。CUP をクライアント側に配布する必要はありません。

図 1-2　TP1/Server と TP1/Client/J の関係（Java アプリケーションの場合）

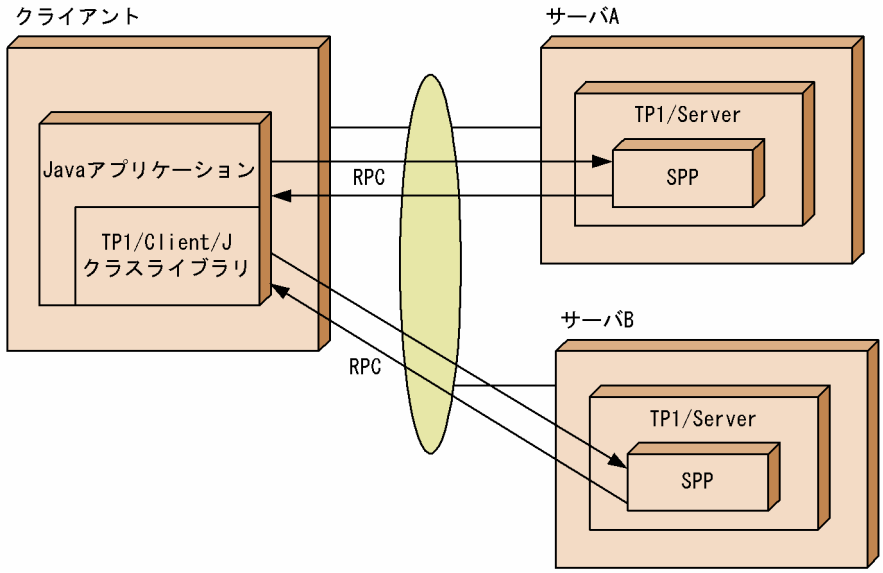
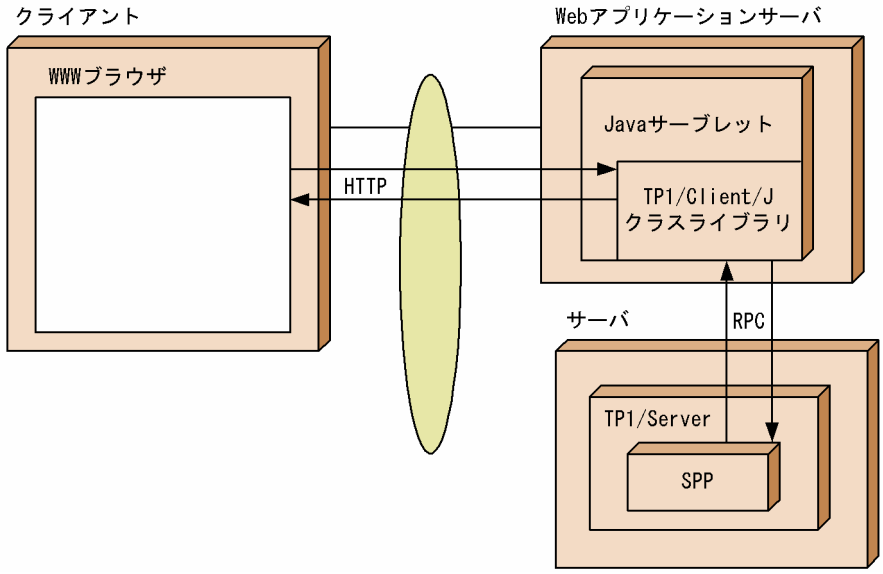


図 1-3　TP1/Server と TP1/Client/J の関係（Java サーブレットの場合）



1.2 TP1/Client/J の動作の仕組み

TP1/Client/J 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。これを**リモートプロシジャコール**（RPC）といいます。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。TP1/Client/J が使用できる RPC には，次の四つがあります。

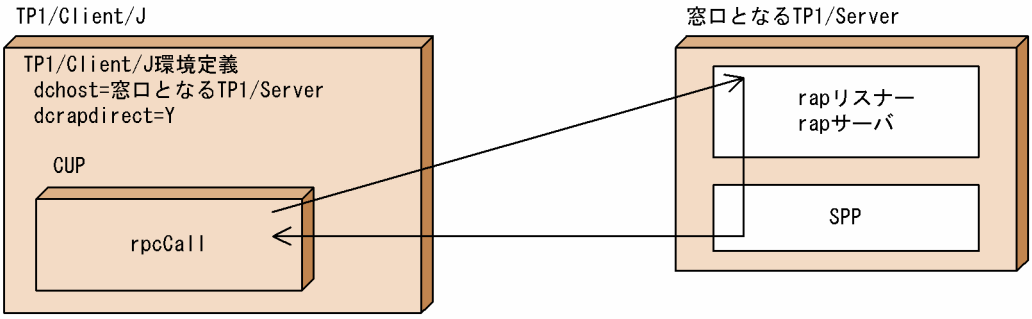
- リモート API 機能を使用した RPC
- スケジューラダイレクト機能を使用した RPC
- ネームサービスを使用した RPC
- 通信先を指定した RPC

窓口となる TP1/Server は，TP1/Client/J 環境定義の **dchost オペランド**で定義します。

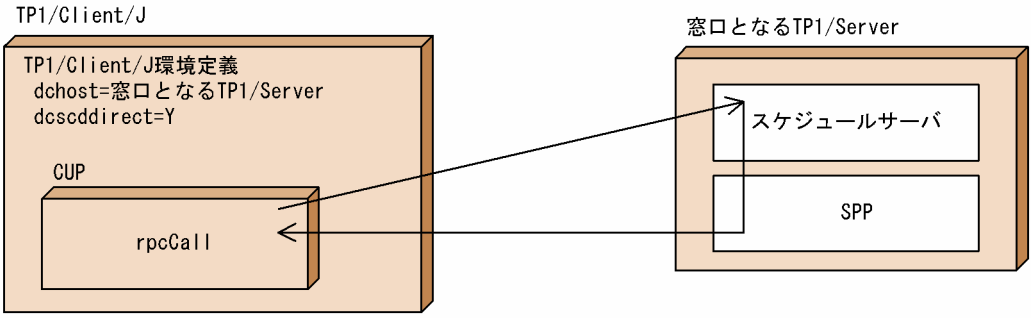
TP1/Client/J の動作の仕組みを，次の図に示します。

図 1-4　TP1/Client/J の動作の仕組み（1/2）

● リモートAPI機能を使用したRPC



● スケジューラダイレクト機能を使用したRPC



● ネームサービスを使用したRPC

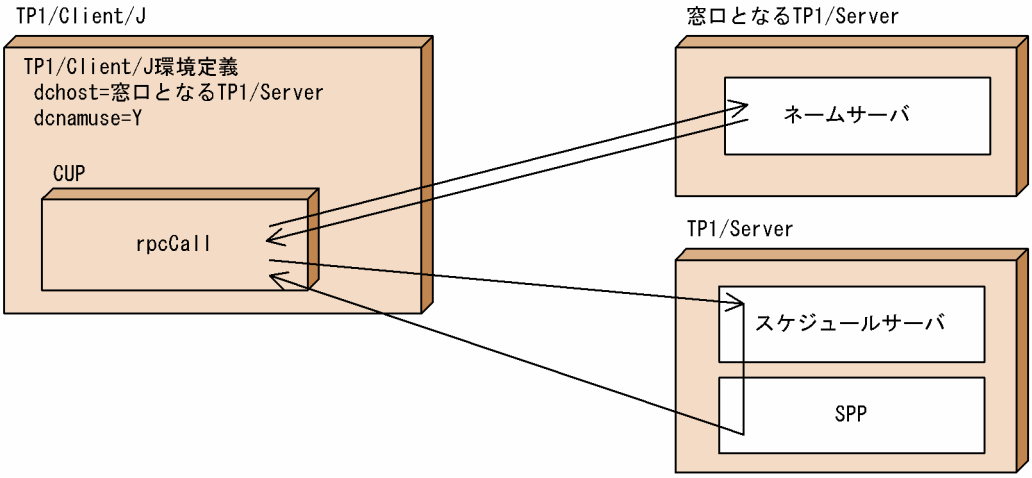
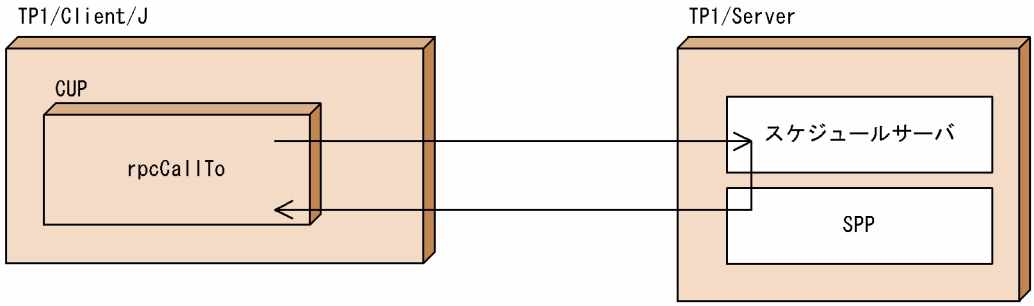


図 1-5　TP1/Client/J の動作の仕組み（2/2）

● 通信先を指定したRPC



1.3 TP1/Client/J の環境

TP1/Client/J は，OpenTP1 サーバへアプリケーションを呼び出すクラスライブラリファイルを提供します。TP1/Client/J のインストール先，開発環境，および実行環境について説明します。

1.3.1 インストール

(1) 提供媒体および提供形態

TP1/Client/J は，複数のプラットフォームに対応するため，Joliet フォーマットの CD-ROM で提供しています。PC 以外のマシンにインストールする場合は，一度 PC にインストールしたあと，インストールしたいマシンにバイナリモードでファイル転送するなどしてインストールしてください。

なお，TP1/Client/J のファイルはロングファイルネーム形式で格納されています。そのため，ファイル転送元の PC の OS は，ロングファイルネーム形式を識別できる必要があります。

TP1/Client/J のファイルは，大きく分けて次の表に示す二つのディレクトリに格納されています。

表 1-1 TP1/Client/J のファイルと格納ディレクトリ

格納ディレクトリ	TP1/Client/J のファイル	説明
LIB	TP1Client.jar	TP1/Client/J のクラスライブラリが，jar 形式で格納されています。
SAMPLES	AppletSample.html	TP1/Client/J を使用するアプレット，アプリケーション，およびサーブレットのサンプルソースが格納されています。
	AppletSample.java	
	ApplicationSample.java	
	betran.ini	
	ServletSample.java	

(2) クラスライブラリファイルのインストール先

クラスライブラリファイルのインストール先について説明します。

Java アプレット開発時および Java アプレット実行時

任意のディレクトリに TP1Client.jar ファイルを格納し，HTML の applet タグで指定してください。

Java アプリケーション実行時

JDK を使用する場合，任意のディレクトリに TP1Client.jar ファイルを格納し，環境変数 CLASSPATH を設定します。

Java サーブレット実行時

使用する Java サーブレットの仕様に従って，TP1Client.jar ファイルを格納してください。

(3) ファイルアクセス権限の設定

TP1/Client/J でアクセスするファイルとそのファイルに必要なアクセス権限を次の表に示します。

表 1-2 TP1/Client/J で使用するファイルとアクセス権限

ファイル種別	アクセス種別	格納場所
TP1/Client/J クラスライブラリ	読み取り	任意
TP1/Client/J 環境定義	読み取り	任意
UAP トレースファイル	書き込み	TP1/Client/J 環境定義の dcuaptracepath オペランド，または setUapTraceMode メソッドの path 引数で指定したディレクトリ
データトレースファイル	書き込み	TP1/Client/J 環境定義の dcdatatracepath オペランド，または setDataTraceMode メソッドの path 引数で指定したディレクトリ
メソッドトレースファイル	書き込み	TP1/Client/J 環境定義の dcmethodtracepath オペランド，または setMethodTraceMode メソッドの path 引数で指定したディレクトリ
エラートレースファイル	書き込み	TP1/Client/J 環境定義の dcerrtracepath オペランド，または setErrorTraceMode メソッドの path 引数で指定したディレクトリ
デバッグトレースファイル	書き込み	Java VM 実行ユーザのホームディレクトリ下の TP1clientJ ディレクトリ

TP1/Client/J を使用するクライアントプログラムを動作させる Java VM でセキュリティマネジャを適用する場合，TP1/Client/J のクラスライブラリがこれらファイルにアクセスできるよう，適切なパーミッションをセキュリティポリシファイルに記述してください。Java サーブレット，または EJB の場合，これらコンテナを実装しているアプリケーションサーバ製品で，セキュリティの制約のためデフォルトでセキュリティマネジャを使用していることがあります。使用するアプリケーションサーバの仕様をご確認の上，適切に設定してください。

(4) バージョンアップおよび修正版のインストール手順

バージョンアップおよび修正版のインストール手順について説明します。

TP1/Client/J が提供している実行環境に必要なファイルは，TP1Client.jar ファイルだけです。

入れ替え先の TP1Client.jar ファイルを使用している CUP が動作していない状態で，次の手順でファイルを入れ替え，Java VM に認識させてください。

なお，入れ替えの場合，CUP のリコンパイルが必要になりますが，修正版の場合，CUP のリコンパイルは不要です。

1. PC に CD-ROM をセットする。
   CD-ROM の「LIB」ディレクトリ下に TP1Client.jar ファイルが格納されています。
2. CD-ROM 上の TP1Client.jar ファイルを任意のディレクトリにコピーする。
   PC 以外のマシンにインストールする場合は，次のどちらかの方法でインストールしてください。

- 一度 PC にインストールしたあと，インストールしたいマシンにバイナリモードでファイル転送する。
- インストールしたいマシンに CD-ROM をセットし，マウント後，任意のディレクトリにコピーする。

3. TP1/Client/J を動作させるプログラムから見て，TP1Client.jar に読み取り権限がなければ，読み取り権限を付与する。

(5) JDK のバージョンアップ（update 版含む）に伴う CUP のリコンパイル

JDK をバージョンアップ（update 版含む）する場合，CUP のリコンパイル有無については JDK のドキュメントもしくは，製造元に確認してください。

1.3.2 開発環境

TP1/Client/J を稼働させるには，Java アプレット，Java アプリケーション，および Java サーブレットを開発する環境（Java コンパイラなど）が必要です。

JDK のバージョン

Java(TM)2 Software Development Kit, Standard Edition, Version 5.0（JDK 5.0）以降に対応します。

1.3.3 実行環境

TP1/Server サーバのバージョン

TP1/Client/J からの要求を処理できるのは，TP1/Server Base 03-05 以降です。

Cosminexus Application Server をご利用になる場合

Cosminexus Application Server の J2EE サーバモード※で Java サーブレット，EJB，または JSP から TP1/Client/J を使用する場合，あらかじめ次の設定を行ってください。

1. J2EE サーバ用オプション定義ファイル※を編集し，次の文字列を記述する。

   ```
   add.class.path=<TP1/Client/J インストールディレクトリ>/TP1Client.jar
   ```

2. サーバ管理コマンド用オプション定義ファイル※を編集し，次の文字列を記述する。

   Windows 版の場合

   ```
   set USRCONF_JVM_CLASSPATH=<TP1/Client/J インストールディレクトリ>/TP1Client.jar
   ```

   UNIX 版の場合

   ```
   set USRCONF_JVM_CLPATH=<TP1/Client/J インストールディレクトリ>/TP1Client.jar
   ```

   インストールディレクトリパス名の指定に関しては，使用する OS の記述形式に従って記述してください。

注※

J2EE サーバモード，J2EE サーバ用オプション定義，サーバ管理コマンド用オプション定義については，マニュアル「Cosminexus 機能解説」，マニュアル「Cosminexus リファレンス 定義編」を参照してください。

2 機能

OpenTP1 のクライアント機能について説明します。

2.1 常設コネクション

TP1/Client/J では，CUP（TP1/Client/J を使用した Java アプレット，Java アプリケーション，または Java サーブレット）と rap サーバの間のコネクションを確立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。なお，リモート API 機能を使用した RPC を実行するとき，連鎖型 RPC を使用するとき，およびトランザクション機能を使用するときは，必ず常設コネクションを確立してください。

常設コネクションを確立すると，コネクション確立・解放のための制御用パケットを減らせるため，通信の効率が向上します。

2.1.1 常設コネクションの確立・解放

常設コネクションを確立するには，**openConnection メソッド**を呼び出します。接続要求先の rap サーバの位置は，openConnection メソッドの引数で指定するか，TP1/Client/J 環境定義の dchost オペランド，および dcrapport オペランドで指定します。ただし，CUP と rap リスナー，rap サーバの間にファイアウォールがある場合は，rap サーバの代わりにファイアウォールの位置を指定してください。

openConnection メソッドが例外を返した場合は，常設コネクションは確立されていません。

TP1/Client/J が同時に確立できる常設コネクションは，TP1Client クラスの 1 つのインスタンスに対して 1 つのコネクションだけです。TP1/Client/J の常設コネクションには，管理方式によって次の 2 つのモードがあります。

非オートコネクトモード

非オートコネクトモードは，CUP で明示的に openConnection メソッドを呼び出して TP1/Server の rap リスナー，rap サーバとのコネクションを確立します。openConnection メソッドを呼び出さないで rpcCall メソッドを呼び出した場合，rpcCall メソッドは ErrProtoException を返します。コネクションを解放するには，CUP で closeConnection メソッドを明示的に呼び出します。

オートコネクトモード

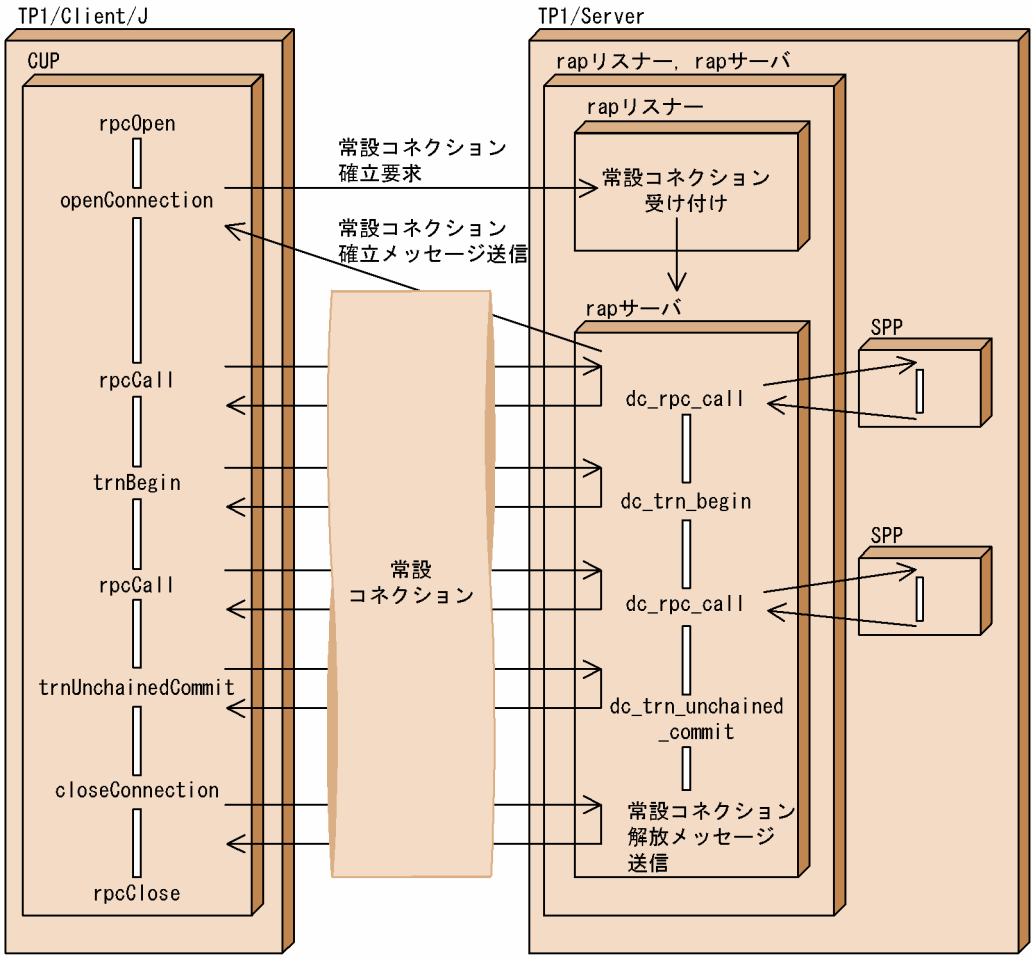
オートコネクトモードは，TP1/Client/J でコネクションを管理します。TP1/Client/J は，rpcCall メソッドまたは trnBegin メソッドが最初に呼び出されたときにコネクションが確立されていないと判断すると，自動的に rap リスナー，rap サーバとの間に常設コネクションを確立します。このため，openConnection メソッドの呼び出しは不要です。呼び出した場合，ErrProtoException を返します。コネクションの解放は rpcClose メソッド内で行います。TP1/Client/J で管理しているコネクションを強制的に解放したい場合は，closeConnection メソッドを呼び出します。

非オートコネクトモードを使用するか，オートコネクトモードを使用するかは TP1/Client/J 環境定義の dcrapautoconnect オペランドで指定するか，または setRpcextend メソッドで指定します。

closeConnection メソッドを呼び出して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が使用されます。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



2.1.2　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。

- TP1/Client/J 環境定義
  - dccltinquiretime
  - dchost
  - dcrapautoconnect
  - dcrapdirect
  - dcrapport
- rap リスナーサービス定義

  rap リスナーサービス定義の詳細については，マニュアル「OpenTP1 システム定義」を参照してください。

2.1.3　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は常設コネクションを確立できません。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。

2.2　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。

2.2.1　RPC の実現方法

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。

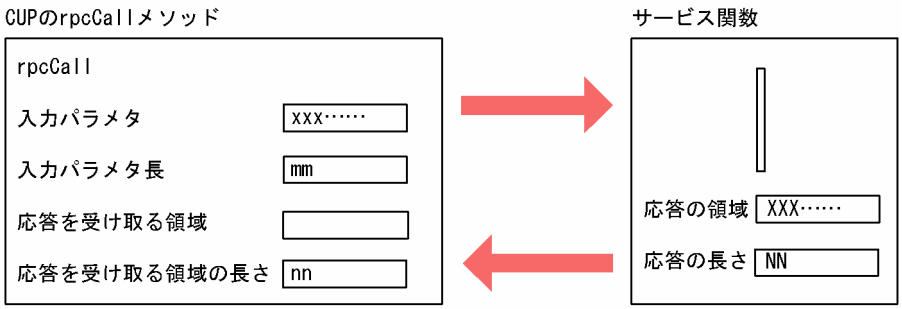
CUP からサービスを要求するメソッドを呼び出して，SPP にサービス要求します。サービスを要求するときには，SPP のサービスグループ名とサービス名をパラメタに設定した rpcCall メソッドを呼び出します。

2.2.2　RPC でのデータの受け渡し

RPC でのデータの受け渡しは，rpcCall メソッドに，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，応答長を指定して呼び出します。rpcCall メソッドで設定できる応答長の最大値は 1 メガバイトです。この値は，TP1/Client/J 環境定義の dccltrpcmaxmsgsize オペランドで変更できます。

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，rpcCall メソッドで設定した値（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。

2.2.3　RPC の形態

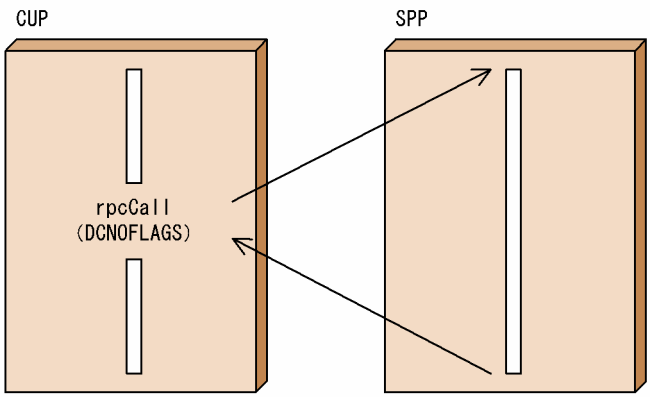
TP1/Client/J で実行できる RPC の形態は，同期応答型 RPC，非応答型 RPC，および連鎖型 RPC です。

(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。CUP は，SPP から処理結果が返ってくるまで，次の処理を待ちます。同一サービスを複数回呼び出した場合，サーバプロセスはそのたびにスケジュールされます。

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



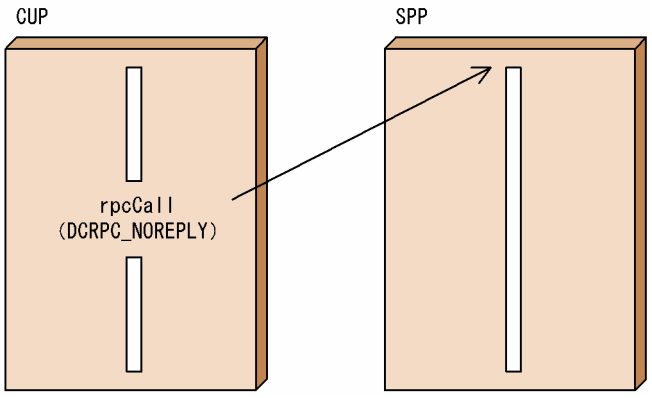
(2) 非応答型 RPC

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。CUP は，SPP へ問い合わせメッセージを送信したあと，SPP から処理結果を受け取らないで，すぐに次の処理を実行します。

非応答型 RPC では，何らかの通信障害が発生した場合や呼び出したサービスが誤っていた場合でもエラーを受け取れません。

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



(3) 連鎖型 RPC

TP1/Client/J のリモート API 機能を使用した CUP から SPP へ問い合わせメッセージを送信し，問い合わせた応答メッセージを受信する形態です。CUP は，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 との違いは，サーバプロセスを固定できることです。連鎖型 RPC の場合，同一サービスに複数回問い合わせメッセージを送信しても，サーバプロセスは再スケジュールされません。

連鎖型 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型 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。

連鎖型 RPC の開始

連鎖型 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 rpcCall メソッドの flags 引数に 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 は連鎖型 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 引数にも DCRPC_CHAINED を指定します。

連鎖型 RPC の終了

連鎖型 RPC は，次のどちらかの方法で終了します。

- 連鎖型 RPC を実行しているサービスグループに対して，flags 引数に DCNOFLAGS を指定した rpcCall メソッド（同期応答型 RPC）を実行する。※
- 連鎖型 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。

注※

同期応答型 RPC を実行しないで closeConnection メソッド，または rpcClose メソッドを呼び出すと，次の状態となります。

- グローバルトランザクションの範囲外の場合
  サービスを実行していたプロセスは連鎖型 RPC タイムアウトになるまで占有されます。
- グローバルトランザクションの範囲内の場合
  暗黙的にコミットされ，連鎖型 RPC を終了します。

連鎖型 RPC の時間監視

連鎖型 RPC でサービスを要求される UAP は，CUP に応答を返してから，次のサービス要求が来るまでの間，またはトランザクションの同期点処理が来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判断して，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 オペランドで指定します。

2.2.4　スケジュール機能

TP1/Server のスケジュール機能は，CUP から 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。

2.2.5　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。

- 複数のノードに同一の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して，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 TP1/Client/J 側，TP1/Server 側の関連する定義，処理，および RPC の処理を説明します。

(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/J 側の定義

TP1/Client/J 側の定義で必要な設定はありません。

(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/J 側の定義

TP1/Client/J 環境定義で次の設定をする必要があります。

- dcnamuse=Y
- 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（デフォルト値）

(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，スケジュールを依頼する OpenTP1 ノードが複数ある場合，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。RPC ごとにサービス要求先スケジューラをラウンドロビン方式で分散させる場合は，TP1/Client/J 環境定義に dcscdhostchange=Y を指定します。

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，TP1/Client/J 環境定義に dchostselect=Y を指定します。

(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，TP1/Client/J 環境定義に dccltloadbalance=Y を指定します。これによって，TP1/Client/J 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の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の情報は，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の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の情報を取得する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，TP1/Client/J 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。

注意事項

- TP1/Client/J 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ヘッドが掛かります。
- TP1/Client/J 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ただちに削除され，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ヘッドが掛かります。
- TP1/Client/J 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。

(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 RPC を使用した場合，ノード間負荷バランス機能は使用できません。

2.2.6 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，TP1/Client/J 環境定義の dcwatchtim オペランドで指定します。

また，CUP から **setDcwatchtim メソッド**を呼び出して，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。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。

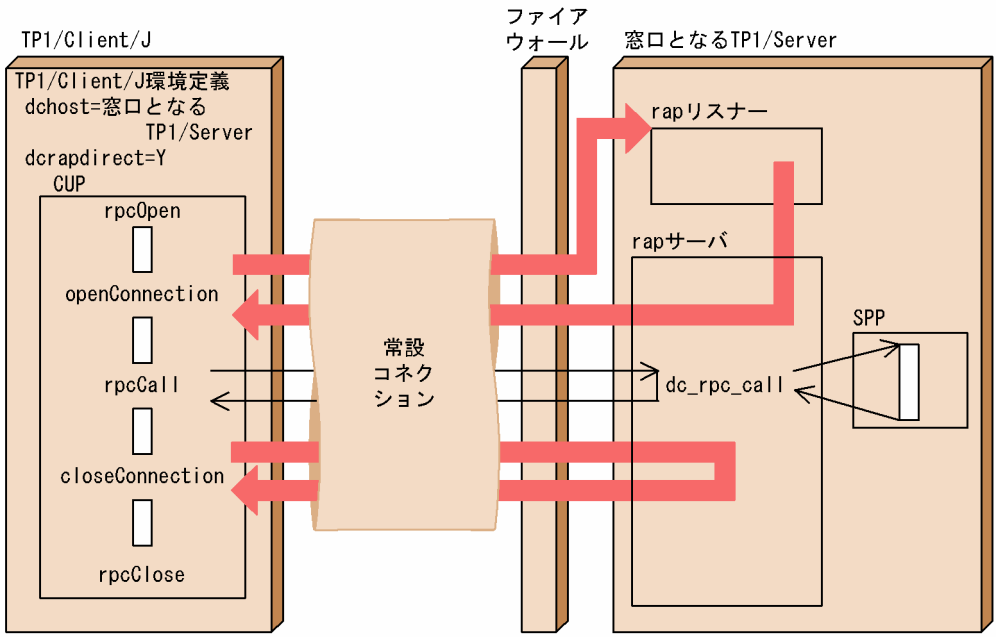
2.2.7 リモート API 機能を使用した RPC

TP1/Client/J では，CUP と TP1/Server 間に常設コネクションを確立して，CUP が発行した API を，TP1/Server 側に転送して TP1/Server 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。ただし，Java アプレットの場合は，Java のセキュリティの制約のため，その Java アプレットが属する WWW サーバ上の TP1/Server で起動している rap サーバとのコネクションしか確立できません。Java アプレットが属する WWW サーバ上の TP1/Server 以外で起動している rap サーバとコネクションを確立しようとすると，セキュリティマネジャが SecurityException を返します。なお，リ

モート API 機能を使用した RPC は，TP1/Server のほかに，DCCM3 および TMS-4V/SP を使用できます。

リモート API 機能を使用したサービス要求の流れを次の図に示します。

図 2-5　リモート API 機能を使用したサービス要求の流れ



1. TP1/Client/J が提供する TP1Client クラスのインスタンスを作成する。
2. openConnection メソッドを呼び出して TP1/Server の rap サーバとの間にコネクションを確立する。
3. rpcCall メソッドを呼び出して rap サーバ経由で目的の SPP にサービス要求をする。
   rpcCall メソッドは，コネクション確立中に何回でも呼び出せます。
4. closeConnection メソッドを呼び出して TP1/Server の rap サーバとのコネクションを解放する。

注

DCE で管理されているセキュリティ対象サーバに対して rap サーバを使用してサービス要求をすると，ErrSecchkException が返されます。

TP1/Client/J 環境定義の dccltrpcmaxmsgsize オペランドを指定してリモート API 機能を使用する場合，システム共通定義に rpc_max_message_size オペランドを指定していない TP1/Server の SPP に対してサービス要求をすると，RPC がエラーリターンします。

2.2.8　スケジューラダイレクト機能を使用した RPC

スケジューラダイレクト機能を使用した RPC は，Java アプリケーション，および Java サーブレットから実行できるサービス要求です。Java アプレットのセキュリティの制約があるため，Java アプレットからは実行できません。

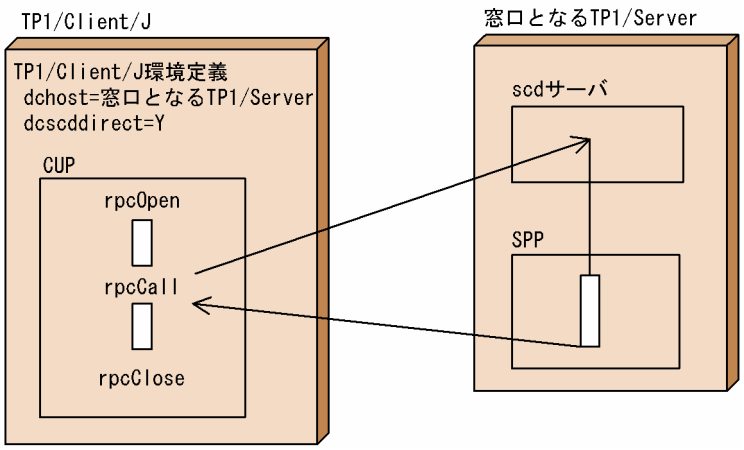
スケジューラダイレクト機能を使用した RPC の流れと，RPC ごとにサービス要求先スケジューラを分散させる場合の定義について説明します。

(1)　スケジューラダイレクト機能を使用した RPC の流れ

スケジューラダイレクト機能を使用した RPC を行う場合は，TP1/Client/J 環境定義に dcscddirect=Y を指定します。さらに，TP1/Client/J 環境定義に dchost および dcscdport オペランドを指定するか，または setDchost メソッドで scd サーバ（スケジュールサーバ）を指定します。また，rpcCall メソッドを呼び出す前には rpcOpen メソッドを呼び出す必要があります。CUP の最後には rpcClose メソッドを呼び出す必要があります。

スケジューラダイレクト機能を使用した RPC のサービス要求の流れを次の図に示します。

図 2-6　スケジューラダイレクト機能を使用した RPC のサービス要求の流れ



1. TP1/Client/J が提供する TP1Client クラスのインスタンスを作成する。
2. rpcOpen メソッドを呼び出して CUP の RPC 環境を初期化する。
3. rpcCall メソッドを呼び出して TP1/Server の scd サーバを経由して目的の SPP にサービス要求をする。※

   rpcCall メソッドは，rpcOpen メソッドを呼び出してから rpcClose メソッドを呼び出すまでの間，何回でも呼び出せます。
4. rpcClose メソッドを呼び出して RPC 環境を解放する。

注※

rpcCall メソッド内部の処理の流れを次に示します。

1. dchost オペランドおよび dcscdport オペランドで定義された scd サーバに対してコネクションを確立する。
2. scd サーバとのコネクションの確立後，サービス要求を送信しコネクションを切断する。
3. SPP から応答メッセージ送信用のコネクション確立要求を受信後，コネクションを確立し応答メッセージを受信する。

4. SPP との間で確立したコネクションを解放する。

なお，この機能の使用時は，ソケット受信型 SPP に対して RPC を発行できません。また，TP1/Client/J 環境定義の dccltrpcmaxmsgsize オペランドを指定してこの機能を使用した場合，通信先の TP1/Server ノードでエラーが発生することがあります。

(2) 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の定義

スケジューラダイレクト機能を使用した RPC では，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，分散させることができます。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，TP1/Client/J 環境定義に dcscdhostchange=Y を指定します。

2.2.9 ネームサービスを使用した RPC

ネームサービスを使用した RPC は，Java アプリケーションおよび Java サーブレットから実行できるサービス要求です。Java アプレットのセキュリティの制約があるため，Java アプレットからは実行できません。

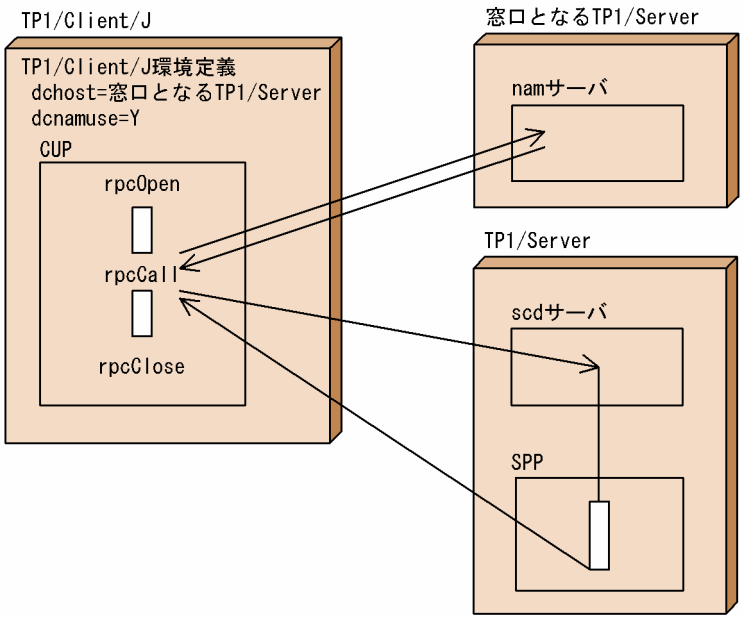
ネームサービスを使用した RPC の流れと，RPC ごとにサービス要求先スケジューラを分散させる場合の定義について説明します。

(1) ネームサービスを使用した RPC の流れ

ネームサービスを使用した RPC を行う場合は，TP1/Client/J 環境定義に dcnamuse=Y を指定します。さらに，TP1/Client/J 環境定義に dchost オペランド，および dcnamport オペランドを指定するか，または **setDchost メソッド**で nam サーバを指定します。また，rpcCall メソッドを呼び出す前には rpcOpen メソッドを呼び出す必要があります。CUP の最後には **rpcClose メソッド**を呼び出す必要があります。

ネームサービスを使用した場合のサービス要求の流れを次の図に示します。

図 2-7　ネームサービスを使用した場合のサービス要求の流れ



1. TP1/Client/J が提供する TP1Client クラスのインスタンスを作成する。
2. rpcOpen メソッドを呼び出して CUP の RPC 環境を初期化する。
3. rpcCall メソッドを呼び出して該当する SPP にサービス要求をする。※
4. rpcClose メソッドを呼び出して RPC 環境を解放する。

注※

rpcCall メソッド内部の処理の流れを次に示します。

1. dchost オペランドおよび dcnamport オペランドで定義された nam サーバに対してコネクションを確立する。
2. サービス情報を取得するための要求を送信し，nam サーバとの間に確立したコネクションを解放する。
3. nam サーバから応答メッセージ送信用のコネクション確立要求を受信後，コネクションを確立しサービス情報を受信する。
4. nam サーバとのコネクションを解放し，nam サーバからの応答メッセージのサービス情報を基に，サービスを実行している TP1/Server 上の scd サーバに対してコネクションを確立する。
5. scd サーバとのコネクションの確立後，サービス要求を送信しコネクションを切断する。
6. SPP から応答メッセージ送信用のコネクション確立要求を受信後，コネクションを確立し応答メッセージを受信する。
7. SPP との間で確立したコネクションを解放する。

なお，この機能の使用時は，ソケット受信型 SPP に対して RPC を発行できません。また，TP1/Client/J 環境定義の dccltrpcmaxmsgsize オペランドに 2 以上を指定した場合，ネームサービスを使用した RPC は，システム共通定義の rpc_max_message_size オペランドをサポートする TP1/Server 上で稼働してい

るサービスの情報だけを取得します。サービスの入力パラメタ長が，サービス要求先のシステム共通定義の rpc_max_message_size オペランドの値を超える場合もサービス要求は実行されます。システム共通定義の rpc_max_message_size オペランドをサポートしない TP1/Server 上の SPP に対しては RPC を発行できません。

(2) 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の定義

ネームサービスを使用した RPC では，サービス要求先スケジューラの情報をネームサーバからキャッシュに格納し，キャッシュの情報を参照して RPC ごとにサービス要求先スケジューラを分散させることができます。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の TP1/Client/J 側，TP1/Server 側の関連する定義について説明します。

(a) TP1/Client/J 側の定義

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，TP1/Client/J 環境定義に dccltloadbalance=Y を指定します。

これによって，TP1/Client/J 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の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は，一つだけではなく，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の情報を取得する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。

キャッシュの有効期限の指定

TP1/Client/J 環境定義の dccltcachetim オペランドで，サービス要求先スケジューラの情報を保持する，キャッシュの有効期限を指定できます。

キャッシュにサービス要求先スケジューラの情報を格納したあとで，キャッシュの有効期限を満了した場合，そのサービス要求先スケジューラの情報を破棄します。その後，サービス要求先スケジューラの情報を再度取得して，その情報をキャッシュに格納することで，キャッシュを更新します。

TP1/Client/J 環境定義の dccltcachetim オペランドは，TP1/Client/J 環境定義に dccltloadbalance=Y を指定した場合だけ有効になります。

(b) TP1/Server 側の定義

次に示す TP1/Server 側の定義のオペランドを指定すると，ネームサーバからサービス要求先スケジューラの情報を取得するときに，スケジューラの負荷情報も同時に取得します。

スケジュールサービス定義

- scd_announce_server_status=Y（デフォルト）※1

ユーザサービス定義，またはユーザサービスデフォルト定義

- loadcheck_interval

- levelup_queue_count※2
- leveldown_queue_count※2

注※1

このオペランドに N を指定した場合，スケジューラの負荷レベルは常に LEVEL0（負荷が低い状態）となります。負荷レベルが常に LEVEL0 のスケジューラが，TP1/Client/J のサービス要求先スケジューラとなった場合，実際の負荷状態に関係なく，常にサービス要求先スケジューラとして選択されます。

注※2

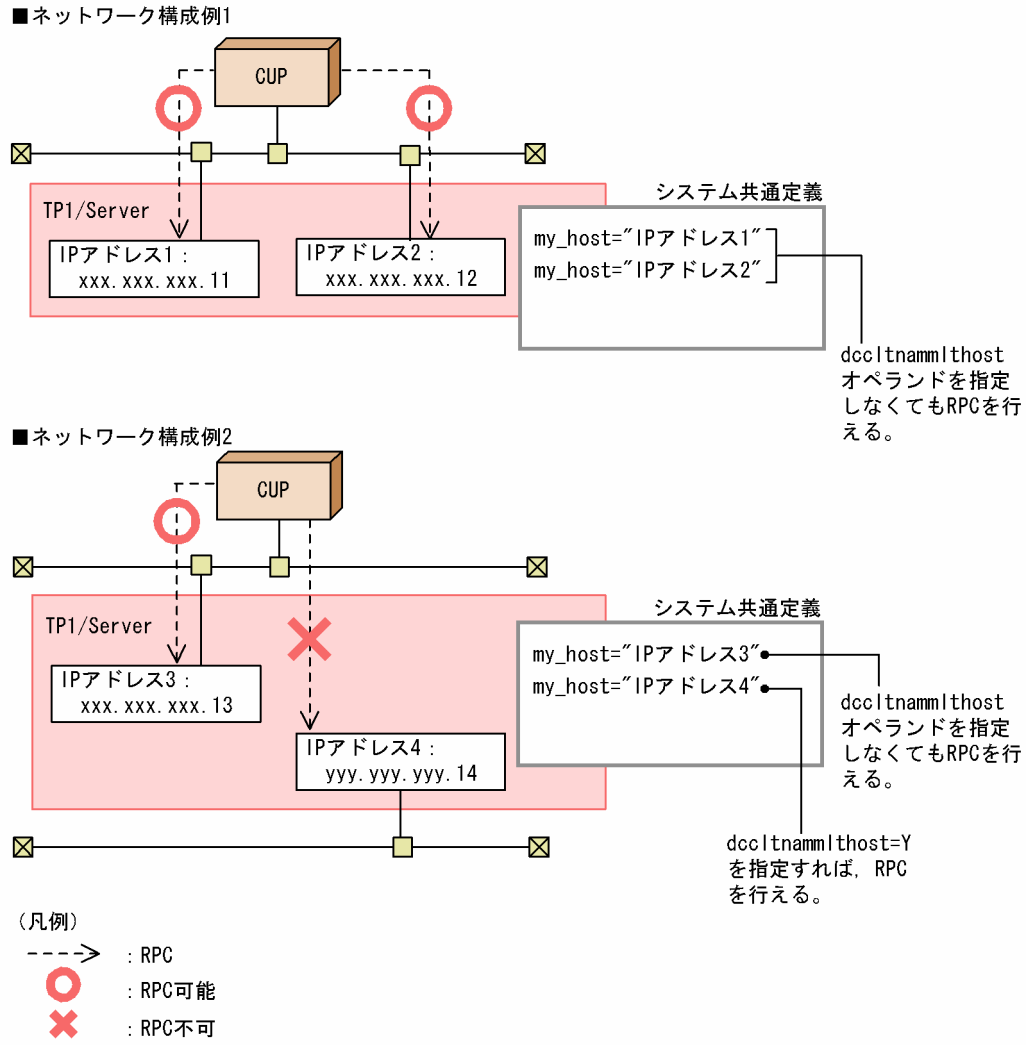
スケジューラの負荷レベルがサービス要求の滞留数で決まる方式とする場合に指定します。

(3) マルチホームドホスト形態の TP1/Server に対して RPC を行う場合の定義

ネームサービスを使用した RPC では，通信先 TP1/Server がマルチホームドホスト形態の場合に通信障害が発生することがあります。これは，CUP が存在するネットワークから，通信先 TP1/Server のシステム共通定義の my_host オペランドで指定したホスト名（IP アドレス）に対応するネットワークアダプタに通信できないことが原因です。

これを回避するためには，TP1/Client/J 環境定義に dccltnammlthost=Y を指定します。ネットワークの構成例を次の図に二つ示し，それぞれのネットワーク構成での TP1/Client/J 環境定義の dccltnammlthost オペランドの要否を示します。

図 2-8　マルチホームドホスト形態の TP1/Server に対して RPC を行う場合の例



ネットワーク構成例 2 の CUP が TP1/Server と通信するには，TP1/Client/J 環境定義に dccltnammlthost=Y の指定が必要です。

通信先 TP1/Server がマルチホームドホスト形態で，かつ同じ TP1/Server 上にサービス要求先の SPP が存在しているときは，TP1/Client/J 環境定義に dccltnammlthost=Y を指定することで RPC が行えるようになります。通信先 TP1/Server のシステム共通定義の all_node オペランドに，CUP が接続されていないネットワーク上に存在する TP1/Server が指定されている場合，この TP1/Server 上の SPP に対して RPC を行うと通信障害が発生します。なお，通信先 TP1/Server がマルチホームドホスト形態でない場合，TP1/Client/J 環境定義の dccltnammlthost オペランドの指定に関係なく RPC を行えます。

2.2.10　通信先を指定した RPC

通信先を指定した RPC を使用すると，サービス要求を実行するサーバ（通信先の OpenTP1 ノード）を明示的に指定できます。

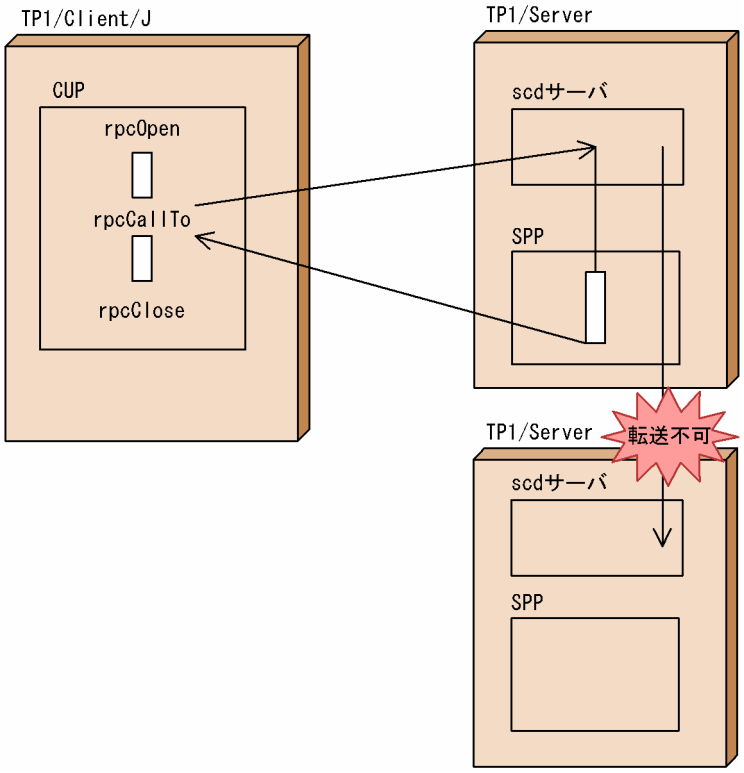
通信先を指定した RPC を使用した場合，サービス要求を実行するサーバに指定した通信先の OpenTP1 ノードでサービス要求が実行されます。指定した OpenTP1 ノードが停止，または閉塞しているときでも，他 OpenTP1 ノードへの転送は行われません。また，ノード間負荷バランス機能は使用できません。

通信先を指定した RPC を使用するには，rpcCallTo メソッドを呼び出します。rpcCallTo メソッドは，DCRpcBindTbl オブジェクトに格納されたホスト名にある TP1/Server のスケジュールサービスに対して直接サービス要求を実行します。

なお，この機能はリモート API 機能を使用した RPC との併用はできません。

通信先指定 RPC を使用した場合のサービス要求の流れを次の図に示します。

図 2-9　通信先指定 RPC を使用したサービス要求の流れ



1. TP1/Client/J が提供する TP1Client クラスのインスタンスを作成する。
2. rpcOpen メソッドを呼び出して CUP の RPC 環境を初期化する。
3. rpcCallTo メソッドを呼び出して TP1/Server の scd サーバを経由して目的の SPP にサービス要求をする。

   通信先 SPP の状態に関係なく，同一ノード内の SPP でサービス要求処理が行われます。

   rpcCallTo メソッドは，rpcOpen メソッドを呼び出してから rpcClose メソッドを呼び出すまでの間，何回でも呼び出せます。
4. rpcClose メソッドを呼び出して RPC 環境を解放する。

なお，この機能の使用時は，ソケット受信型 SPP に対して RPC を発行できません。また，TP1/Client/J 環境定義の dccltrpcmaxmsgsize オペランドを指定してこの機能を使用した場合，通信先の TP1/Server ノードでエラーが発生することがあります。

2.2.11　同期応答型 RPC タイムアウト時のサーバ負荷軽減

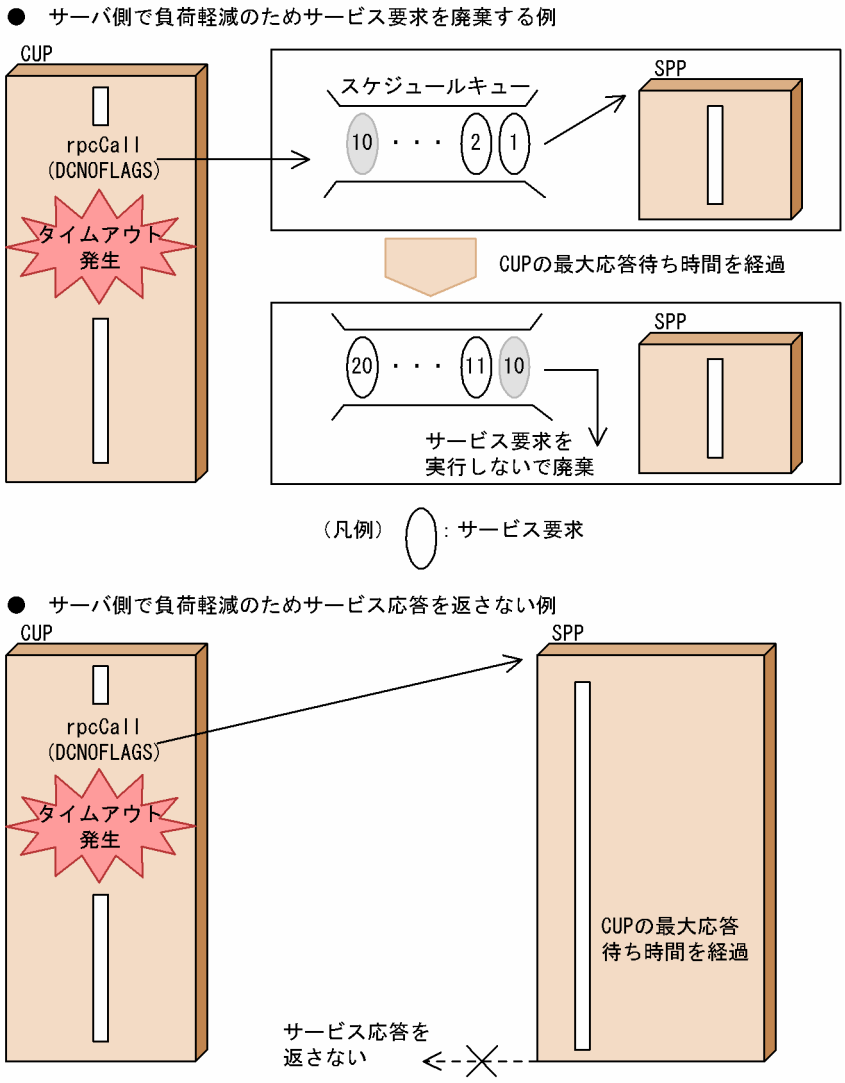
TP1/Client/J の CUP から rpcCall メソッドを呼び出すと，TP1/Server ではサービス要求を受け付けます。

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，遅れる可能性があるため，TP1/Client/J 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/J の最大応答待ち時間を認識していないため，TP1/Client/J 側でタイムアウトを検出していても，TP1/Server 側で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 側での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，TP1/Client/J 環境定義の dcwatchtimrpcinherit オペランドで指定します。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 2-10　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



2.2.12　ServerSocket を使用する RPC

TP1/Client/J で，スケジューラダイレクト機能を使用した RPC，ネームサービスを使用した RPC，および通信先を指定した RPC を行う場合，ServerSocket を使用します。TP1/Client/J を使用するクライアントプログラムを動作させる Java VM でセキュリティマネジャを適用する場合，TP1/Client/J のクラスライブラリが ServerSocket の機能（connect，listen，および accept）を使用できるよう，適切なパーミッションをセキュリティポリシファイルに記述してください。Java サーブレット，または EJB の場合，これらコンテナを実装しているアプリケーションサーバ製品で，セキュリティの制約のためデフォルトでセキュリティマネジャを使用していることがあります。使用するアプリケーションサーバの仕様をご確認の上，適切に設定してください。

2.2.13　マルチスケジューラ機能を使用した RPC

CUP からスケジュールキューを使う 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 のスケジュールキューに格納します。スケジューラデーモンとは，スケジュール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ル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**，サービス要求受信専用デーモンを**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方法

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，スケジューラダイレクト機能，またはネームサービスを使用した RPC を実行できます。

(a)　スケジューラダイレクト機能を使用した RPC

TP1/Client/J 環境定義 dcscddirect オペランドに Y を指定して，スケジューラダイレクト機能を使用した RPC を行う場合について説明します。

TP1/Client/J では，窓口となる TP1/Server のネームサービスへ問い合わせないで，マルチスケジューラデーモンをランダムに選択できます。これによって通信回数が削減され，ネームサービスの負荷を軽減できます。

マルチスケジューラデーモンをランダムに選択するためには，TP1/Client/J 環境定義の dchost オペランド，または dcscdport オペランドに次のポート番号を指定します。

- スケジュールサービス定義の scd_port オペランドに指定された，スケジュールサービスのポート番号
- スケジュールサービス定義の scdmulti の-p オプションに指定されたポート番号

また，あらかじめ TP1/Server で起動しているマルチスケジューラデーモンのプロセス数を，TP1/Client/J 環境定義の dcscdmulticount オペランドに指定しておく必要があります。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は，次に示す範囲の値からランダムに選択されます。

- 下限値：TP1/Client/J 環境定義の dchost オペランド，または dcscdport オペランドに指定したポート番号の値
- 上限値：下限値 ＋ TP1/Client/J 環境定義の dcscdmulticount オペランドに指定したプロセス数 - 1

ただし，TP1/Client/J 環境定義の dchost オペランド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。

(b) ネームサービスを使用した RPC

マルチスケジューラ機能を使用して，ネームサービスを使用した RPC 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ときは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) TP1/Client/J 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 TP1/Client/J 環境定義の指定によって異なります。

スケジューラダイレクト機能を使用した RPC の場合の，TP1/Client/J 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 TP1/Client/J 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（スケジューラダイレクト機能を使用した RPC）

TP1/Client/J 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
dcscddirect	dcscdmulti	dcscdmulticount	
Y	Y	○	ランダムに選択したマルチスケジューラデーモン※
		－	TP1/Client/J 環境定義の dchost オペランド，または dcscdport オペランド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
	N	無効	TP1/Client/J 環境定義の dchost オペランド，または dcscdport オペランド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

（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

－：オペランドに値を指定しない

注※

マルチ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されます。

下限値：TP1/Client/J 環境定義の dchost オペランド，または dcscdport オペランドに指定したポート番号の値

上限値：下限値 ＋ TP1/Client/J 環境定義の dcscdmulticount オペランドに指定したプロセス数 - 1

ネームサービスを使用した RPC の場合の，TP1/Client/J 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 2-2　TP1/Client/J 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（ネームサービスを使用した RPC）

TP1/Client/J 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
dcnamuse	dcscdmulti	dcscdmulticount	
Y	Y	無効	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※
	N		マスタスケジューラデーモン※

（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

注※

ネームサービスへの問い合わせが発生します。

2.3　トランザクション制御

この機能は，リモート API 機能を使用する場合で，要求先 TP1/Server Base のバージョンが 05-00 以降のときだけ使用できます。

CUP からトランザクションを制御するメソッドを呼び出せます。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に atomic_update=Y を指定しておく必要があります。

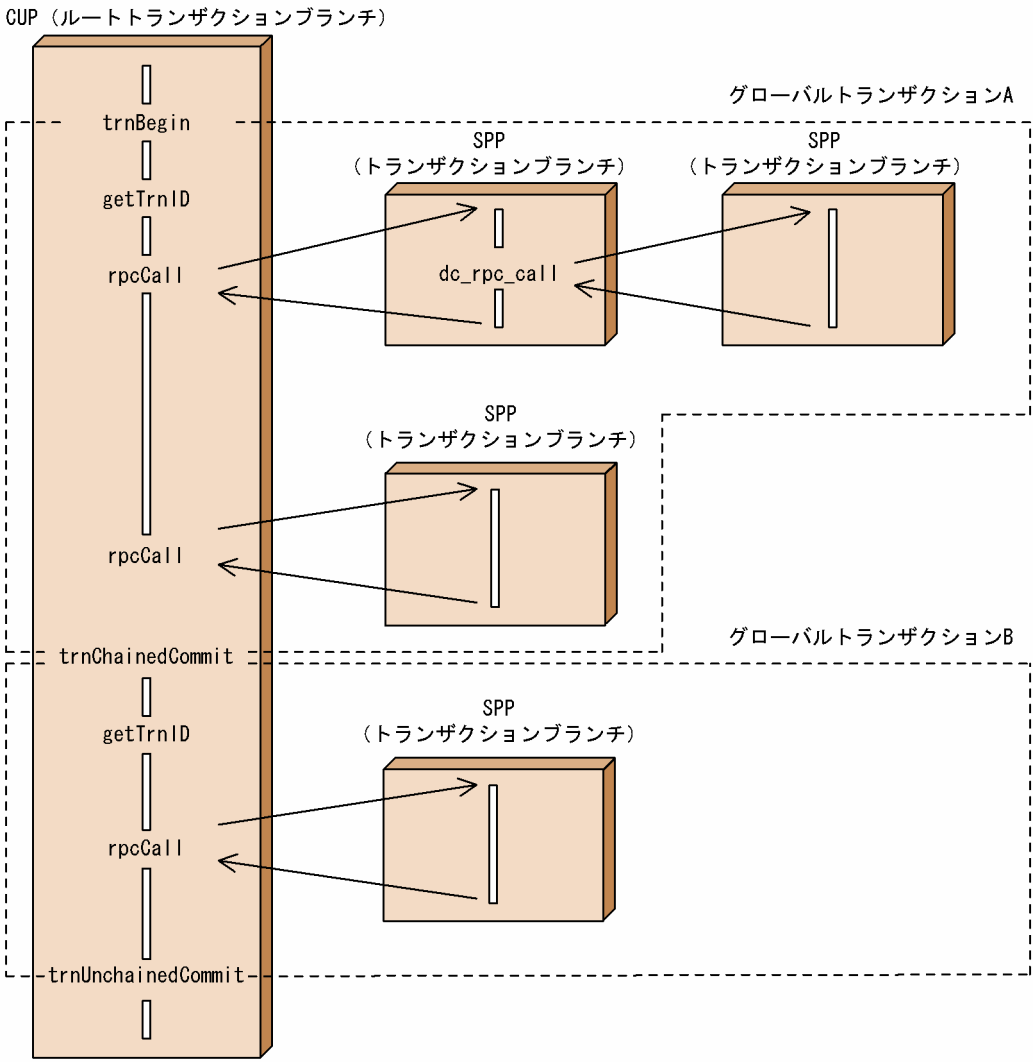
トランザクション制御の詳細については，マニュアル「OpenTP1 プログラム作成の手引」を参照してください。

2.3.1　トランザクションの開始と同期点取得

CUP から trnBegin メソッドを呼び出して，トランザクションを開始します。

trnBegin メソッドを呼び出してから，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です。trnBegin メソッドを呼び出したあと，そのグローバルトランザクションの中で新たな trnBegin メソッドは呼び出せません。CUP から SPP へ RPC を実行すると，CUP は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。TP1/Client/J 環境定義の dcrapautoconnect オペランドに Y を指定した場合のトランザクションと RPC の関係を次の図に示します。

図 2-11　トランザクションと RPC の関係（dcrapautoconnect オペランドに Y を指定した場合）



2.3.2　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メソッド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

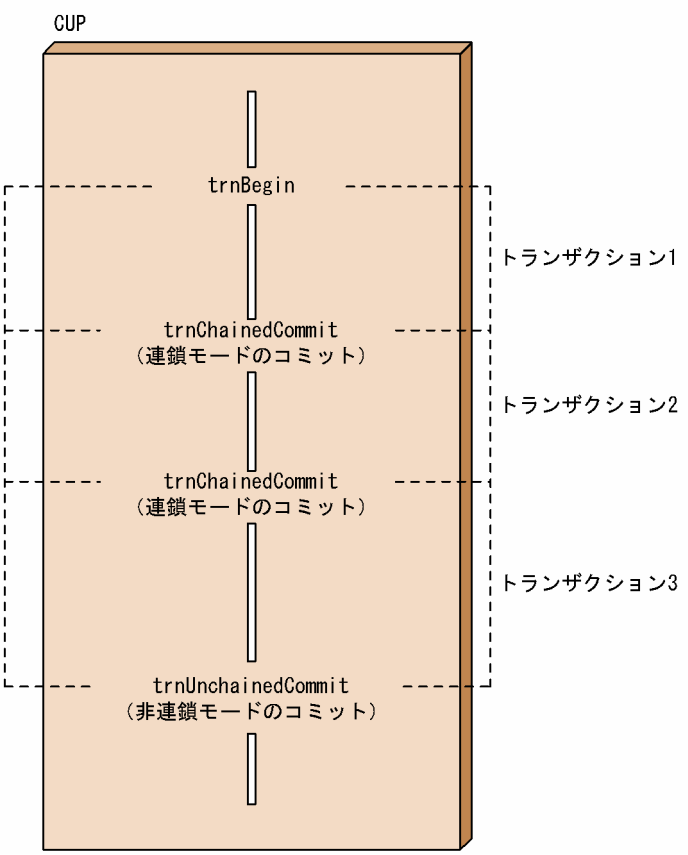
トランザクション処理の同期点取得には，一つのトランザクションの終了後，同期点を取得して次のトランザクションを続けて起動する**連鎖モードのコミット**と，トランザクションの終了で同期点を取得したあと，新たなトランザクションを起動しない**非連鎖モードのコミット**があります。

連鎖モードのコミットは，trnChainedCommit メソッドを呼び出して要求します。

非連鎖モードのコミットは，trnUnchainedCommit メソッドを呼び出して要求します。

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 2-12　トランザクションの連鎖，非連鎖モード



(b)　コミット要求のメソッドを呼び出さない場合の処理

コミット要求のメソッドを呼び出さないで CUP が終了したとき，またはコミット要求のメソッドを呼び出す前に CUP が異常終了したときは，トランザクションはロールバックされます。

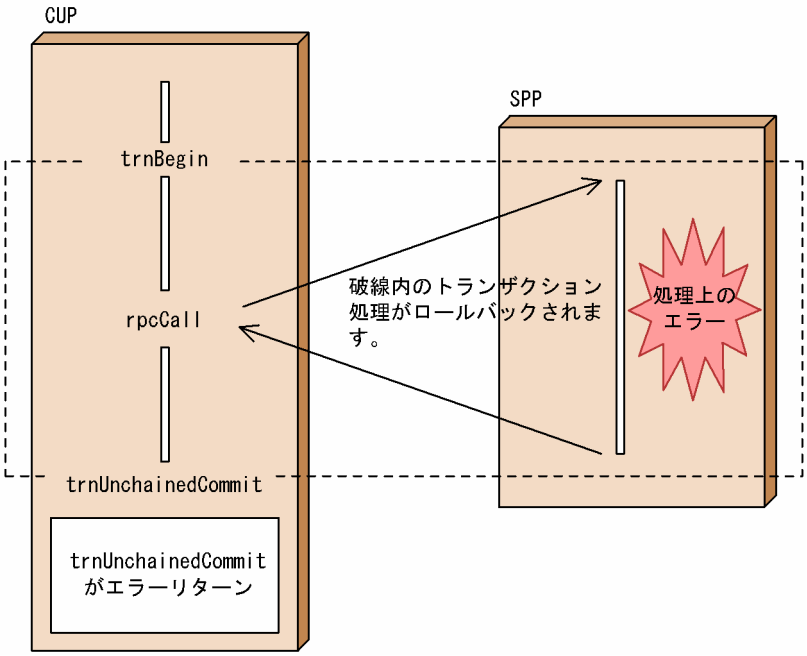
(2)　ロールバック

(a)　TP1/Server の処理がエラーの場合

トランザクションでエラーが発生すると，コミット要求のメソッド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 2-13　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メソッドの呼び出し

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メソッドを呼び出します。

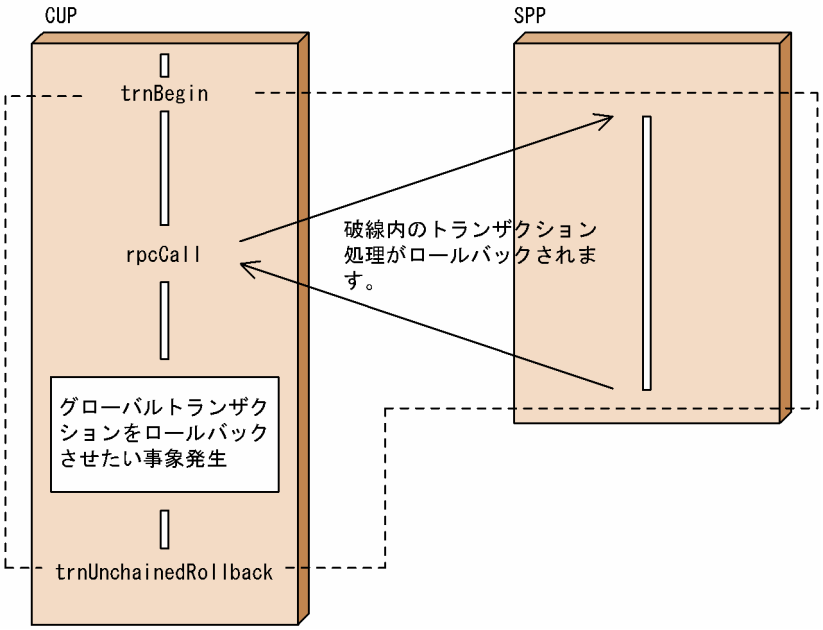
ロールバックには，**連鎖モードのロールバック**と**非連鎖モードのロールバック**があります。

連鎖モードのロールバックは，**trnChainedRollback メソッド**を呼び出して要求します。
trnChainedRollback メソッドを呼び出してロールバックすると，このメソッドを呼び出した CUP のプロセスは，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，**trnUnchainedRollback メソッド**を呼び出して要求します。
trnUnchainedRollback メソッドを呼び出してロールバックすると，このメソッドを呼び出した CUP のプロセスは，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。

ロールバック要求のメソッドを呼び出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 2-14　トランザクションのロールバック（ロールバック要求のメソッドを呼び出した場合）



(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す。この場合の例外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 … ErrHeuristicException
- 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 … ErrHazardException

これらの例外が返される原因，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，TP1/Server の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 TP1/Client/J 環境定義で指定できます。詳細については「5.2　TP1/Client/J 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監視時間に，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を含むかどうか
- 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

(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，TP1/Client/J 環境定義で指定できます。詳細については「5.2　TP1/Client/J 環境定義の詳細」を参照してください。

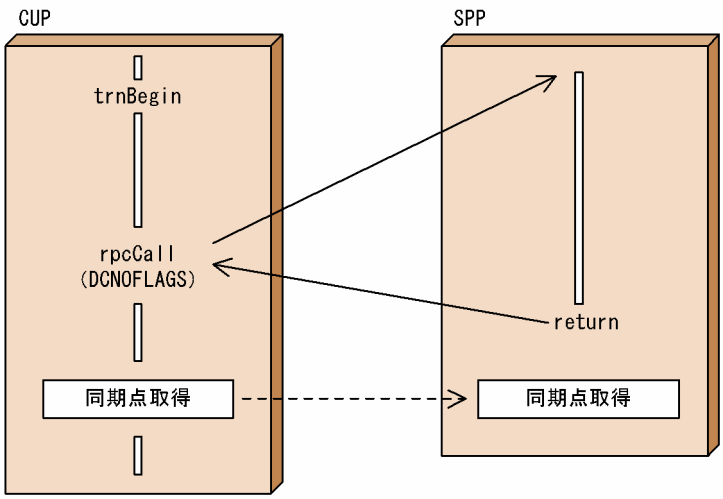
2.3.3　リモートプロシジャコールの形態と同期点の関係

(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 2-15　同期応答型 RPC と同期点の関係

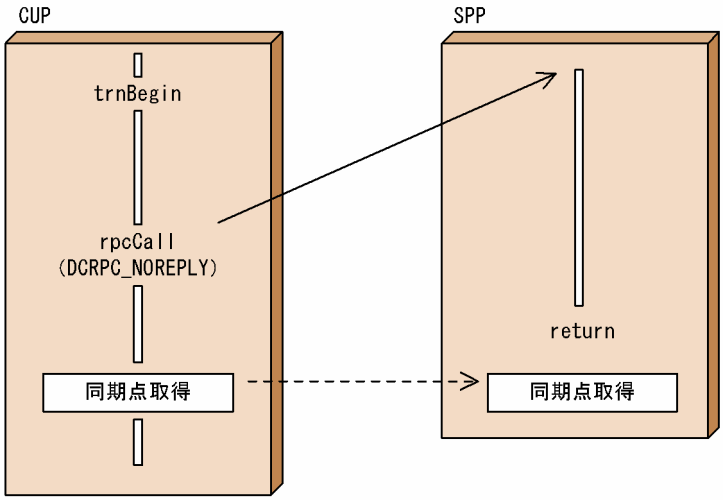


(2)　非応答型 RPC と同期点の関係

非応答型 RPC のトランザクション処理の場合は，CUP と SPP の処理終了で同期を取ります。

非応答型 RPC と同期点の関係を次の図に示します。

図 2-16　非応答型 RPC と同期点の関係



2.3.4　現在のトランザクションに関する識別子の取得

CUP から getTrnID メソッドを呼び出すと，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，次に示すメソッドを呼び出したあとには，必ず getTrnID メソッドを呼び出してください。

- trnBegin メソッド
- trnChainedCommit メソッド
- trnChainedRollback メソッド

2.3.5　現在のトランザクションに関する情報の報告

CUP から trnInfo メソッドを呼び出すと，TP1Client オブジェクトがトランザクションとして稼働中かどうかを，戻り値で確認できます。

2.3.6　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

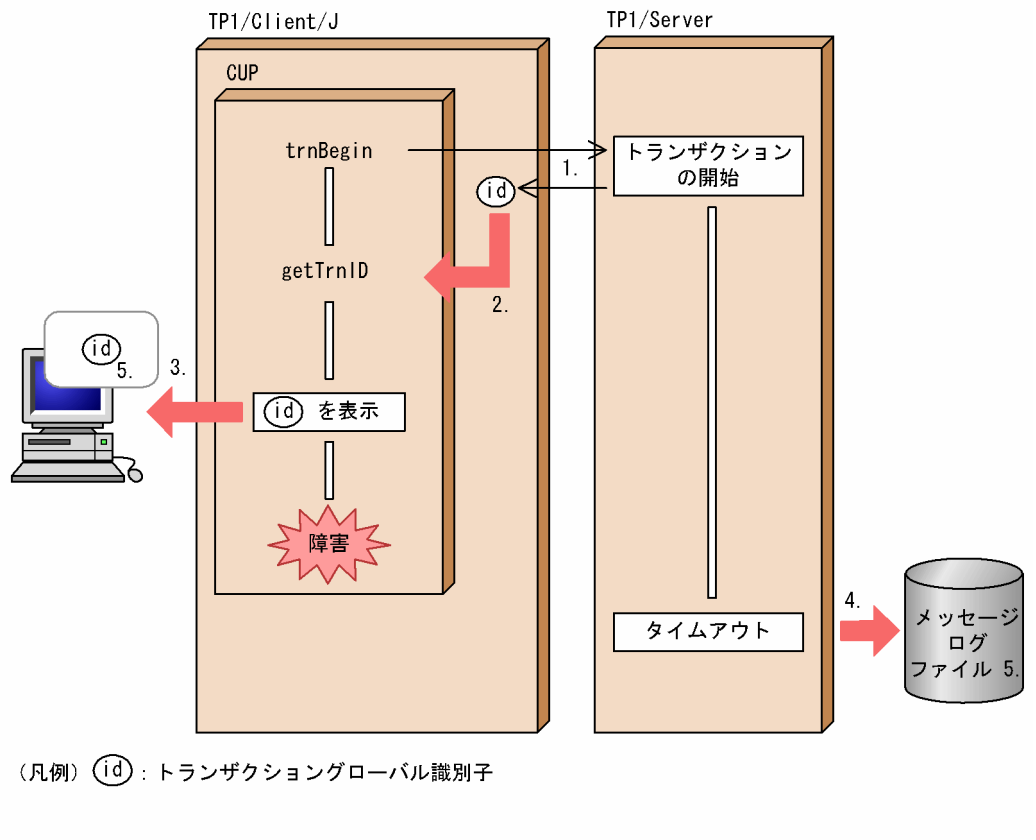
CUP から開始したトランザクションで障害が発生した場合，そのトランザクションブランチがコミットされたかどうかを検証できます。この場合，トランザクション開始後に，必ず getTrnID メソッドを呼び出して，現在のトランザクショングローバル識別子，およびトランザクションブランチ識別子を取得しておく必要があります。

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，TP1/Server 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と，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。

TP1/Server の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



1. CUP から trnBegin メソッドを呼び出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/J に通知します。
2. CUP は getTrnID メソッドを呼び出して，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 TP1/Server の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。

2.3.7　TP1/Server 側の定義の指定

TP1/Client/J と TP1/Server 間でトランザクション連携をする場合，TP1/Server の定義に rpc_extend_function オペランドを指定するときは次のとおりにしてください。

- ユーザサービスデフォルト定義の rpc_extend_function オペランドは，00000002 ビットを ON にしないでください。00000002 ビットが ON になっている場合，動作を保証できません。
- ユーザサービスデフォルト定義の rpc_extend_function オペランドで 00000002 ビットが ON になっている場合，rap リスナーサービス定義の rpc_extend_function オペランドで 00000002 ビットが OFF になるように定義し，rapdfgen コマンドで rap リスナー用ユーザサービス定義，および rap サーバ用ユーザサービス定義を再作成してください。再作成後，rap リスナー，および rap サーバを再起動してください。

2.4　TCP/IP 通信機能

TP1/Client/J の TCP/IP 通信機能は，CUP がクライアント型かサーバ型かによって通信方法が異なります。CUP がクライアント型の場合は，CUP から定義や引数で指定した通信相手に対してコネクションを確立して通信します。CUP がサーバ型の場合は，定義で指定したポートで通信相手がコネクションを確立してくるのを待って通信します。コネクション確立後は，引数で指定した領域に対してデータの送受信を行います。

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」と呼んでいますが，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

2.4.1　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。

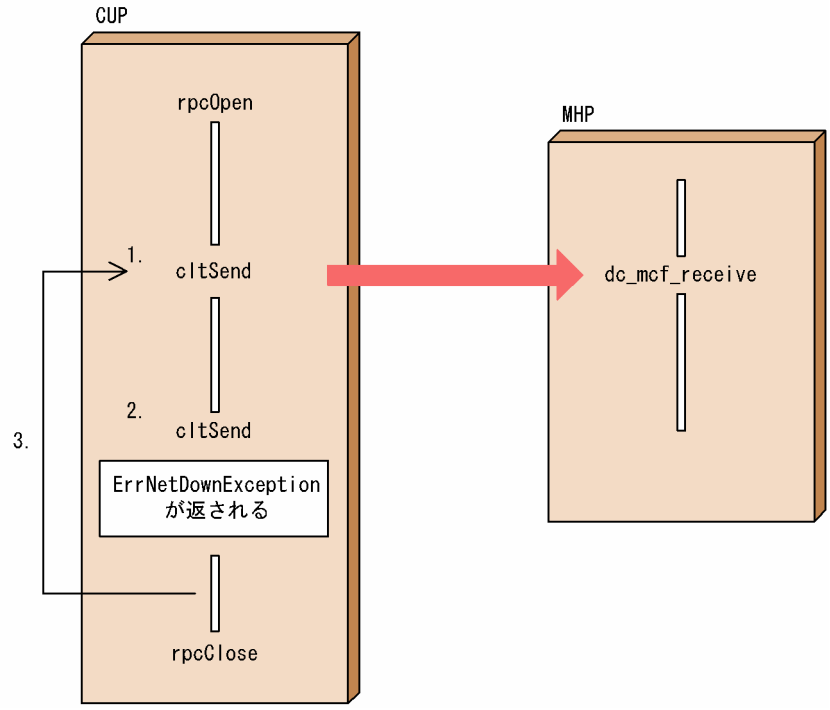
CUP から **cltSend メソッド**を実行して，MHP へメッセージを送信します。メッセージの組み立て機能を使用する場合は，**cltAssemSend メソッド**を実行します。

メッセージを一方送信するには，次に示す指定をしておく必要があります。

- 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) cltSend メソッド，または cltAssemSend メソッドの引数 hostname に指定
  (b) TP1/Client/J 環境定義 dcsndhost オペランドに指定
- 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 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。
  (a) cltSend メソッド，または cltAssemSend メソッドの引数 portnum に指定
  (b) TP1/Client/J 環境定義 dcsndport オペランドに指定
- TP1/Client/J 環境定義 dcsndrcvtype オペランドに DCCLT_ONEWAY_SND を指定します。

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



1. MHP が起動されたあと，CUP を起動し，cltSend メソッドを実行します。
2. cltSend メソッドを発行したときに，MHP からコネクションが解放されていた場合，CUP に例外 ErrNetDownException を返します。
3. 再びメッセージを一方送信するには，cltSend メソッドを実行します。

2.4.2　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。

CUP は，**cltReceive メソッド**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を使用する場合は，**cltAssemReceive メソッド**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。

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/J 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。タイムアウトやエラーが発生したときに，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。

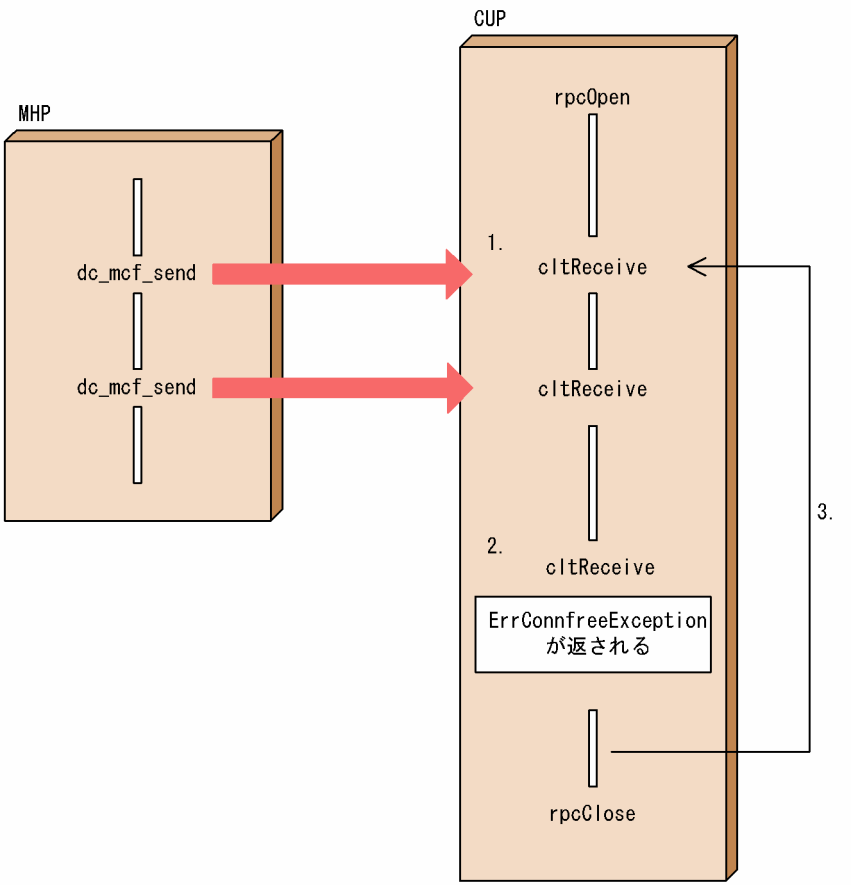
メッセージを一方受信するには，あらかじめ，TP1/Client/J 環境定義で次に示す指定をしておく必要があります。

- dcrcvport オペランドに CUP のポート番号（MCF 通信構成定義の定義コマンド mcftalccn の oportno オペランドで指定したポート番号）を指定
- dcsndrcvtype オペランドに DCCLT_ONEWAY_RCV を指定

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，TP1/Client/J 環境定義 dcrcvport オペランドを 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 TP1/Client/J 環境定義の設定方法については，「5.2.3　TP1/Client/J 環境定義を指定するときの注意事項」を参照してください。受信用ソケットの開設は，rpcOpen メソッド実行時に行います。

メッセージの一方受信を，次の図に示します。

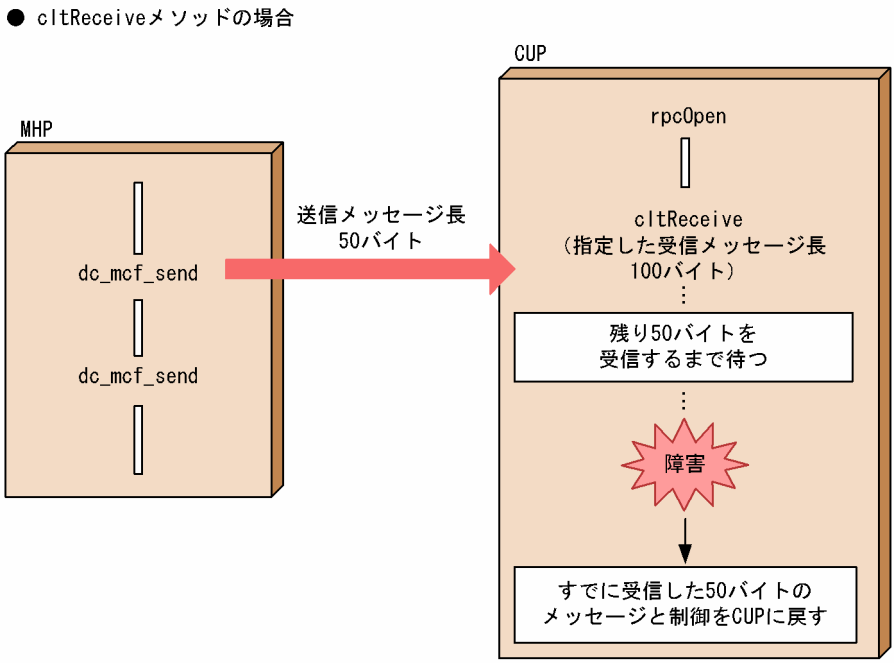
図 2-19　メッセージの一方受信



1. MHP がコネクションの確立要求を再試行している間に CUP を起動し，cltReceive メソッドを実行します。再試行中にコネクションを確立できなかった場合，mcftactcn コマンドを入力してコネクションを確立します。
2. MHP からコネクションが解放された場合，CUP に例外 ErrConnfreeException を返します。
3. 再びメッセージを一方受信するには，cltReceive メソッドを実行します。この場合，mcftactcn コマンドを入力して，コネクション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）



2.4.3　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は，コネクションが解放されていなければ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 cltSend メソッドを実行して MHP へメッセージを送信し，cltReceive メソッドを実行して，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を使用する場合は，CUP から cltAssemSend メソッドを実行して MHP へメッセージを送信し，cltAssemReceive メソッドを実行して 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 cltReceive メソッドを実行して，MHP からのメッセージを受信し，cltSend メソッドを実行して 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を使用する場合は，CUP で cltAssemReceive メソッドを実行して MHP からのメッセージを受信し，cltAssemSend メソッドを実行して 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。TP1/Client/J 環境定義の各オペランドの詳細は「5.2.2　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) cltSend メソッド，または cltAssemSend メソッドの引数 hostname に指定

  (b) TP1/Client/J 環境定義 dcsndhost オペランドに指定

- 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。

  (a) cltSend メソッド，または cltAssemSend メソッドの引数 portnum に指定

  (b) TP1/Client/J 環境定義 dcsndport オペランドに指定

- TP1/Client/J 環境定義 dcsndrcvtype オペランドに DCCLT_SNDRCV を指定します。

- TP1/Client/J 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ないため，TP1/Client/J 環境定義 dcsockopenatrcv 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。

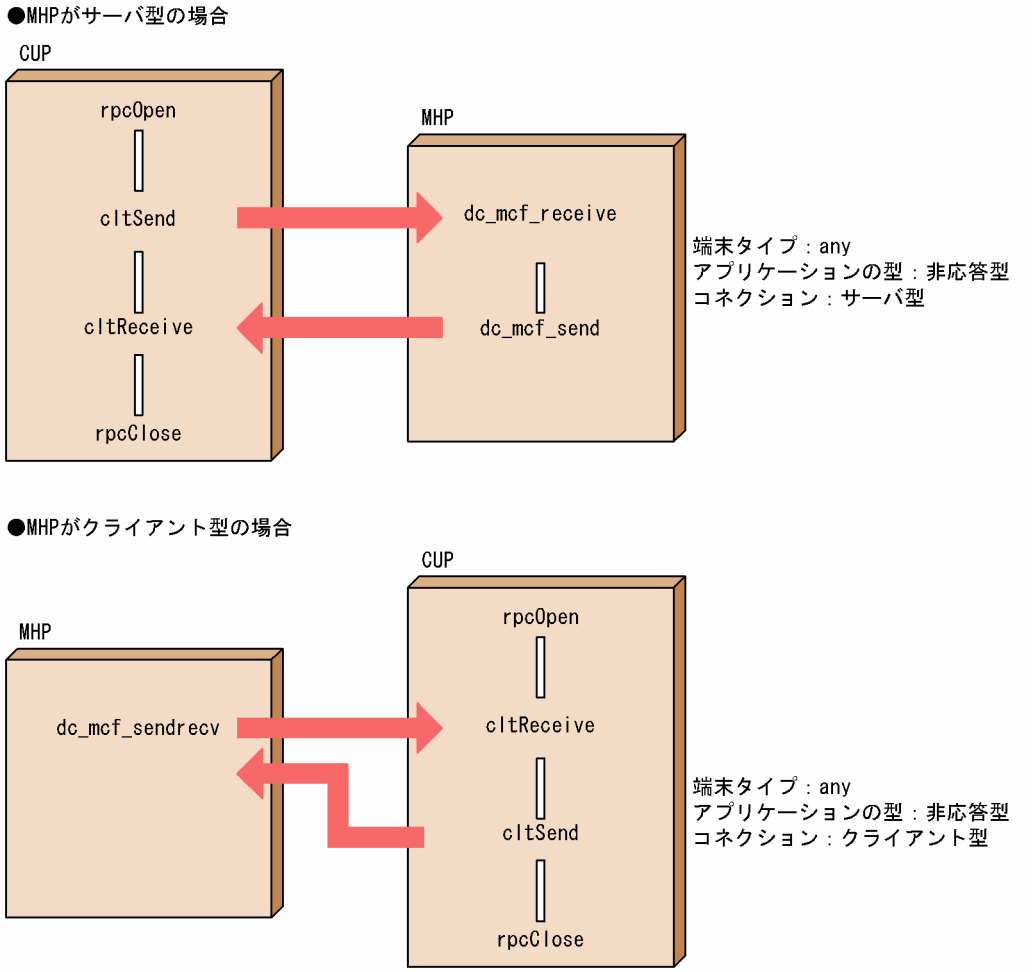
MHP がクライアント型の場合

- TP1/Client/J 環境定義 dcrcvport オペランドに 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，TP1/Client/J 環境定義 dcrcvport オペランドを 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 TP1/Client/J 環境定義の設定方法については，「5.2.3　TP1/Client/J 環境定義を指定するときの注意事項」を参照してください。

- TP1/Client/J 環境定義 dcsndrcvtype オペランドに DCCLT_SNDRCV を指定します。

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

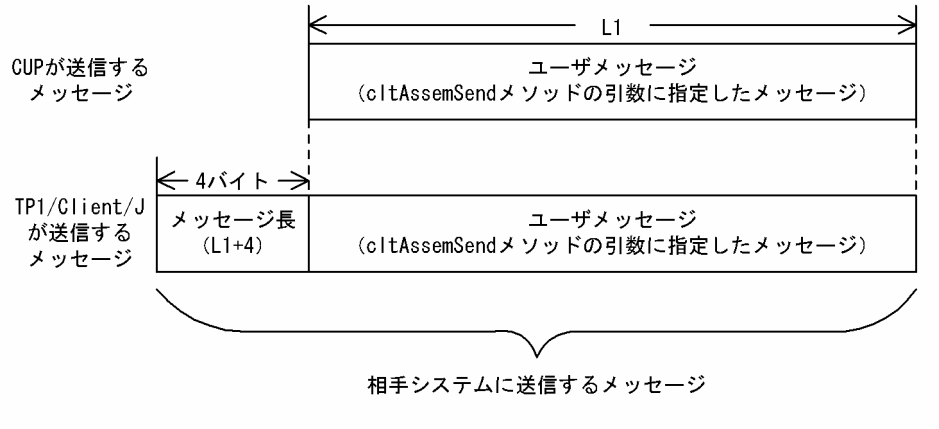


2.4.4　受信メッセージの組み立て機能

受信メッセージの組み立て機能は，送受信メッセージの先頭 4 バイトをメッセージ長エリアにして TCP/IP 通信をします。通信相手が TP1/NET/TCP/IP の受信メッセージの組み立て機能を使用している場合にこの機能を使用すると，CUP でメッセージ長エリアを意識しないで通信できます。

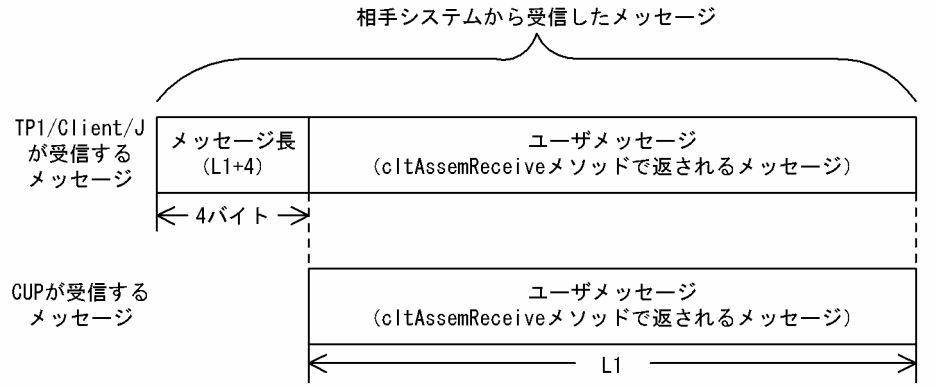
(1)　メッセージ送信

CUP が cltAssemSend メソッドでメッセージを送信すると，TP1/Client/J はメッセージの先頭に 4 バイトのメッセージ長エリアを付加して相手システムに送信します。TP1/Client/J が付加するメッセージ長は，ネットワークバイトオーダーで設定します。cltAssemSend メソッドを使用するには，TP1/Client/J 環境定義の dcsndrcvtype オペランドに DCCLT_ONEWAY_SND または DCCLT_SNDRCV を指定してください。



(2)　メッセージ受信

CUP が cltAssemReceive メソッドでメッセージの受信要求をすると，TP1/Client/J は受信したメッセージの先頭 4 バイトをメッセージ長として，メッセージを組み立てまたは分割して CUP に通知します。メッセージ長として設定されている値は，ネットワークバイトオーダーとして処理します。送信元の相手システムは，メッセージ長をネットワークバイトオーダーで設定してください。cltAssemReceive メソッドを使用するには，TP1/Client/J 環境定義の dcsndrcvtype オペランドに DCCLT_ONEWAY_RCV または DCCLT_SNDRCV を指定してください。



2.4.5　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 2-3 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 2-4 に示します。

表 2-3　TCP/IP 通信機能のユースケース

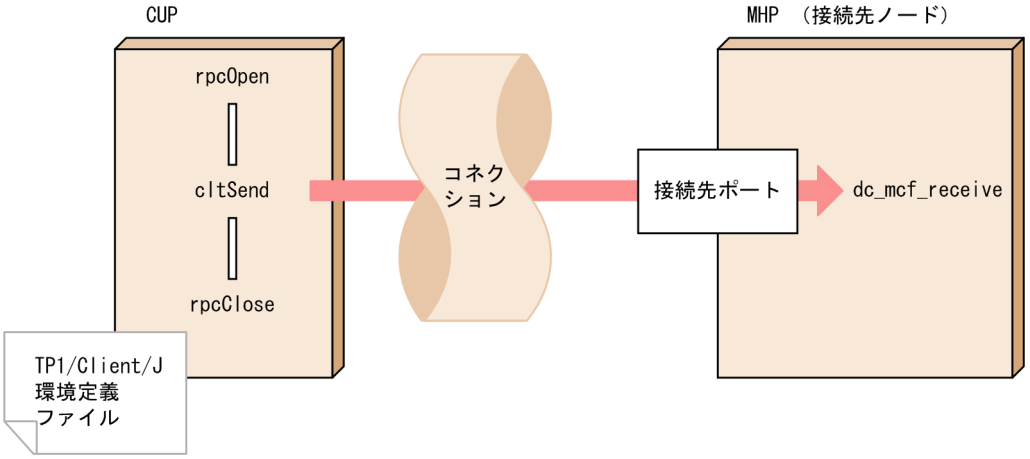
項番	ユースケース
1	MHP がサーバ型でメッセージの一方送信を行う場合
2	MHP がクライアント型でメッセージの一方受信を行う場合
3	MHP がサーバ型でメッセージの送受信を一つのコネクションで行う場合

項番	ユースケース
4	MHP がクライアント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

(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 2-22　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。

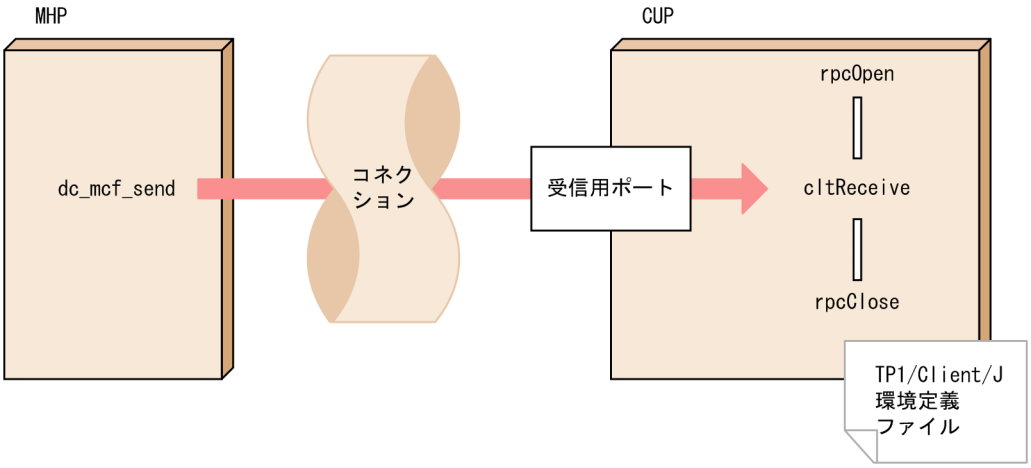
- 接続先のノード名を次のどちらかの方法で指定します。
  (a) cltSend メソッド，または cltAssemSend メソッドの引数 hostname に指定
  (b) TP1/Client/J 環境定義 dcsndhost オペランドに指定
- 接続先のポート番号を次のどちらかの方法で指定します。
  (a) cltSend メソッド，または cltAssemSend メソッドの引数 portnum に指定
  (b) TP1/Client/J 環境定義 dcsndport オペランドに指定
- TP1/Client/J 環境定義 dcsndrcvtype オペランドに DCCLT_ONEWAY_SND 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 TP1/Client/J 環境定義の設定方法については，「5.2.3　TP1/Client/J 環境定義を指定するときの注意事項」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 2-23　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。

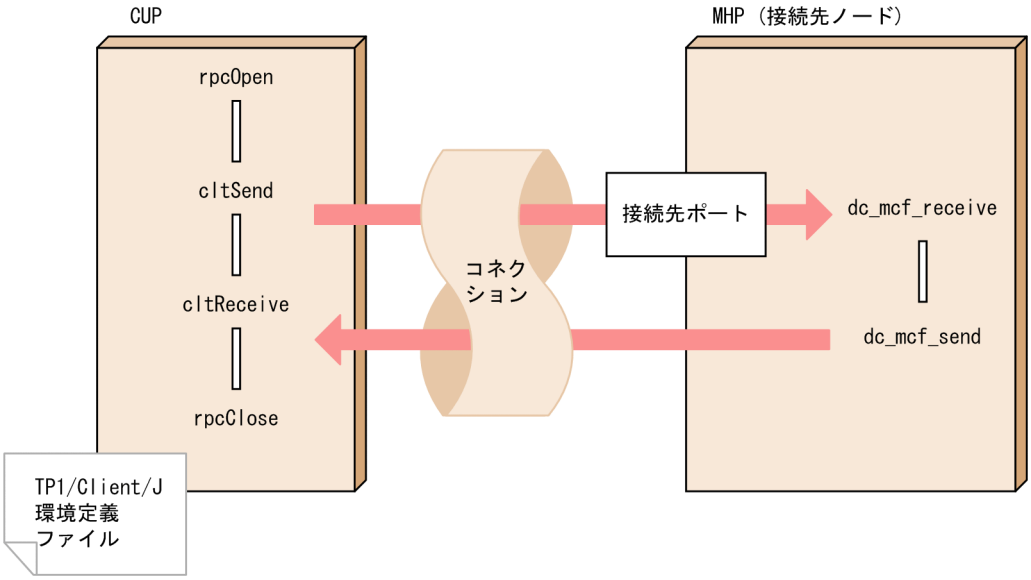
- TP1/Client/J 環境定義 dcrcvport オペランドに，CUP の受信用ポート番号（MCF 通信構成定義の定義コマンド mcftalccn の oportno で指定したポート番号）を指定します。
- TP1/Client/J 環境定義 dcsndrcvtype オペランドに DCCLT_ONEWAY_RCV を指定します。
- TP1/Client/J 環境定義 dcsockopenatrcv オペランドに N を指定，または指定を省略します。

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，TP1/Client/J 環境定義 dcrcvport オペランドを 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 TP1/Client/J 環境定義の設定方法については，「5.2.3　TP1/Client/J 環境定義を指定するときの注意事項」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 2-24　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，CUP から接続先のポートに対してコネクションを確立し，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，TP1/Client/J 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コネクションで行う場合は，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) cltSend メソッド，または cltAssemSend メソッドの引数 hostname に指定
  (b) TP1/Client/J 環境定義 dcsndhost オペランドに指定
- 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) cltSend メソッド，または cltAssemSend メソッドの引数 portnum に指定
  (b) TP1/Client/J 環境定義 dcsndport オペランドに指定
- TP1/Client/J 環境定義 dcsockopenatrcv オペランドに Y を指定します。
- TP1/Client/J 環境定義 dcsndrcvtype オペランドに DCCLT_SNDRCV を指定します。
- cltSend メソッド，または cltAssemSend メソッドの引数 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 TP1/Client/J 環境定義の設定方法については，「5.2.3　TP1/Client/J 環境定義を指定するときの注意事項」を参照してください。

注意事項

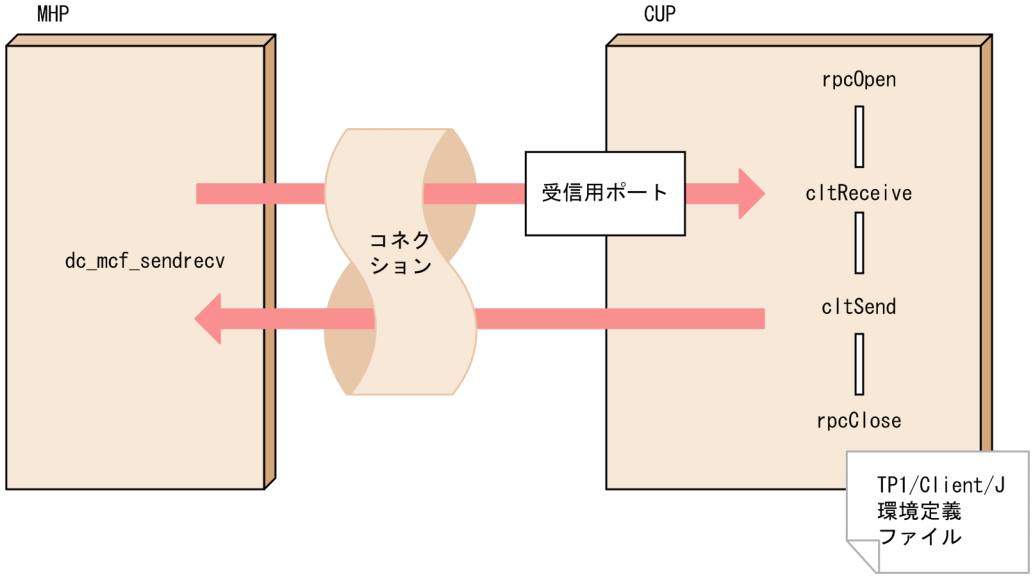
何らかの障害が発生しコネクションが切断された場合，cltReceive メソッド，および cltAssemReceive メソッドを実行しないでください。コネクションの切断を検知した場合は，

cltSend メソッド，または cltAssemSend メソッドを再実行し，コネクション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コネクションで行う場合

MHP がクライアント型でメッセージの送受信を一つのコネクションで行う場合を次の図に示します。

図 2-25 MHP がクライアント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 がクライアント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 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，そのコネクションを使用してメッセージを送受信します。

MHP がクライアント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。

- TP1/Client/J 環境定義 dcrcvport オペランドに，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。
- TP1/Client/J 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。
- TP1/Client/J 環境定義 dcsndrcvtype オペランドに DCCLT_SNDRCV を指定します。
- cltReceive メソッド，または cltAssemReceive メソッドの引数 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，TP1/Client/J 環境定義 dcrcvport オペランドを 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 TP1/Client/J 環境定義の設定方法については，「5.2.3 TP1/Client/J 環境定義を指定するときの注意事項」を参照してください。

表 2-4　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	TP1/Client/J 環境定義のオペランド					提供 API の発行順序※2	CUP の使用ポート番号	
		dcsockopenatrcv	dcsndhost※1	dcsndport※1	dcrcvport	dcsndrcvtype		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，「(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_ONEWAY_SND	cltSend	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)　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。	N または 省略	−	−	CUP のポート番号	DCCLT_ONEWAY_RCV	cltReceive	使用しない	dcrcvport 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，「(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」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_SNDRCV	cltSend※5 → cltReceive	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コネクションで行う場合 詳細については，「(4)　MHP がクライアント型でメッセージ	N または 省略※4	−	−	CUP のポート番号		cltReceive※6 → cltSend	使用しない	dcrcvport の値

項番	ユースケース	TP1/Client/J 環境定義のオペランド					提供 API の発行順序※2	CUP の使用ポート番号	
		dcsockopenatrcv	dcsndhost※1	dcsndport※1	dcrcvport	dcsndrcvtype		送信	受信
4	の送受信を一つのコネクションで行う場合」を参照してください。	N または 省略※4	−	−	CUP のポート番号	DCCLT_SNDRCV	cltReceive※6 → cltSend	使用しない	dcrcvport の値

注※1

接続先ノード名，およびポート番号は，それぞれ cltSend メソッド，および cltAssemSend メソッドの引数 hostname（ノード名），portnum（ポート番号）に指定できます。

注※2

表中の cltSend メソッドは，cltAssemSend メソッドを含みます。また，cltReceive メソッドは，cltAssemReceive メソッドを含みます。

注※3

dcsockopenatrcv オペランドに N を指定，または省略する設定は，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 とのコネクション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

cltSend メソッドまたは cltAssemSend メソッドの引数 flags には DCNOFLAGS を指定してください。

注※6

cltReceive メソッドまたは cltAssemReceice メソッドの引数 flags には DCNOFLAGS を指定してください。

2.4.6　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場合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/J，および MHP 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。

- ソケットのバッファに TP1/Client/J 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/J 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/J 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/J 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/J，および MHP 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。

- 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/J 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

メッセージは，TCP/IP プロトコルを使用して受信します。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。

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/J 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/J 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/J 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。

2.5　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。

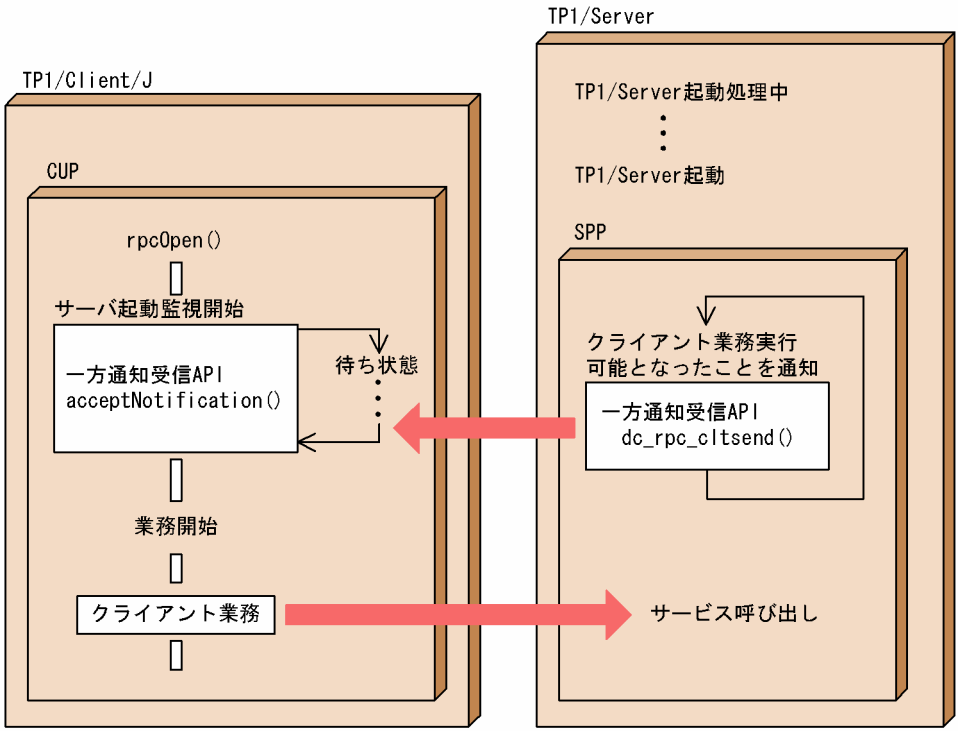
2.5.1　一方通知受信機能の処理の流れ

一方通知受信機能を使用すると，オンライン開始の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。

一方通知受信機能を使用する場合，acceptNotification メソッドを実行します。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メソッド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。

一方通知受信機能の処理の流れを次の図に示します。

図 2-26　一方通知受信機能の処理の流れ



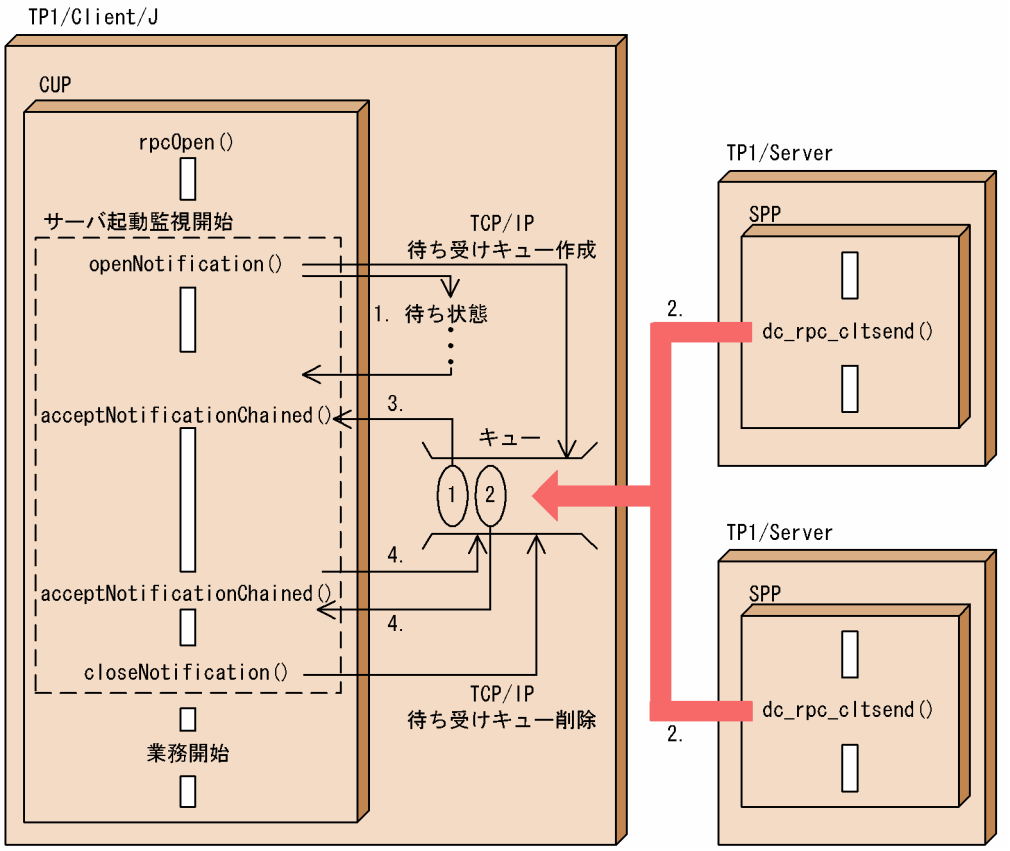
2.5.2　一方通知連続受信機能の処理の流れ

一方通知連続受信機能を使用すると，openNotification メソッドを実行してから closeNotification メソッド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 acceptNotificationChained メソッドを実行した時点で，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 2-27　一方通知連続受信機能の処理の流れ



1. 一方通知メッセージが送られてくるのを待ちます。
2. TP1/Server が起動し，クライアント業務が実行できる状態になったことを，一方通知メッセージを送信して通知します。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。
3. 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。
4. acceptNotificationChained メソッドの発行時に，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。

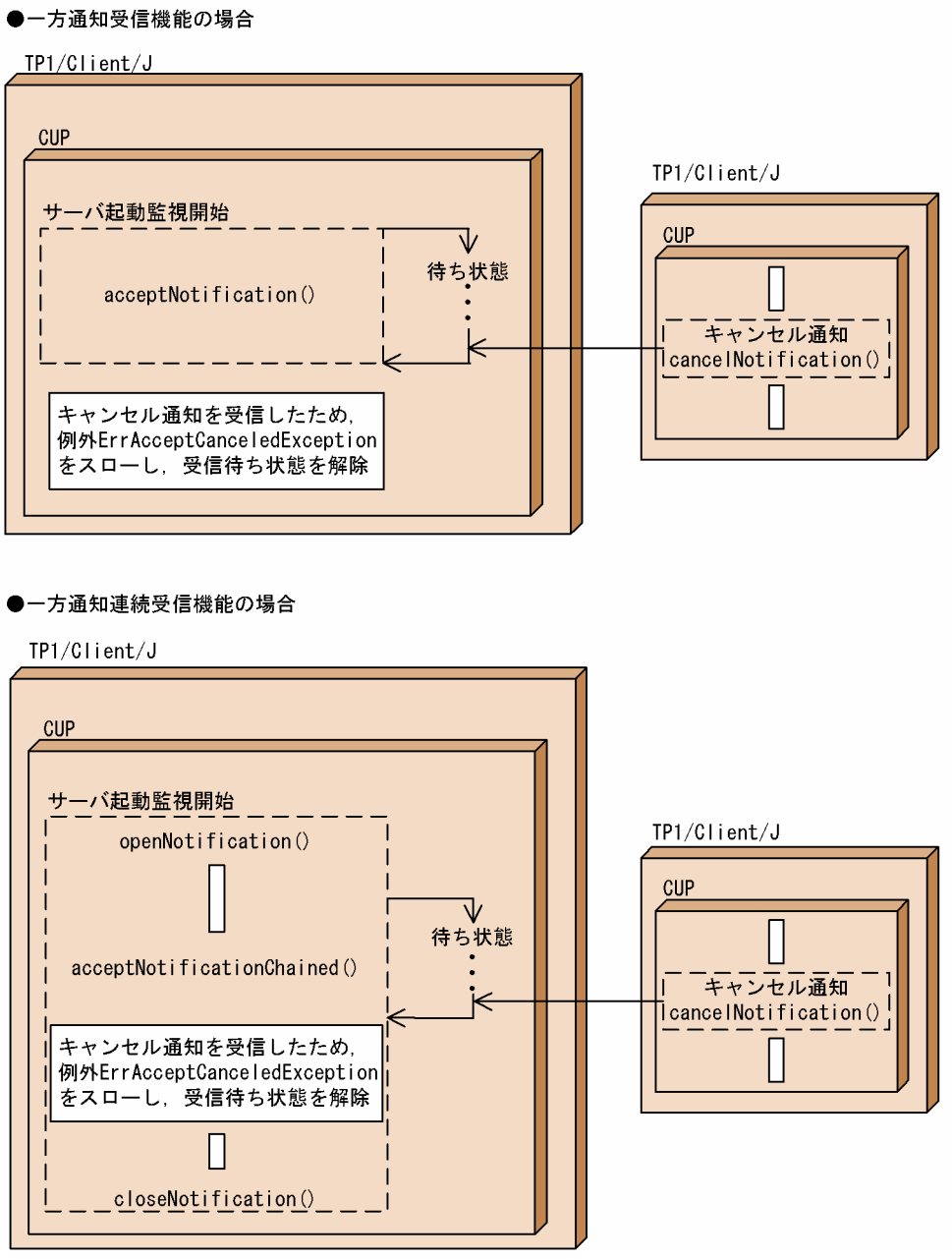
2.5.3　一方通知連続受信機能を使用するときの注意事項

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，JavaVM の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。

2.5.4　一方通知受信待ち状態の解除

acceptNotification メソッドまたは acceptNotificationChained メソッドを発行し，サーバからの一方通知受信待ち状態となった CUP が，別の CUP からキャンセル通知を受信した場合，一方通知受信待ち状態を解除します。キャンセル通知は，cancelNotification メソッドを発行することで送信できます。一方通知受信待ち状態を解除する処理の流れを次の図に示します。

図 2-28　一方通知受信待ち状態を解除する処理の流れ



2.6 TP1/Web 接続機能

TP1/Web 接続機能は，Java アプレットとして作成した CUP と TP1/Web を接続して，HTTP プロトコルに基づいた通信サービスをするための機能です。

TP1/Web と接続することによって，HTTP プロトコルでの通信では本来できなかったセションの関連づけができるようになり，連続した対話的業務を行えます。

TP1/Web と接続するには，**openConnection メソッド**を使用します。openConnection メソッドの詳細については，「4. TP1/Client/J で使用するクラス」を参照してください。

なお，TP1/Web 接続機能を使用した場合，トランザクション制御機能は使用できません。また，TP1/Client/J 環境定義の dccltrpcmaxmsgsize オペランドを指定して TP1/Web 接続機能を使用した場合，通信先の TP1/Server ノードでエラーが発生することがあります。

2.6.1 TP1/Web との接続（セションの開始）

TP1/Web との接続（セション）は，openConnection(String url, short flags)メソッドを呼び出すことによって開始されます。セションは，closeConnection()を呼び出すまで継続されます。

セションを開始する方法には，次に示す 3 種類があります。

- TP1/Client/J 環境定義で dcweburl オペランドにサービス要求先の URL を，dcrapautoconnect オペランドに Y を指定して，rpcOpen メソッドを呼び出したあとで，rpcCall メソッドを呼び出す。TP1/Web とのセションは，TP1/Client/J が自動的に確立する。
- TP1/Client/J 環境定義で dcweburl オペランドにサービス要求先の URL を指定して，rpcOpen メソッドを呼び出したあとで，引数なしの openConnection メソッドを呼び出す。
- TP1/Client/J の API の openConnection(String url, short flags)メソッドを呼び出す。

2.6.2 TP1/Web へのサービス要求

TP1/Web へのサービス要求は，rpcCall メソッドを呼び出すことで行われます。

TP1/Web の Java to SPP ゲートウェイ機能（DC_JGW 機能）を使用する場合は，TP1/Web が，rpcCall メソッドで送信されたメッセージを基に，バックエンドの OpenTP1 上の SPP に対してサービス要求を行います。OpenTP1 の SPP から応答されたメッセージは TP1/Web から HTTP プロトコルで応答します。

TP1/Web の Java to ユーザサービス機能（DC_JUSR 機能）を使用する場合は，TP1/Web 上で動作しているユーザサービスを呼び出せます。TP1/Web 上で動作しているユーザサービスでは，ユーザが自由に処理を記述できます。応答は TP1/Web が HTTP プロトコルで返します。

TP1/Client/J が提供する TP1/Web 接続機能を使用する場合のサービス要求の形態には，同期応答型および連鎖型の 2 種類があります。ただし，連鎖型は TP1/Web のスタティックセションスケジュール機能を使用し，DC_JGW 機能を使用した場合にだけ使用できます。

同期応答型および連鎖型の詳細については，「2.2.3　RPC の形態」を参照してください。

(1)　同期応答型

TP1/Client/J を使用した CUP（Java アプレット）から TP1/Web に問い合わせメッセージを送信し，応答メッセージを受け取る形態です。同期応答型 RPC では，CUP が TP1/Web からの応答を受け取るまで，それに引き続く処理は待たされます。

(2)　連鎖型

TP1/Client/J を使用した CUP（Java アプレット）から TP1/Web に問い合わせメッセージを送信し，応答メッセージを受け取る形態です。連鎖型 RPC では，CUP が TP1/Web からの応答を受け取るまで，それに引き続く処理は待たされます。連鎖型 RPC を使用する場合は，TP1/Web の設定がスタティックセションスケジュール機能で，かつ，openConnection メソッドで要求した TP1/Web のサービスセットは DC_JGW でなければなりません。

2.6.3　TP1/Web との接続解除（セションの終了）

TP1/Web との接続は，closeConnection メソッドを呼び出すことで解除されます。

closeConnection メソッドを受信した TP1/Web は，接続しているクライアントとのセションを終了し，TP1/Web のリソースを解放します。

2.6.4　TP1/Web と連携した RPC 機能

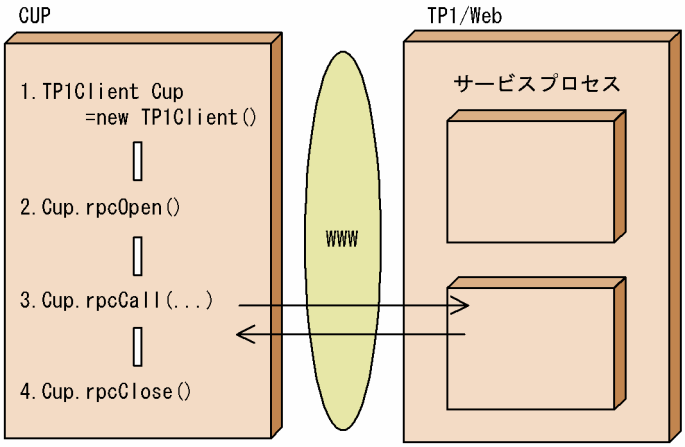
TP1/Client/J から利用できる TP1/Web のサービスセットを次に示します。

- Java to SPP ゲートウェイ機能（DC_JGW 機能）
- Java to ユーザサービス機能（DC_JUSR 機能）

各サービスセットの詳細については，マニュアル「TP1/Web 使用の手引」を参照してください。

TP1/Web と連携した RPC 機能のサービス要求の流れを図 2-29～図 2-33 に示します。

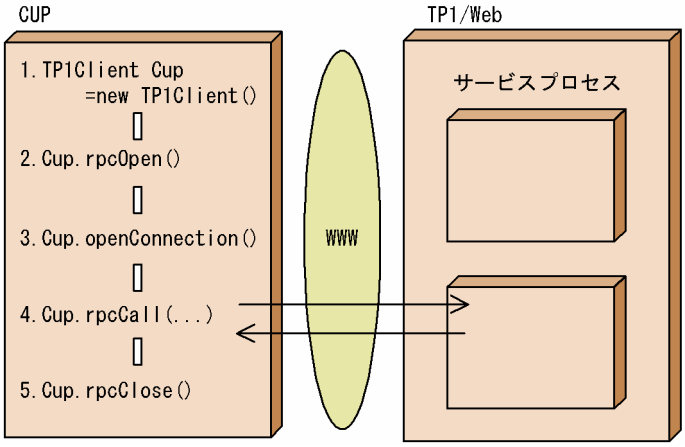
図 2-29　dcrapautoconnect=Y を指定した場合の連携



dcrapautoconnect=Y を指定した場合，次のような流れになります。

1. TP1/Client/J が提供する TP1Client クラスのインスタンスを作成する。
2. rpcOpen メソッドを呼び出して CUP の RPC 環境を初期化する。
3. rpcCall メソッドを呼び出して該当する SPP にサービス要求を行う。
   rpcCall メソッド内部では，次のように処理される。
   dcrapautoconnect=Y なので，まず dcweburl オペランドに定義された TP1/Web の CGI プロセスに対してセションを開始する。次に，サービス要求メッセージを送信する。
4. rpcClose メソッドを呼び出して RPC 環境を解放する（セションを終了する）。

図 2-30　dcrapautoconnect=N を指定し，dcweburl オペランドを定義した場合の連携

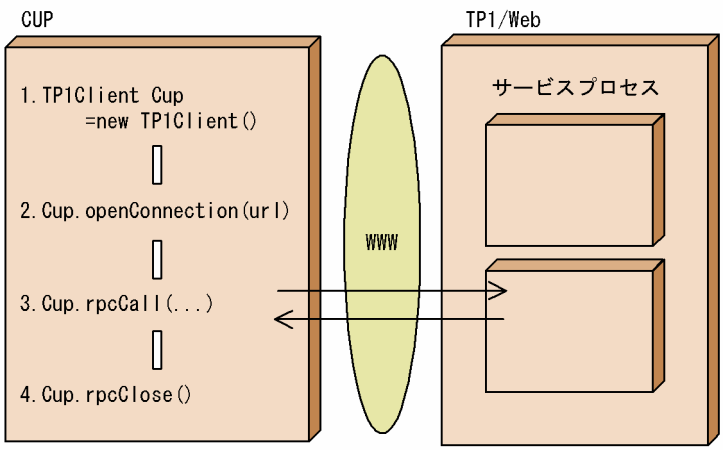


dcrapautoconnect=N を指定し，dcweburl オペランドを定義した場合，次のような流れになります。

1. TP1/Client/J が提供する TP1Client クラスのインスタンスを作成する。
2. rpcOpen メソッドを呼び出して CUP の RPC 環境を初期化する。
3. 引数なしの openConnection メソッドを呼び出して dcweburl で定義された TP1/Web の CGI プロセスに対してセションを開始する。

4. rpcCall メソッドを呼び出して該当する SPP にサービス要求を行う。
5. rpcClose メソッドを呼び出して RPC 環境を解放する（セションを終了する）。

図 2-31　URL を指定した openConnection メソッドを使用する場合の連携



URL を指定した openConnection メソッドを使用する場合，次のような流れになります。

1. TP1/Client/J が提供する TP1Client クラスのインスタンスを作成する。
2. URL を指定した openConnection メソッドを呼び出して TP1/Web の CGI プロセスに対してセションを開始する。
3. rpcCall メソッドを呼び出して該当する SPP にサービス要求を行う。
4. rpcClose メソッドを呼び出して RPC 環境を解放する（セションを終了する）。

図 2-32　DC_JGW サービスセットを使用した場合にサービスが実行される流れ

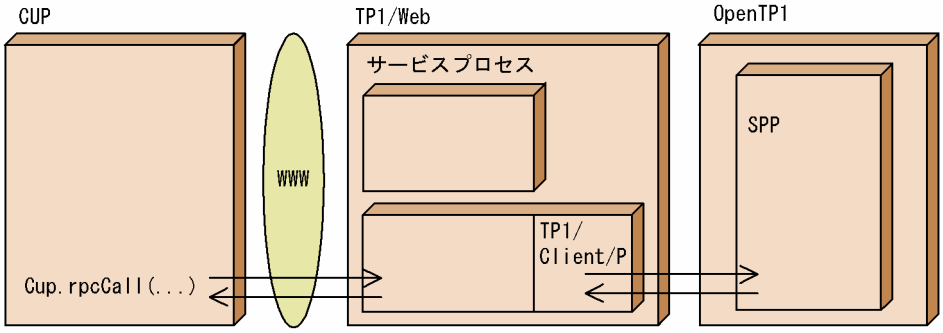
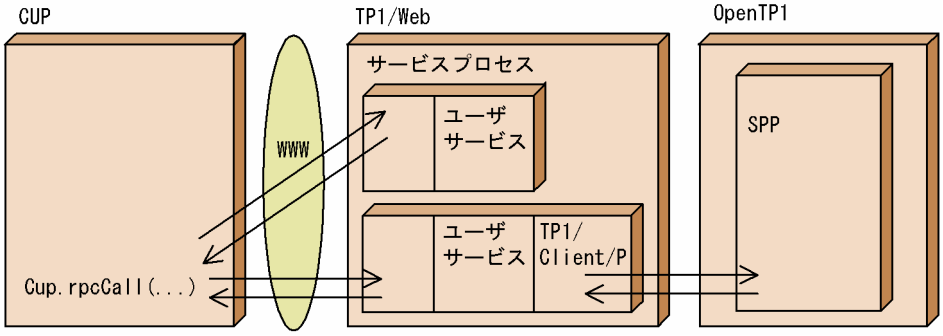


図 2-33　DC_JUSR サービスセットを使用した場合にサービスが実行される流れ



2.7 動的定義変更機能

TP1/Client/J は，TP1/Client/J 環境定義を rpcOpen メソッド内で解析し，CUP に一意の情報として格納しています。

動的定義変更機能を使用すると，TP1/Client/J 環境定義で定義しなかったオペランドの値を設定したり，TP1/Client/J 環境定義で定義したオペランドの値を動的に変更したりできます。なお，動的定義変更メソッドを呼び出して変更したオペランドの値は，CUP が終了するまで（rpcClose メソッドが呼び出されるまで）の間は，同じ動的定義変更メソッドを呼び出してオペランドの値を変更しないかぎり，変更されません。

TP1/Client/J 環境定義の値と，動的定義変更機能で設定した値のどちらが有効になるかは，メソッドの呼び出し順序で決定します。

rpcOpen メソッドを呼び出したあとに動的定義変更メソッドを呼び出した場合

　動的変更メソッドで変更したオペランドの値が有効になります。

動的定義変更メソッドを呼び出したあとに rpcOpen メソッドを呼び出した場合

　TP1/Client/J 環境定義で定義されているオペランドの値が有効になります。

2.8　TCP/IP コネクションの確立の監視機能

TCP/IP コネクションの確立の監視機能では，データ送信時の TCP/IP コネクションの確立処理に対する最大監視時間を指定し，コネクションの確立処理を監視できます。

コネクションの確立処理には，TP1/Client/J の API 内部の JavaAPI である java.net.Socket.connect メソッド（以降「connect メソッド」と呼びます）が使用されます。connect メソッドのタイムアウト監視時間は，各プラットフォームで固有となっています。TP1/Client/J 環境定義でコネクション確立処理に対する最大監視時間を指定してコネクションの確立処理を監視すると，プラットフォーム固有の connect メソッドのタイムアウト監視時間よりも短い時間で TP1/Client/J の API をエラーリターンさせることができます。connect メソッドでのコネクション確立処理に対する最大監視時間は，TP1/Client/J 環境定義の dccltconnecttimeout オペランドで指定します。

2.8.1　内部で connect メソッドを使用する API

API が内部で connect メソッドを使用するかどうかは，TP1/Client/J 環境定義のオペランドの指定によって異なります。内部で connect メソッドを使用する API（メソッド）と，内部で connect メソッドが使用される TP1/Client/J 環境定義のオペランドの条件を次の表に示します。条件が二つ以上ある API（メソッド）の場合，内部で connect メソッドが使用されるには，一つ以上の条件を満たしている必要があります。

表 2–5　内部で connect メソッドを使用する API（メソッド）

項番	内部で connect メソッドを使用するメソッド	内部で connect メソッドが使用される TP1/Client/J 環境定義のオペランドの条件
1	cltAssemSend(byte[] buff, int sendleng, String hostname, int portnum, int timeout, int flags)	• dcsndrcvtype=DCCLT_ONEWAY_SND • dcsndrcvtype=DCCLT_SNDRCV
2	cltSend(byte[] buff, int sendleng, String hostname, int portnum, int flags)	• dcsndrcvtype=DCCLT_ONEWAY_SND • dcsndrcvtype=DCCLT_SNDRCV
3	openConnection(String host, int port)	dcrapdirect=Y かつ dcrapautoconnect=N
4	openConnection()	dcrapdirect=Y かつ dcrapautoconnect=N
5	rpcCall(String group, String service, byte[] in_data, int[] in_len, byte[] out_data, int[] out_len, int flags)	• dcscddirect=Y • dcnamuse=Y または • dcrapdirect=Y かつ dcrapautoconnect=Y
6	rpcCall(String group, String service, byte[] in_data, byte[] out_data, int flags)	
7	rpcCallTo(DCRpcBindTbl direction, String group, String service, byte[] in_data, int[] in_len, byte[] out_data, int[] out_len, int flags)	• dcscddirect=Y • dcnamuse=Y
8	trnBegin()	dcrapdirect=Y かつ dcrapautoconnect=Y

2.8.2　connect メソッドがタイムアウトした場合の例外

各プラットフォームで固有の connect メソッドのタイムアウト監視時間が満了した場合と，dccltconnecttimeout オペランドに指定した時間が満了した場合とでは，TP1/Client/J の API から返る例外が異なります。これは，満了する監視時間の違いが，connect メソッドで発生する例外に影響するためです。

dccltconnecttimeout オペランドに，各プラットフォーム固有のタイムアウト監視時間よりも大きい値を指定した場合，各プラットフォーム固有のタイムアウト監視時間が先に満了します。このため，dccltconnecttimeout オペランドの指定は無効となります。dccltconnecttimeout オペランドの指定を省略した場合，connect メソッドのタイムアウト監視時間は，各プラットフォームで固有となります。

各プラットフォームで固有の connect メソッドのタイムアウト監視時間が満了した場合と，dccltconnecttimeout オペランドに指定した時間が満了した場合の，TP1/Client/J の API が返す例外を次の表に示します。

表 2-6　connect メソッドがタイムアウトした場合に API（メソッド）が返す例外

項番	メソッド	各プラットフォーム固有※1	dccltconnecttimeout オペランド※2
1	cltAssemSend(byte[] buff, int sendleng, String hostname, int portnum, int timeout, int flags)	ErrNetDownAtClientException	ErrClientTimedOutException
2	cltSend(byte[] buff, int sendleng, String hostname, int portnum, int flags)	ErrNetDownAtClientException	ErrClientTimedOutException
3	openConnection(String host, int port)	ErrNetDownAtClientException	ErrClientTimedOutException
4	openConnection()	ErrNetDownAtClientException	ErrClientTimedOutException
5	rpcCall(String group, String service, byte[] in_data, int[] in_len, byte[] out_data, int[] out_len, int flags)	ErrNetDownAtClientException	ErrClientTimedOutException
6	rpcCall(String group, String service, byte[] in_data, byte[] out_data, int flags)	ErrNetDownAtClientException	ErrClientTimedOutException
7	rpcCallTo(DCRpcBindTbl direction, String group, String service, byte[] in_data, int[] in_len, byte[] out_data, int[] out_len, int flags)	ErrConnRefusedException	ErrNetDownAtClientException
8	trnBegin()	ErrConnRefusedException	ErrClientTimedOutException

注※1

各プラットフォームで固有の connect メソッドのタイムアウト監視時間が満了した場合に発生する例外です。

注※2

dccltconnecttimeout オペランドに指定した時間が満了した場合に発生する例外です。

2.9 DCCM3 との接続機能

TP1/Client/J では，RPC や TCP/IP 通信機能を使用して，VOS3 や VOS1 上の DCCM3 と接続できます。

2.9.1 DCCM3 との RPC

TP1/Client/J では，OpenTP1 のサーバだけでなく，DCCM3 のサーバとも RPC を使用した通信ができます。DCCM3 と RPC を行うには，DCCM3 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です。次に示す製品を DCCM3 側に組み込むことで，DCCM3 とも RPC を使用した通信ができます。

DCCM3 側の適用 OS が VOS3 の場合

DCCM3/Internet

DCCM3 側の適用 OS が VOS1 の場合

DCCM3/SERVER/TP1

相手サーバが DCCM3 の場合の特徴は次のとおりです。

- 使用できる RPC の形態は，同期応答型 RPC と非応答型 RPC です。連鎖型 RPC は使用できません。
- トランザクション制御機能は使用できません。
- 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，「2.9.1(4)　DCCM3 論理端末に対して 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称と評価されます。

(1)　相手サーバの指定方法

DCCM3 のサーバと RPC を使用した通信をする場合，相手サーバをサービスグループ名とサービス名で指定します。この指定方法は OpenTP1 のサーバに対して RPC を行う場合と同じです。

- サービスグループ名

  サービスグループ名として不正でないダミーの文字列を指定します。1～31 文字の識別子を指定してください。
- サービス名

  DCCM3 側のトランザクション名称を指定します。使用できる文字は，アルファベット（A～Z，a～z）と数字（0～9）です。文字数の合計が 1～8 文字になるように指定してください。

(2) 相手サーバのアドレス定義

DCCM3 のサーバと RPC を使用した通信をする場合，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，TP1/Client/J 側でサービス名ごとに定義を分けて，サーバのアドレスを定義する必要があります。サーバのアドレスを定義するには，TP1/Client/J 環境定義の dchost オペランドに，それぞれ RPC 受け付け窓口のホスト名およびポート番号を指定します。

(3) RPC の実行手順

相手サーバのアドレスを定義したあと RPC を実行します。TP1/Client/J 環境定義の指定内容によって次に示すように RPC の実行手順は異なります。

(a) dcrapautoconnect オペランドに Y を指定した場合

1. rpcOpen メソッドを実行して，定義を読み込みます。
2. rpcCall メソッドを実行します。
   TP1/Client/J 環境定義の dchost オペランドで指定した RPC 受け付け窓口に，コネクションが確立されます。コネクションの確立後，RPC が行われます。

(b) dcrapautoconnect オペランドに N を指定した場合，または指定を省略した場合

1. rpcOpen メソッドを実行して，定義を読み込みます。
2. 引数なしの openConnection メソッドを実行します。
   TP1/Client/J 環境定義の dchost オペランドで指定した RPC 受け付け窓口に，コネクションが確立されます。
3. コネクションの確立後，rpcCall メソッドを実行して RPC を行います。

(c) dchost オペランドで指定したサーバ以外と通信する場合

1. rpcOpen メソッドを実行して，定義を読み込みます。
   この手順は省略できます。省略した場合でも，2.および 3.は実行できます。
2. openConnection メソッドの引数に RPC 受け付け窓口（ホスト名およびポート番号）を指定して，openConnection メソッド実行します。
3. コネクションの確立後，rpcCall メソッドを実行して RPC を行います。

(4) DCCM3 論理端末に対して RPC を行う場合の負荷分散

TP1/Client/J 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，コネクション確立時に接続先を複数の DCCM3 に振り分けて負荷を分散できます。TP1/Client/J は，TP1/Client/J 環境定義の dchostselect オペランドに Y を指定することで，TP1/Client/J 環境定義の dchost オペランドに指

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。ある DCCM3 論理端末との接続に失敗すると，それ以外の DCCM3 論理端末を dchost オペランドに指定された順で選択し，接続を試みます。この処理を繰り返し，TP1/Client/J 環境定義の dchost オペランドに指定されたすべての DCCM3 論理端末との接続にすべて失敗したときに，初めてエラーを検知します。

TP1/Client/J が DCCM3 論理端末と通信する場合の TP1/Client/J 環境定義とコネクションを確立するメソッドの関係を次の表に示します。

表 2-7　DCCM3 論理端末と通信する場合の TP1/Client/J 環境定義とコネクションを確立するメソッド

項番	TP1/Client/J 環境定義 dcrapautoconnect	コネクションを確立するメソッド
1	Y	rpcCall※
2	N	openConnection（引数なし）
3	省略	

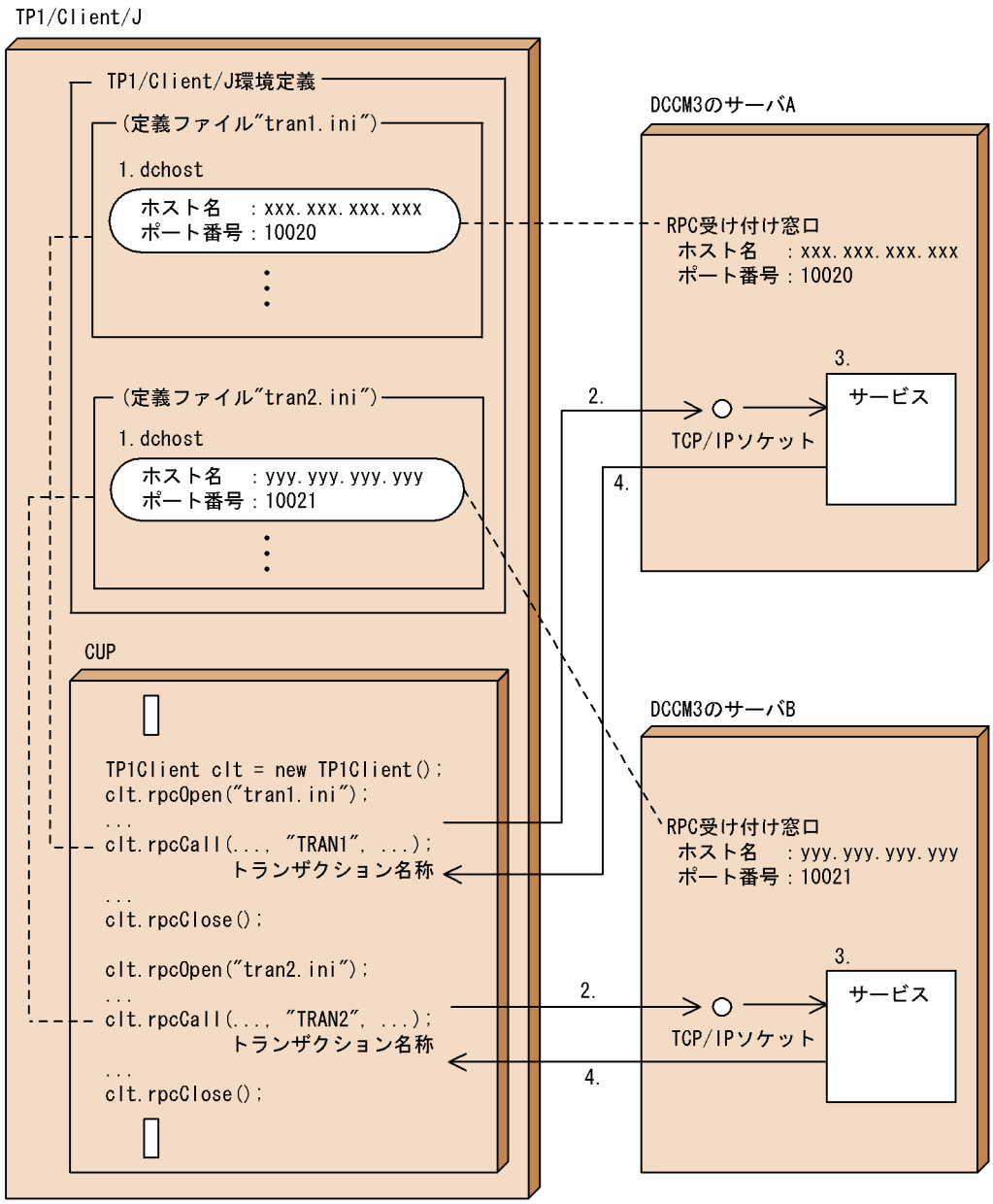
注※

コネクションが確立されていない場合，自動でコネクションを確立します。

(5)　TP1/Client/J 環境定義の定義例

次に示す図のように動く，DCCM3 接続時の TP1/Client/J 環境定義の定義例を示します。

図 2-34　DCCM3 のサーバとの RPC



1. TP1/Client/J 環境定義で，各トランザクションを処理するサーバのアドレス（RPC 受け付け窓口）を，定義ファイルを分けて dchost オペランドに定義します。
2. RPC を行うとき，定義ファイルから 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します。
3. RPC のメッセージを解釈し，要求されたサービスを実行します。
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します。

定義ファイル"tran1.ini"の定義例

```
dcrapdirect=Y
dcwatchtim=180
dcrapautoconnect=Y
dchostselect=Y
```

```
#”TRAN1”のトランザクションを処理できるサーバのアドレスを定義
dchost=xxx.xxx.xxx.xxx:10020,zzz.zzz.zzz.zzz:10022
```

定義ファイル"tran2.ini"の定義例

```
dcrapdirect=Y
dcwatchtim=180
dcrapautoconnect=Y
dchostselect=Y
#”TRAN2”のトランザクションを処理できるサーバのアドレスを定義
dchost=yyy.yyy.yyy.yyy:10021,zzz.zzz.zzz.zzz:10022
```

上記の TP1/Client/J 環境定義を使用したプログラム例を次に示します。

プログラム例

```
import JP.co.Hitachi.soft.OpenTP1.*;
public class DCCM3Caller
{
  ...
  public void Function1(){
    TP1Client clt = new TP1Client();

    // TRAN1のトランザクションを呼び出すRPC
    clt.rpcOpen(”tran1.ini”);
    ...
    // 同期応答型RPC
    clt.rpcCall(”dummysvg”, ”TRAN1”, ..., TP1Client.DCNOFLAGS);
    ...
    clt.rpcClose();

    // TRAN2のトランザクションを呼び出すRPC
    // 要求先のサーバアドレス取得のために定義を読み直す
    clt.rpcOpen(”tran2.ini”);
    ...
    // 非応答型RPC
    clt.rpcCall(”dummysvg”, ”TRAN2”, ..., TP1Client.DCRPC_NOREPLY);
    ...
    clt.rpcClose();
  }
}
```

(6) DCCM3 論理端末に対して RPC を行う場合の注意事項

- リモート API 機能を使用した RPC だけ使用できます。
- 連鎖型 RPC は使用できません。
- トランザクション制御機能は使用できません。
- 文字列のデータを含むメッセージを送受信する場合は，あらかじめメッセージ中の文字コードを通信先システムと決めた上で，必要に応じて UAP で文字コード変換を行ってください。Java 上では，Unicode が文字列データのメモリ上の内部表現として使用されます。

- TP1/Client/J 環境定義の dccltinquiretime オペランドを指定しても無効になります。CUP からサーバに対する問い合わせ間隔最大時間は，DCCM3 の「端末放置監視時間」で指定してください。
- 常設コネクション，かつオートコネクトモードを使用したシステムで，rap サーバが常設コネクションを解放するタイミングと，CUP 実行プロセスが RPC を実行するタイミングが重なった場合，RPC の実行に失敗して例外を返すことがあります。この現象を回避するには，TP1/Client/J 環境定義の dcinquiretimecheck オペランドに Y を指定し，dccltinquiretime オペランドに DCCM3 の「端末放置監視時間」と同じ値を指定してください。dcinquiretimecheck オペランド，および dccltinquiretime オペランドの説明については「5.2.2　オペランド」を参照してください。
- DCCM3 側の注意事項については，マニュアル「VOS3 データマネジメントシステム XDM E2 系 解説」およびマニュアル「VOS1 データコミュニケーションマネジメントシステム DCCM3 解説」を参照してください。

2.9.2　DCCM3 との TCP/IP 通信

DCCM3 論理端末と TCP/IP 通信を行う場合の注意事項は次のとおりです。

文字列のデータを含むメッセージを送受信する場合は，あらかじめメッセージ中の文字コードを通信先システムと決めた上で，必要に応じて UAP で文字コード変換を行ってください。Java 上では，Unicode が文字列データのメモリ上の内部表現として使用されます。

2.9.3　DCCM3 論理端末への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，端末識別情報を DCCM3 論理端末に通知し，CUP に割り当てられる DCCM3 の論理端末を固定することができます。

(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/J 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，複数の CUP を同一マシンから起動した場合，どの CUP からの要求でも，DCCM3 論理端末から見ると IP アドレスが同じになってしまいます。複数の CUP から同じ DCCM3 論理端末のポートにサービスを要求すると，DCCM3 の定義では区別が付きません。そのため，該当する DCCM3 の論理端末が複数定義されていた場合，DCCM3 のどの論理端末に CUP が割り当てられるのかが不定になってしまいます。サービス要求を受け付ける DCCM3 の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

CUP が DCCM3 論理端末と常設コネクションを確立するときに，端末識別情報を DCCM3 論理端末に通知することで，CUP に割り当てられる DCCM3 の論理端末を固定することができます。これを**端末識別**

情報設定機能といいます。この機能を使用することで，CUP を常に同じ DCCM3 の論理端末に割り当てることができます。なお，DCCM3 側では，この機能を**端末固定割り当て機能**といいます。

端末識別情報設定機能を使用していない場合と使用している場合の CUP と DCCM3 論理端末の関係を，次の図にそれぞれ示します。

図 2-35　CUP と DCCM3 論理端末の関係（端末識別情報設定機能を使用していない場合）

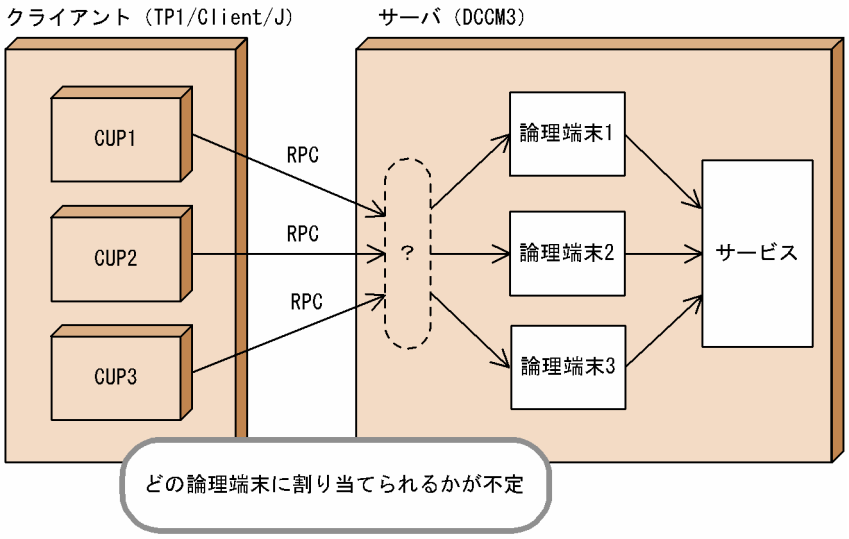
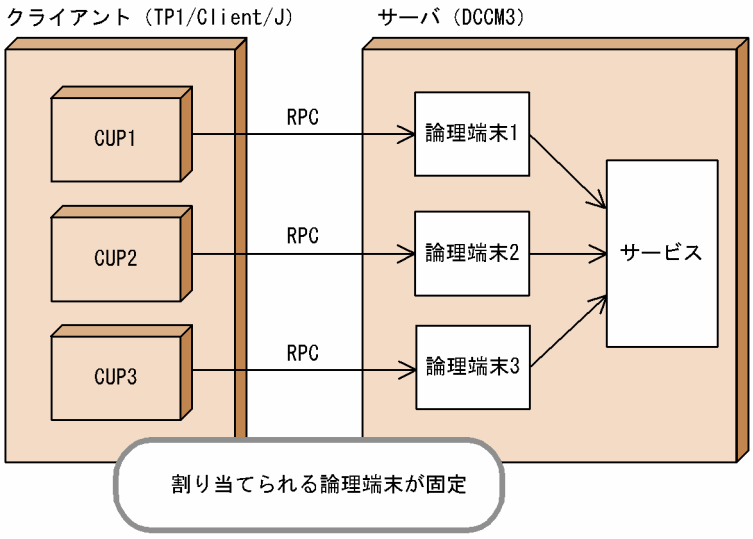


図 2-36　CUP と DCCM3 論理端末の関係（端末識別情報設定機能を使用している場合）



端末識別情報設定機能は，次のどちらかの方法で使用できます。

方法 1

1. TP1/Client/J 環境定義の dchost オペランドに DCCM3 論理端末のホスト名を指定します。
2. TP1/Client/J 環境定義の dchost オペランドまたは dcrapport オペランドに DCCM3 論理端末のポート番号を指定します。
3. setConnectInformation メソッドに端末識別情報を設定し，メソッドを呼び出します。

4. 次のどちらかの方法で DCCM3 論理端末との常設コネクションを確立します。

・ openConnection メソッドを呼び出します。引数有りの openConnection メソッドの場合，引数 host に DCCM3 論理端末のホスト名，引数 port に DCCM3 論理端末のポート番号を指定します。

・ TP1/Client/J 環境定義の dcrapautoconnect オペランドに Y を指定し，rpcCall メソッドを呼び出します。

方法 2

1. TP1/Client/J 環境定義の dchost オペランドに DCCM3 論理端末のホスト名を指定します。
2. TP1/Client/J 環境定義の dchost オペランドまたは dcrapport オペランドに DCCM3 論理端末のポート番号を指定します。
3. TP1/Client/J 環境定義の dccltconnectinf オペランドに端末識別情報を設定します。
4. 次のどちらかの方法で DCCM3 論理端末との常設コネクションを確立します。

・ openConnection メソッドを呼び出します。引数有りの openConnection メソッドの場合，引数 host に DCCM3 論理端末のホスト名，引数 port に DCCM3 論理端末のポート番号を指定します。

・ TP1/Client/J 環境定義の dcrapautoconnect オペランドに Y を指定し，rpcCall メソッドを呼び出します。

注意事項

TP1/Client/J 環境定義の dccltconnectinf オペランドに端末識別情報を設定して，setConnectInformation メソッドに端末識別情報を設定した場合は，setConnectInformation メソッドの設定が有効になります。dccltconnectinf オペランドに設定した値は，setConnectInformation メソッドを呼び出したあと，再び rpcOpen メソッドを呼び出すまで無視されます。

(3) DCCM3 論理端末に端末識別情報を通知する場合の注意事項

- 端末固定割り当て機能を使用した DCCM3 の論理端末名称と，TP1/Client/J で定義した端末識別情報とが一致していない場合に，次のメソッドを呼び出したときは ErrNetDownAtClientException 例外を返します。
  - openConnection メソッド
  - rpcCall メソッド（ただし，TP1/Client/J 環境定義の dcrapautoconnect オペランドに Y を指定した場合）
- 端末固定割り当て機能を使用した DCCM3 の論理端末に対して，TP1/Client/J で端末識別情報を設定しないで次のメソッドを呼び出した場合，ErrNetDownAtClientException 例外を返します。
  - openConnection メソッド
  - rpcCall メソッド（ただし，TP1/Client/J 環境定義の dcrapautoconnect オペランドに Y を指定した場合）

- 端末固定割り当て機能を使用していない DCCM3 の論理端末に対して，TP1/Client/J から端末識別情報を設定して常設コネクションの確立を要求した場合，DCCM3 は TP1/Client/J が設定した端末識別情報の設定内容を無視します。
- TP1/Client/J から端末識別情報を設定して，TP1/Server の rap サーバと常設コネクションを確立する場合，rap サーバは TP1/Client/J が設定した端末識別情報の設定内容を無視します。また，TP1/Client/J が rap サーバを介して DCCM3 へ RPC を発行する場合，TP1/Client/J で設定した端末識別情報は DCCM3 に伝達されません。
- 端末識別情報は，リモート API 機能を使用した RPC の場合だけ有効となります。ネームサービスを使用した RPC，またはスケジューラダイレクト機能を使用した RPC の場合は，端末識別情報を設定しても無視されます。

2.10　XA リソースサービス機能

XA リソースサービス機能を使用する場合，uCosminexus TP1 Connector または Cosminexus TP1 Connector が必要です。TP1/Client/J 単体での動作は保証できません。ここでは，XA リソースサービス機能を使用した場合，TP1/Client/J が使用できる機能について説明します。

TP1/Client/J がサポートする通信方式を次の表に示します。

表 2-8　通信方式別サポート一覧

TP1/Client/J の通信方式	リモート API 機能を使用	スケジューラダイレクト機能を使用	ネームサービスを使用
サポート可否	○	×	×

（凡例）

○：サポートしています。

×：サポートしていません。

TP1/Client/J がサポートするコネクトモードを次の表に示します。

表 2-9　TP1/Client/J がサポートするコネクトモード

コネクトモード	サポート可否
非オートコネクトモード	×
オートコネクトモード	○

（凡例）

○：サポートしています。

×：サポートしていません。

TP1/Client/J がサポートする RPC の呼び出し形態を次の表に示します。

表 2-10　TP1/Client/J がサポートする RPC の呼び出し形態

RPC の呼び出し形態	サポート可否
同期応答型 RPC	○
非同期応答型 RPC	○
連鎖型 RPC	○

（凡例）

○：サポートしています。

なお，TP1/Client/J 環境定義の dchost オペランドに指定できる窓口となる TP1/Server は一つだけです。複数指定した場合の動作は保証できません。

2.11　トラブルシュート機能

TP1/Client/J は，トラブルシュート機能として，UAP トレース，データトレース，エラートレース，メモリトレース，メソッドトレース，デバッグトレース，性能解析トレース，および性能検証用トレースを取得できます。

UAP トレース，データトレース，エラートレース，およびメソッドトレースはファイルに出力されます。メモリトレースは CUP で用意した String 型の配列に格納されます。デバッグトレースは，TP1/Client/J のメモリ内に取得されます。また，TP1/Client/J の提供するメソッドが例外を返した場合は，ファイルまたは標準出力に出力されます。性能解析トレースは Cosminexus Application Server 上で，性能検証用トレースは TP1/Server 上で出力します。なお，TP1/Client/J は各トレースのファイル出力時に排他制御をしているため，マルチスレッドで作成された CUP の場合，出力頻度の高い UAP トレース，データトレース，およびメソッドトレースの出力先を同一にすると，スループットが劣化するおそれがあります。この場合は，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。

Java アプレットの場合，Java のセキュリティの制約で，UAP トレース，データトレース，エラートレース，メソッドトレース，および性能検証用トレースは使用できません。また，性能解析トレースは取得できません。

Java サーブレットの場合，使用するアプリケーションサーバによっては，セキュリティの制約で使用できないトレースがあります。Java サーブレットのセキュリティの制約の詳細については，アプリケーションサーバのマニュアルを確認してください。

トレースを出力する場合，環境変数 TZ にタイムゾーンを設定してください。設定していないと，トレース情報の時刻が正しく出力されません。

2.11.1　トレースファイルの出力内容

トレースファイルに指定できるオプションと出力ファイル名を次の表に示します。

表 2-11　トレースファイルのオプションとファイル名

トレースファイル	オプション	ファイル出力先指定方法	出力ファイル名
UAP トレース	• ファイル出力ディレクトリ • トレースファイルのサイズ	setUapTraceMode メソッドの TrcPath 引数，または dcuaptracepath オペランド	dcuap1.trc dcuap2.trc
データトレース	• ファイル出力ディレクトリ • トレースファイルのサイズ • データの最大サイズ	setDataTraceMode メソッドの TrcPath 引数，または dcdatatracepath オペランド	dcdat1.trc dcdat2.trc
エラートレース	• ファイル出力ディレクトリ • トレースファイルのサイズ	setErrorTraceMode メソッドの TrcPath 引数，または dcerrtracepath オペランド	dcerr1.trc dcerr2.trc

トレースファイル	オプション	ファイル出力先指定方法	出力ファイル名
メソッドトレース	• ファイル出力ディレクトリ • トレースファイルのサイズ	setMethodTraceMode メソッドの TrcPath 引数，または dcmethodtracepath オペランド	dcmtd1.trc dcmtd2.trc

指定したディレクトリがない場合，および指定したディレクトリに書き込み権限がない場合は，トレースファイルを出力できません。指定したファイル出力先ディレクトリに出力ファイル名のファイルがない場合は，該当するディレクトリにトレースファイルを作成します。ファイルへの書き込みは追加モードで実行されます。二つのトレースファイルはラウンドロビン方式で使用されます。指定したトレースファイルのサイズを超えた書き込みが発生すると，ファイルの切り替えを行い，次のファイルの先頭から書き込みを再開します。

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。
- 一つのトレースデータの出力サイズによっては指定したトレースファイルのサイズよりファイルサイズが大きくなる場合があります。

2.11.2 UAP トレース

UAP トレース情報は，setUapTraceMode メソッドの TrcPath 引数で指定したディレクトリ，または TP1/Client/J 環境定義の dcuaptracepath オペランドで指定したディレクトリに，dcuap1.trc および dcuap2.trc というファイル名で出力します。ファイルのサイズは setUapTraceMode メソッドの size 引数，または TP1/Client/J 環境定義の dcuaptracesize オペランドで指定します。

UAP トレースは，TP1/Client/J が提供するメソッドの開始時と終了時に次の情報を取得します。

- 日付
- 時刻
- 実行スレッド名
- 呼び出したメソッド名
- 発生した例外
- 指定した引数の情報
- 呼び出したメソッドに関係する情報
- 送受信したデータの内容

なお，送受信データ長が 60 バイトを超える場合は先頭の 60 バイトを取得します。

UAP トレースファイルの出力形式を次に示します。

```
yyyy/mm/dd hh:mm:ss.uuu Location = lll ThreadName = ttt
MethodName = nnnnnnnnnnnnnnnnnnn
Exception = eeeeeeeeeeeeeeeeeee
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 00 saispp..........
00000010 00 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 00 saisvr01........
:         :         :
```

- yyyy/mm/dd hh:mm:ss.uuu：UAP トレースを取得した日時
- lll：出入口種別
- ttt：実行スレッド名
- nnnnnnnnnnnnnnnnnnn：呼び出したメソッド名
- eeeeeeeeeeeeeeeeeee：発生した例外
- ADDRESS 以降の文字列：各種情報（指定した引数の情報，呼び出したメソッドに関係する情報，および送受信したデータの内容）

各種情報に出力する情報は，メソッドおよび取得するタイミングによって異なります。各メソッドの各種情報に出力する情報を次に示します。

表 2-12　openConnection()メソッドおよび openConnection(host,port)メソッドの各種情報

取得場所	項目	位置		長さ（バイト）	内容
		16 進数	10 進数		
入り口	接続先ホスト名	0	0	64※	host 引数で指定した接続先ホスト名。 引数を指定していないと，取得されません。その場合，領域はゼロクリアされます。
	接続先ポート番号	40	64	4	引数で指定した接続先ポート番号。 引数を指定していないと，取得されません。その場合，-1 が設定された状態で，UAP トレースが取得されます。
	問い合わせ間隔最大時間	44	68	4	メソッドの実行時に使用した問い合わせ間隔最大時間の値。 TP1/Client/J 環境定義の dccltinquiretime オペランド，または setDccltinquiretime メソッドで指定した問い合わせ間隔最大時間の値です。
	最大応答待ち時間	48	72	4	メソッドの実行時に使用した最大応答待ち時間の値。 TP1/Client/J 環境定義の dcwatchtim オペランド，または setDcwatchtim メソッドで指定した最大応答待ち時間の値です。
	入力ストリームチェック間隔	4C	76	4	メソッドの実行時に使用した入力ストリームチェック間隔の値。

取得場所	項目	位置		長さ（バイト）	内容
		16 進数	10 進数		
入り口	入力ストリームチェック間隔	4C	76	4	TP1/Client/J 環境定義の dcselint オペランド，または setDcselint メソッドで指定した入力ストリームチェック間隔の値です。
出口	性能解析トレースの識別情報	0	0	26	性能解析トレースの識別情報

注※

host 引数に 65 バイト以上のホスト名を指定した場合，長さは指定したホスト名のバイト数になります。また，以降の項目の位置は，（host 引数で指定したホスト名のバイト数 − 64）分だけずれて出力されます。

表 2-13　openConnection(url,flags)メソッドの各種情報

取得場所	項目	位置		長さ（バイト）	内容
		16 進数	10 進数		
入り口	接続先 URL	0	0	128※	url 引数で指定した TP1/Web の URL
	オプションフラグ	80	128	2	openConnection メソッドの flags 引数で指定した値。次の値が設定されます。 • DCSESSION 0x0001
	最大応答待ち時間	82	130	4	setDcwatchtim メソッドで指定した最大応答待ち時間
出口	なし	−	−	−	−

（凡例）−：該当なし

注※

url 引数に 129 バイト以上の接続先 URL を指定した場合，長さは指定した接続先 URL のバイト数になります。また，以降の項目の位置は，（url 引数で指定した接続先 URL のバイト数 − 128）分だけずれて出力されます。

表 2-14　rpcCall メソッドの各種情報

取得場所	項目	位置		長さ（バイト）	内容
		16 進数	10 進数		
入り口	サービスグループ名	0	0	32	呼び出すサービスのサービスグループ名
	サービス名	20	32	32	呼び出すサービス名
	送信データ長	40	64	4	送信データの長さ
	送信データ	44	68	60	送信するデータ
	受信データ長	80	128	4	受信するデータの長さ
	オプションフラグ	84	132	4	rpcCall メソッドの flags 引数で指定した値。次のどれかの値が設定されます。 • DCNOFLAGS 0x00000000 • DCRPC_NOREPLY 0x00000001

取得場所	項目	位置		長さ（バイト）	内容
		16 進数	10 進数		
入り口	オプションフラグ	84	132	4	• DCRPC_CHAINED 0x00000004 • DCRPC_TPNOTRAN 0x00000020
	最大応答待ち時間	88	136	4	メソッドの実行時に使用した最大応答待ち時間の値。 TP1/Client/J 環境定義の dcwatchtim オペランド，または setDcwatchtim メソッドで指定した最大応答待ち時間の値です。
	サーバ通信遅延時間	8C	140	4	メソッドの実行時に使用したサーバ通信遅延時間の値。 TP1/Client/J 環境定義の dccltdelay オペランド，または setDccltdelay メソッドで指定したサーバ通信遅延時間の値です。リモート API 機能を使用しない場合，0 が設定されます。
	入力ストリームチェック間隔	90	144	4	メソッドの実行時に使用した入力ストリームチェック間隔の値。 TP1/Client/J 環境定義の dcselint オペランド，または setDcselint メソッドで指定した入力ストリームチェック間隔の値です。
出口※	受信データ長	0	0	4	受信した応答の長さ
	受信データ	4	4	60	受信したデータ
	性能解析トレースの識別情報	40	64	26	性能解析トレースの識別情報

注※

非応答型サービス要求の場合，出口情報は性能解析トレースの識別情報だけです。

表 2-15　rpcCallTo メソッドの各種情報

取得場所	項目	位置		長さ（バイト）	内容
		16 進数	10 進数		
入り口	接続先ホスト名	0	0	64※1	メソッドの実行時に使用した通信相手のホスト名。 DCRpcBindTbl で指定した接続先ホスト名称です。
	接続先ポート番号	40	64	4	メソッドの実行時に使用した通信相手のホストのポート番号。 DCRpcBindTbl で指定した接続先ホストのポート番号です。
	接続先オプションフラグ	44	68	4	rpcCallTo メソッドの通信プロトコルフラグ • DCRPC_SCDPORT 0x00000002
	サービスグループ名	48	72	32	呼び出すサービスのサービスグループ名
	サービス名	68	104	32	呼び出すサービス名
	送信データ長	88	136	4	送信データの長さ

取得場所	項目	位置		長さ（バイト）	内容
		16 進数	10 進数		
入り口	送信データ	8C	140	60	送信するデータ
	受信データ長	C8	200	4	受信するデータの長さ
	オプションフラグ	CC	204	4	rpcCallTo メソッドの flags 引数で指定した値。次のどれかの値が設定されます。 • DCNOFLAGS 0x00000000 • DCRPC_NOREPLY 0x00000001
	最大応答待ち時間	D0	208	4	メソッドの実行時に使用した最大応答待ち時間の値。 TP1/Client/J 環境定義の dcwatchtim オペランド，または setDcwatchtim メソッドで指定した最大応答待ち時間の値です。
	サーバ通信遅延時間	D4	212	4	メソッドの実行時に使用したサーバ通信遅延時間の値。 TP1/Client/J 環境定義の dccltdelay オペランド，または setDccltdelay メソッドで指定したサーバ通信遅延時間の値です。リモート API 機能を使用しない場合，0 が設定されます。
	入力ストリームチェック間隔	D8	216	4	メソッドの実行時に使用した入力ストリームチェック間隔の値。 TP1/Client/J 環境定義の dcselint オペランド，または setDcselint メソッドで指定した入力ストリームチェック間隔の値です。
出口※2	受信データ長	0	0	4	受信した応答の長さ
	受信データ	4	4	60	受信したデータ
	性能解析トレースの識別情報	40	64	26	性能解析トレースの識別情報

注※1

65 バイト以上のホスト名を指定した場合，長さは指定したホスト名のバイト数になります。また，以降の項目の位置は，（指定したホスト名のバイト数 − 64）分だけずれて出力されます。

注※2

非応答型サービス要求の場合，出口情報は性能解析トレースの識別情報だけです。

表 2-16　closeConnection メソッドの各種情報

取得場所	項目	位置		長さ（バイト）	内容
		16 進数	10 進数		
入り口	最大応答待ち時間	0	0	4	メソッドの実行時に使用した最大応答待ち時間の値。 TP1/Client/J 環境定義の dcwatchtim オペランド，または setDcwatchtim メソッドで指定した最大応答待ち時間の値です。

取得場所	項目	位置		長さ（バイト）	内容
		16 進数	10 進数		
入り口	入力ストリームチェック間隔	4	4	4	メソッドの実行時に使用した入力ストリームチェック間隔の値。 TP1/Client/J 環境定義の dcselint オペランド，または setDcselint メソッドで指定した入力ストリームチェック間隔の値です。
出口	性能解析トレースの識別情報	0	0	26	性能解析トレースの識別情報

表 2-17　setDcwatchtim メソッドの各種情報

取得場所	項目	位置		長さ（バイト）	内容
		16 進数	10 進数		
入り口	最大応答待ち時間	0	0	4	最大応答待ち時間の値
出口	なし	－	－	－	－

（凡例）－：該当なし

表 2-18　setDccltinquiretime メソッドの各種情報

取得場所	項目	位置		長さ（バイト）	内容
		16 進数	10 進数		
入り口	問い合わせ間隔最大時間	0	0	4	問い合わせ間隔最大の値
出口	なし	－	－	－	－

（凡例）－：該当なし

表 2-19　setDccltdelay メソッドの各種情報

取得場所	項目	位置		長さ（バイト）	内容
		16 進数	10 進数		
入り口	サーバ通信遅延時間	0	0	4	サーバ通信遅延時間の値
出口	なし	－	－	－	－

（凡例）－：該当なし

表 2-20　setDcselint メソッドの各種情報

取得場所	項目	位置		長さ（バイト）	内容
		16 進数	10 進数		
入り口	入力ストリームチェック間隔	0	0	4	入力ストリームチェック間隔の値
出口	なし	－	－	－	－

（凡例）－：該当なし

表 2-21　setDccltextend メソッドの各種情報

取得場所	項目	位置		長さ（バイト）	内容
		16 進数	10 進数		
入り口	拡張機能のレベル	0	0	4	TP1/Client/J の機能の拡張レベル
出口	なし	－	－	－	－

（凡例）－：該当なし

表 2-22　setRpcextend メソッドの各種情報

取得場所	項目	位置		長さ（バイト）	内容
		16 進数	10 進数		
入り口	RPC 拡張機能のレベル	0	0	4	TP1/Client/J の RPC 機能の拡張レベル。 次の値の論理和が設定されます。 • DCRPC_SCD_LOAD_PRIORITY 0x00000008 • DCRPC_WATCHTIMINHERIT 0x00000010 • DCRPC_RAP_AUTOCONNECT 0x00000020 • DCRPC_WATCHTIMRPCINHERIT 0x00000040
出口	なし	－	－	－	－

（凡例）－：該当なし

表 2-23　setDchost メソッドの各種情報

取得場所	項目	位置		長さ（バイト）	内容
		16 進数	10 進数		
入り口	接続先ホスト名	0	0	64※	host 引数で指定したホスト名
	接続先ポート番号	40	64	4	port 引数で指定したポート番号
出口	なし	－	－	－	－

（凡例）－：該当なし

注※

host 引数に 65 バイト以上のホスト名を指定した場合，長さは指定したホスト名のバイト数になります。また，以降の項目の位置は，（host 引数で指定したホスト名のバイト数 － 64）分だけずれて出力されます。

表 2-24　rpcOpen メソッドの各種情報

取得場所	項目	位置		長さ（バイト）	内容
		16 進数	10 進数		
入り口	TP1/Client/J 環境定義ファイル名	0	0	256	deffilename 引数で指定した TP1/Client/J 環境定義ファイル名 引数を指定していないと，取得されません。その場合，領域はゼロクリアされます。
出口	なし	－	－	－	－

（凡例）－：該当なし

表 2-25　trnBegin メソッドの各種情報

取得場所	項目	位置		長さ（バイト）	内容
		16 進数	10 進数		
入り口	トランザクションブランチ限界経過時間	0	0	4	TP1/Client/J 環境定義の dcclttrexptm オペランドで指定した値
	トランザクションブランチ CPU 監視時間	4	4	4	TP1/Client/J 環境定義の dcclttrcputm オペランドで指定した値
	トランザクションブランチの監視時間の対象	8	8	1	TP1/Client/J 環境定義の dcclttrexpsp オペランドで指定した値 • Y 89 • N 78 • F 70
	統計情報取得項目	9	9	4	TP1/Client/J 環境定義の dcclttrstatisitem オペランドで指定した値 • base 0x80000000 • executiontime 0x40000000 • cputime 0x20000000 • function 0x10000000
	トランザクション最適化項目	D	13	4	TP1/Client/J 環境定義の dcclttroptiitem オペランドで指定した値 • nothing 0 • base 0x00000003 • asyncprepare 0x00000004 • recursivemigrate 0x00000008
	トランザクション同期点処理時の最大通信待ち時間	11	17	4	TP1/Client/J 環境定義の dcclttrwatchtime オペランドで指定した値
	ロールバック情報取得の値	15	21	4	TP1/Client/J 環境定義の dcclttrrbinfo オペランドで指定した値 • no 0 • self 0x00000001

取得場所	項目	位置		長さ（バイト）	内容
		16 進数	10 進数		
入り口	ロールバック情報取得の値	15	21	4	• remote 0x00000002 • all 0x00000003
	トランザクションブランチ最大実行可能時間の値	19	25	4	TP1/Client/J 環境定義の dcclttrlimittime オペランドの値
	ロールバック完了通知の値	1D	29	4	TP1/Client/J 環境定義の dcclttrrbrcv オペランドの値 • Y 89 • N 78
	UAP 障害時の同期点処理方式の値	21	33	4	TP1/Client/J 環境定義の dcclttrrecoverytype オペランドの値 • type1 0x00000001 • type2 0x00000002 • type3 0x00000004
出口	性能解析トレースの識別情報	0	0	26	性能解析トレースの識別情報

表 2-26　trnInfo メソッドの各種情報

取得場所	項目	位置		長さ（バイト）	内容
		16 進数	10 進数		
入り口	なし	－	－	－	－
出口	トランザクション中フラグ	0	0	1	• 1 トランザクション中 • 0 トランザクション中ではない

（凡例）－：該当なし

表 2-27　getTrnID メソッドの各種情報

取得場所	項目	位置		長さ（バイト）	内容
		16 進数	10 進数		
入り口	なし	－	－	－	－
出口	トランザクショングローバル識別子	0	0	16	メソッドが正常終了の場合だけ取得されます。
	トランザクションブランチ識別子	10	16	16	メソッドが正常終了の場合だけ取得されます。

（凡例）－：該当なし

表 2-28　cltReceive メソッドの各種情報

取得場所	項目	位置		長さ（バイト）	内容
		16 進数	10 進数		
入り口	受信データ長	0	0	4	受信するデータの長さ
	タイムアウト設定時間	4	4	4	タイムアウト設定時間
	オプションフラグ	8	8	4	cltReceive メソッドの flags 引数で指定した値。次のどれかの値が設定されます。 • DCNOFLAGS 0x00000000 • DCCLT_RCV_CLOSE 0x00000002
出口	受信データ長	0	0	4	受信した応答の長さ
	受信データ	4	4	60	受信したデータ

表 2-29　cltSend メソッドの各種情報

取得場所	項目	位置		長さ（バイト）	内容
		16 進数	10 進数		
入り口	送信データ	0	0	60	送信するデータ
	送信データ長	3C	60	4	送信データの長さ
	接続先ホスト名	40	64	64※	hostname 引数で指定した通信相手のホスト名
	接続先ポート番号	80	128	4	portnum 引数で指定した通信相手のポート番号
	オプションフラグ	84	132	4	cltSend メソッドの flags 引数で指定した値。次のどれかの値が設定されます。 • DCNOFLAGS 0x00000000 • DCCLT_SND_CLOSE 0x00000001
出口	なし	－	－	－	－

（凡例）－：該当なし

注※

hostname 引数に 65 バイト以上のホスト名を指定した場合，長さは指定したホスト名のバイト数になります。また，以降の項目の位置は，（hostname 引数で指定したホスト名のバイト数 － 64）分だけずれて出力されます。

表 2-30　cltAssemSend メソッドの各種情報

取得場所	項目	位置		長さ（バイト）	内容
		16 進数	10 進数		
入り口	送信データ	0	0	60	送信するデータ
	送信データ長	3C	60	4	送信データの長さ
	接続先ホスト名	40	64	64※	hostname 引数で指定した通信相手のホスト名

取得場所	項目	位置		長さ（バイト）	内容
		16 進数	10 進数		
入り口	接続先ポート番号	80	128	4	portnum 引数で指定した通信相手のポート番号
	タイムアウト設定時間	84	132	4	タイムアウト設定時間
	オプションフラグ	88	136	4	cltAssemSend メソッドの flags 引数で指定した値。次のどれかの値が設定されます。 • DCNOFLAGS 0x00000000 • DCCLT_SND_CLOSE 0x00000001
出口	なし	−	−	−	−

（凡例）−：該当なし

注※

hostname 引数に 65 バイト以上のホスト名を指定した場合，長さは指定したホスト名のバイト数になります。また，以降の項目の位置は，（hostname 引数で指定したホスト名のバイト数 − 64）分だけずれて出力されます。

表 2-31　cltAssemReceive メソッドの各種情報

取得場所	項目	位置		長さ（バイト）	内容
		16 進数	10 進数		
入り口	タイムアウト設定時間	0	0	4	タイムアウト設定時間
	オプションフラグ	4	4	4	cltAssemReceive メソッドの flags 引数で指定した値。次のどれかの値が設定されます。 • DCNOFLAGS 0x00000000 • DCCLT_RCV_CLOSE 0x00000002
出口	受信データ長	0	0	4	受信した応答の長さ
	受信データ	4	4	60	受信したデータ

表 2-32　setConnectInformation メソッドの各種情報

取得場所	項目	位置		長さ（バイト）	内容
		16 進数	10 進数		
入り口	DCCM3 論理端末の論理端末名称	0	0	64	inf 引数で指定した DCCM3 論理端末の論理端末名称
	端末識別情報長	40	64	2	inf_len 引数で指定した端末識別情報長
出口	なし	−	−	−	−

（凡例）−：該当なし

表 2-33　acceptNotification メソッドの各種情報

取得場所	項目	位置		長さ（バイト）	内容
		16 進数	10 進数		
入り口	サーバからの通知メッセージを格納する領域の長さ	0	0	4	inf_len 引数で指定した，サーバからの通知メッセージを格納する領域の長さ
	サーバからの通知メッセージを受信するポート番号	4	4	4	port 引数で指定した，サーバからの通知メッセージを受信するポート番号
	タイムアウト値	8	8	4	timeout 引数で指定した，タイムアウト値
出口	サーバからの通知メッセージ	0	0	60	inf 引数に格納した，サーバからの通知メッセージ
	サーバからの通知メッセージの長さ	3C	60	4	inf_len 引数に格納した，サーバからの通知メッセージの長さ
	サーバのホスト名	40	64	64	hostname 引数に格納した，サーバのホスト名
	サーバのノード識別子	80	128	8	nodeid 引数に格納した，サーバのノード識別子

表 2-34　cancelNotification メソッドの各種情報

取得場所	項目	位置		長さ（バイト）	内容
		16 進数	10 進数		
入り口	CUP に通知するメッセージ	0	0	60	inf 引数で指定した，CUP に通知するメッセージ
	CUP に通知するメッセージの長さ	3C	60	4	inf_len 引数で指定した，CUP に通知するメッセージの長さ
	一方通知受信待ち状態の CUP のホスト名	40	64	64※	hostname 引数で指定した，一方通知受信待ち状態の CUP のホスト名
	一方通知受信待ち状態の CUP のポート番号	80	128	4	port 引数で指定した，一方通知受信待ち状態の CUP のポート番号
出口	なし	－	－	－	－

（凡例）－：該当なし

注※

hostname 引数に 65 バイト以上のホスト名を指定した場合，長さは指定したホスト名のバイト数になります。また，以降の項目の位置は，（hostname 引数で指定したホスト名のバイト数 － 64）分だけずれて出力されます。

表 2-35　openNotification メソッドの各種情報

取得場所	項目	位置		長さ（バイト）	内容
		16 進数	10 進数		
入り口	サーバからの通知メッセージを受信するポート番号	0	0	4	port 引数で指定した，サーバからの通知メッセージを受信するポート番号
出口	なし	−	−	−	−

（凡例）−：該当なし

表 2-36　acceptNotificationChained メソッドの各種情報

取得場所	項目	位置		長さ（バイト）	内容
		16 進数	10 進数		
入り口	サーバからの通知メッセージを格納する領域の長さ	0	0	4	inf_len 引数で指定した，サーバからの通知メッセージを格納する領域の長さ
	タイムアウト値	4	4	4	timeout 引数で指定した，タイムアウト値
出口	サーバからの通知メッセージ	0	0	60	inf 引数に格納した，サーバからの通知メッセージ
	サーバからの通知メッセージの長さ	3C	60	4	inf_len 引数に格納した，サーバからの通知メッセージの長さ
	サーバのホスト名	40	64	64	hostname 引数に格納した，サーバのホスト名
	サーバのノード識別子	80	128	8	nodeid 引数に格納した，サーバのノード識別子

表 2-37　setUapTraceMode メソッドの各種情報

取得場所	項目	位置		長さ（バイト）	内容
		16 進数	10 進数		
入り口	UAP トレースを出力するディレクトリ	0	0	256※	TrcPath 引数で指定した，UAP トレースを出力するディレクトリ
	出力する UAP トレースファイルのサイズ	100	256	4	size 引数で指定した，出力する UAP トレースファイルのサイズ
	UAP トレースの取得可否フラグ	104	260	4	flag 引数で指定した，UAP トレースの取得可否フラグ（flag 引数=true の場合は 1，flag 引数=false の場合は 0 が出力される）
出口	なし	−	−	−	−

（凡例）−：該当なし

注※

TrcPath 引数に 257 バイト以上のパスを指定した場合，長さは指定したパスのバイト数になります。また，以降の項目の位置は，（TrcPath 引数で指定したパスのバイト数 − 256）分だけずれて出力されます。

表 2-38　setErrorTraceMode メソッドの各種情報

取得場所	項目	位置		長さ（バイト）	内容
		16 進数	10 進数		
入り口	エラートレースを出力するディレクトリ	0	0	256※	TrcPath 引数で指定した，エラートレースを出力するディレクトリ
	出力するエラートレースファイルのサイズ	100	256	4	size 引数で指定した，出力するエラートレースファイルのサイズ
	エラートレースの取得可否フラグ	104	260	4	flag 引数で指定した，エラートレースの取得可否フラグ(flag 引数=true でならば 1，flag 引数=false ならば 0 が出力される。)
出口	なし	−	−	−	−

（凡例）−：該当なし

注※

TrcPath 引数に 257 バイト以上のパスを指定した場合，長さは指定したパスのバイト数になります。また，以降の項目の位置は，（TrcPath 引数で指定したパスのバイト数 − 256）分だけずれて出力されます。

表 2-39　setMethodTraceMode メソッドの各種情報

取得場所	項目	位置		長さ（バイト）	内容
		16 進数	10 進数		
入り口	メソッドトレースを出力するディレクトリ	0	0	256※	TrcPath 引数で指定した，メソッドトレースを出力するディレクトリ
	出力するメソッドトレースファイルのサイズ	100	256	4	size 引数で指定した，出力するメソッドトレースファイルのサイズ
	メソッドトレースの取得可否フラグ	104	260	4	flag 引数で指定した，メソッドトレースの取得可否フラグ（flag 引数=true の場合は 1，flag 引数=false の場合は 0 が出力される）
出口	なし	−	−	−	−

（凡例）−：該当なし

注※

TrcPath 引数に 257 バイト以上のパスを指定した場合，長さは指定したパスのバイト数になります。また，以降の項目の位置は，（TrcPath 引数で指定したパスのバイト数 − 256）分だけずれて出力されます。

表 2-40　setDataTraceMode メソッドの各種情報

取得場所	項目	位置		長さ（バイト）	内容
		16 進数	10 進数		
入り口	データトレースを出力するディレクトリ	0	0	256※	TrcPath 引数で指定した，データトレースを出力するディレクトリ
	出力するデータトレースファイルのサイズ	100	256	4	size 引数で指定した，出力するデータトレースファイルのサイズ
	データトレースに出力するデータサイズ	104	260	4	DataSize 引数で指定した，データトレースに出力するデータサイズ
	データトレースの取得可否フラグ	108	264	4	flag 引数で指定した，データトレースの取得可否フラグ（flag 引数=true の場合は 1，flag 引数=false の場合は 0 が出力される）
出口	なし	－	－	－	－

（凡例）－：該当なし

注※

TrcPath 引数に 257 バイト以上のパスを指定した場合，長さは指定したパスのバイト数になります。また，以降の項目の位置は，（TrcPath 引数で指定したパスのバイト数 － 256）分だけずれて出力されます。

表 2-41　setRpcServicePrio メソッドの各種情報

取得場所	項目	位置		長さ（バイト）	内容
		16 進数	10 進数		
入り口	プライオリティ	0	0	4	prio 引数で指定したプライオリティ
出口	なし	－	－	－	－

（凡例）－：該当なし

表 2-42　getRpcServicePrio メソッドの各種情報

取得場所	項目	位置		長さ（バイト）	内容
		16 進数	10 進数		
入り口	なし	－	－	－	－
出口	プライオリティ	0	0	4	設定されているサービス要求のプライオリティ

（凡例）－：該当なし

表 2-43　trnChainedCommit メソッドの各種情報

取得場所	項目	位置		長さ（バイト）	内容
		16 進数	10 進数		
入り口	なし	−	−	−	−
出口	性能解析トレースの識別情報	0	0	26	性能解析トレースの識別情報

（凡例）−：該当なし

表 2-44　trnUnchainedCommit メソッドの各種情報

取得場所	項目	位置		長さ（バイト）	内容
		16 進数	10 進数		
入り口	なし	−	−	−	−
出口	性能解析トレースの識別情報	0	0	26	性能解析トレースの識別情報

（凡例）−：該当なし

表 2-45　trnChainedRollback メソッドの各種情報

取得場所	項目	位置		長さ（バイト）	内容
		16 進数	10 進数		
入り口	なし	−	−	−	−
出口	性能解析トレースの識別情報	0	0	26	性能解析トレースの識別情報

（凡例）−：該当なし

表 2-46　trnUnchainedRollback メソッドの各種情報

取得場所	項目	位置		長さ（バイト）	内容
		16 進数	10 進数		
入り口	なし	−	−	−	−
出口	性能解析トレースの識別情報	0	0	26	性能解析トレースの識別情報

（凡例）−：該当なし

2.11.3 データトレース

データトレース情報は，setDataTraceMode メソッドの TrcPath 引数で指定したディレクトリ，または TP1/Client/J 環境定義の dcdatatracepath オペランドで指定したディレクトリに，dcdat1.trc および dcdat2.trc というファイル名で出力されます。ファイルのサイズは setDataTraceMode メソッドの size 引数，または TP1/Client/J 環境定義の dcdatatracesize オペランドで指定します。1 トレースのデータサイズは setDataTraceMode メソッドの DataSize 引数，または TP1/Client/J 環境定義の dcdatatracemaxsize オペランドで指定します。

データトレースは，CUP と TP1/Server 間の送受信メッセージの内容を取得します。

データトレースファイルへの出力形式を次に示します。

```
yyyy/mm/dd hh:mm:ss.uuu Event = eeee ThreadName = ttt
ADDRESS +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +a +b +c +d +e +f 0123456789abcdef
00000000 07 70 c0 00 00 00 00 5c 00 00 00 00 00 06 00 05 .p.....¥........
00000010 00 00 00 10 00 00 00 00 00 00 00 00 15 bf 00 00 ................
00000020 00 00 00 00 00 00 00 00 00 00 00 00 00 01 01 00 ................
:         :         :
```

- yyyy/mm/dd hh:mm:ss.uuu：データトレースを取得した日時
- eeee：送受信種別
- ttt：実行スレッド名
- ADDRESS 以降の文字列：データ

2.11.4 エラートレース，メモリトレース

エラートレース情報は，setErrorTraceMode メソッドの TrcPath 引数で指定したディレクトリ，または TP1/Client/J 環境定義の dcerrtracepath オペランドで指定したディレクトリに，dcerr1.trc および dcerr2.trc というファイル名で出力されます。ファイルのサイズは setErrorTraceMode メソッドの size 引数，または TP1/Client/J 環境定義の dcerrtracesize オペランドで指定します。

エラートレースは，TP1/Client/J が検知した障害をメッセージ形式でファイルに記録します。メソッド実行中に障害が発生した場合，その原因は例外として報告されますが，それだけでは原因を特定できない場合があります。エラートレースを取得すると，詳細なエラー情報がファイルに出力され，原因調査の資料として使用できます。

メモリトレースはエラートレースを取得できない Java アプレットを使用しているときに，setTraceArray メソッドで指定された String 配列にエラー情報を格納する機能です。障害発生時に String 配列を参照するとエラー情報を取得できます。

エラートレースファイルの出力形式，またはメモリトレースの String 配列への格納形式を次に示します。

```
(ttt)yyyy/mm/dd hh:mm:ss.uuu eeeeeeeeeeee
```

- ttt：実行スレッド名
- yyyy/mm/dd hh:mm:ss.uuu：エラートレースまたはメモリトレースを取得した日時
- eeeeeeeeeeee：メッセージ

取得するメッセージを次の表に示します。

表 2-47　エラートレース，メモリトレースで取得するメッセージ

メッセージの内容	意味
Invalid message received. method=aaaaaaaa	TP1/Server から不正なメッセージを受信しました。 aaaaaaaa：このメッセージを出力したメソッド名 　メモリトレースの場合，このメッセージを出力したメソッド名に対応するコード
Error reply received. inf=aaaaaaaa, method=bbbbbbbb	TP1/Server からエラー応答を受け取りました。 aaaaaaaa：受け取ったエラーコード bbbbbbbb：このメッセージを出力したメソッド名 　メモリトレースの場合，このメッセージを出力したメソッド名に対応するコード
Exception occured. inf=aaaaaaaa, exception=bbbbbbbb(cc...cc), method=dddddddd	TP1Client クラス内で Java システムから例外を受け取りました。または TP1Client クラスから Java アプレット，Java アプリケーション，または Java サーブレットに例外を返しました。 aaaaaaaa：例外が発生したときの保守情報 bbbbbbbb：Java から受け取った例外名，または Java アプレット，Java アプリケーション，または Java サーブレットに返した例外名 　メモリトレースの場合，例外名に対応するコード cc...cc：例外の詳細メッセージ 　詳細メッセージがある場合だけ出力します。 dddddddd：このメッセージを出力したメソッド名 　メモリトレースの場合，メソッド名に対応するコード
Invalid data received. (aa...aa), method=bbbbbbbb	cltAssemReceive メソッドで相手システムから不正なデータを受信しました。 aa...aa：不正なデータ 　メッセージ長が不正な場合 　receive message length=メッセージ長（10 進数） bbbbbbbb：このメッセージを出力したメソッド名 　メモリトレースの場合，メソッド名に対応するコード
Receiving message was canceled. aaaaaaaa　(bb...bb) method=cccccccc	cltAssemReceive メソッドで相手システムから受信したメッセージを破棄しました。 aaaaaaaa：メッセージを破棄した原因 bb...bb：データの内容 　受信バッファオーバーフローの場合 　receive buffer overflowed. 　(receive buffer size=受信バッファの大きさ（10 進数）, receive message body length=受信メッセージ本体の長さ（10 進数）)

メッセージの内容	意味
Receiving message was canceled. aaaaaaaa (bb...bb) method=cccccccc	cccccccc：このメッセージを出力したメソッド名 メモリトレースの場合，メソッド名に対応するコード
User data did not compress, group=aa...aa, service=bb...bb, reason= cc...cc	ユーザデータを圧縮しませんでした。データを圧縮しないでサービスを要求します。 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：ユーザデータを圧縮しなかった理由 NO EFFECT：ユーザデータに対する圧縮効果がありません。 NOT SUPPORT VERSION：サービス要求先の TP1/Server がデータ圧縮機能をサポートしていないバージョンです。 「NO EFFECT」の場合，圧縮前より圧縮後のデータの方が大きくなります。そのため，同一の 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を使用するかどうかを再度検討してください。 「NOT SUPPORT VERSION」の場合，サービス要求先の TP1/Server が，データ圧縮機能を使用できるバージョン（TP1/Server Base 03-03 以降）かどうかを確認してください。
Hostname is invalid. method=aa...aa, operand=bb...bb, value=cc...cc	• ホスト名に不正がありました。次のうちのどれかの要因で，指定されたホスト名から名前解決ができません。 a. hosts ファイルや DNS などでホスト名と IP アドレスをマッピングできない b. hosts ファイルのアクセス権限がないため，参照できない c. 一時的なエラーによって DNS サーバへの問い合わせが失敗した • ループバックアドレス（127 で始まる IP アドレス）もしくは，ループバックアドレスに変換されるホスト名を指定しています。 localhost または名前解決した結果が 127 で始まる IP アドレス（例：127.0.0.1）に変換されるホスト名を指定しています。 • オペランドに指定した 10 進ドット記法の IP アドレスに誤りがあります。 aa...aa：このメッセージを出力したメソッド名 bb...bb：エラーが発生したオペランド名 cc...cc：エラーが発生したオペランドの指定値
Definition analysis error occurred. method=aa...aa, operand=bb...bb	定義指定の論理エラーが発生しました。 aa...aa：このメッセージを出力したメソッド名 bb...bb：エラーが発生したオペランド名 dcresponsehost：dcnotifyreshost に Y を指定した場合，dcresponsehost オペランドに指定が必要ですが，値が設定されていません。
Syntax error occurred. method=aa...aa, operand=bb...bb, value=cc...cc	オペランドの指定値が規則に従っていません。 aa...aa：このメッセージを出力したメソッド名 bb...bb：エラーが発生したオペランド名 cc...cc：エラーが発生したオペランドの指定値

メモリトレースに出力されるコードとメソッド名の対応を次の表に示します。

表 2-48　コードとメソッド名の対応

コード	メソッド名
1	TP1Client.openConnection
2	TP1Client.closeConnection
3	TP1Client.rpcCall
4	TP1Client.setDccltinquiretime
5	TP1Client.setDccltdelay
6	TP1Client.setDcwatchtim
7	TP1Client.setDcselint
8	TP1Client.setDccltextend
9	TP1Client.rpcOpen
10	TP1Client.rpcClose
11	TP1Client.setRpcextend
12	TP1Client.setDchost
13	TP1Client.rpcCallTo
14	TP1Client.trnBegin
15	TP1Client.trnChainedCommit
16	TP1Client.trnChainedRollback
17	TP1Client.trnUnchainedCommit
18	TP1Client.trnUnchainedRollback
19	TP1Client.trnInfo
20	TP1Client.getTrnID
37	TP1Client.cltAssemSend
38	TP1Client.cltAssemReceive
100	TP1ClientSocketCommunicator.openConnection
101	TP1ClientSocketCommunicator.closeConnection
102	TP1ClientSocketCommunicator.sendData
103	TP1ClientSocketCommunicator.sendData
104	TP1ClientSocketCommunicator.recvData
105	TP1ClientSocketCommunicator.recvData
106	TP1ClientSocketCommunicator.recvDummyData
107	TP1ClientSocketCommunicator.flush

コード	メソッド名
108	TP1ClientSocketCommunicator.recvSelect
109	TP1ClientSocketCommunicator.openServerSocket
110	TP1ClientSocketCommunicator.acceptServerSocket
111	TP1ClientSocketCommunicator.closeServerSocket
112	TP1ClientSocketCommunicator.getServerPort
113	TP1ClientSocketCommunicator.getLocalIPAddress
114	TP1ClientSocketCommunicator.getLocalPort
300	TP1ClientProperties.TP1ClientProperties
301	TP1ClientProperties.TP1ClientProperties
302	TP1ClientProperties.getValue
400	TP1ClientRpc.rpcOpen
401	TP1ClientRpc.cltConnect
402	TP1ClientRpc.rpcCall
403	TP1ClientRpc.rpcClose
404	TP1ClientRpc.cltDisconnect
405	TP1ClientRpc.defAnalyze
406	TP1ClientRpc.rapConnect
407	TP1ClientRpc.rapDisconnect
408	TP1ClientRpc.rapRpcCall
409	TP1ClientRpc.scdRpcCall
410	TP1ClientRpc.setDccltextend
411	TP1ClientRpc.rapMngConnect
412	TP1ClientRpc.rapMngDisconnect
413	TP1ClientRpc.setRpcextend
414	TP1ClientRpc.setDchost
415	TP1ClientRpc.namRpcCall
416	TP1ClientRpc.getHostEntry
417	TP1ClientRpc.getNextEntry
500	TP1ClientConManage.openMngConnection
501	TP1ClientConManage.closeMngConnection
502	TP1ClientConManage.changeMngConnection

コード	メソッド名
503	TP1ClientConManage.getConnection
504	TP1ClientConManage.putConnection
505	TP1ClientConManage.cancelConnection
506	TP1ClientConManage.registCheck
507	TP1ClientConManage.getMngConInfo
512	TP1ClientConManage.getSync
513	TP1ClientConManage.registConnNum
600	TP1ClientConnectionHost.addTP1ClientConnectionHost
601	TP1ClientConnectionHost.removeTP1ClientConnectionHost
602	TP1ClientConnectionHost.removeTP1ClientConnectionHostAll
603	TP1ClientConnectionHost.changeTP1ClientConnectionHost
604	TP1ClientConnectionHost.getConnection
605	TP1ClientConnectionHost.putConnection
606	TP1ClientConnectionHost.addConnection
700	TP1ClientNam.Lookup
900	TP1ClientTrn.trnBegin
901	TP1ClientTrn.trnChainedCommit
902	TP1ClientTrn.trnChainedRollback
903	TP1ClientTrn.trnUnchainedCommit
904	TP1ClientTrn.trnUnchainedRollback
905	TP1ClientTrn.conTrnCall
1001	Socket.Socket
1002	Socket.getInputStream
1003	Socket.getOutputStream
1004	Socket.close
1005	DataInputStream.read
1006	DataInputStream.available
1007	DataInputStream.close
1008	DataOutputStream.write
1009	DataOutputStream.close
1010	InetAddress.getLocalHost

コード	メソッド名
1013	InputStream.read
1014	InputStream.close
1015	OutputStream.write
1016	OutputStream.close
1019	Socket.setTcpNoDelay
1102	TP1ClientSndRcv.cltAssemSend
1103	TP1ClientSndRcv.cltAssemReceive

メモリトレースに出力されるコードと例外名の対応を次の表に示します。

表 2-49　コードと例外名の対応

コード	例外名
1	ErrInvalidArgsException
2	ErrProtoException
3	ErrNoBufsException
4	ErrNetDownException
5	ErrTimedOutException
6	ErrMessageTooBigException
7	ErrReplyTooBigException
8	ErrNoSuchServiceGroupException
9	ErrNoSuchServiceException
10	ErrServiceClosedException
11	ErrServiceTerminatingException
12	ErrServiceNotUpException
13	ErrNotUpException
14	ErrSyserrAtServerException
15	ErrNoBufsAtServerException
16	ErrSyserrException
17	ErrInvalidReplyException
18	ErrInitializingException
19	ErrServerBusyException
20	ErrTestmodeException

コード	例外名
21	ErrSecchkException
22	ErrServiceTerminatedException
23	ErrIOErrException
24	ErrHostUndefException
25	ErrInvalidPortException
26	ErrConnfreeException
29	ErrFatalException
30	ErrSecurityException
31	NumberFormatException
32	EOFException
33	FileNotFoundException
34	SocketException
35	InterruptedIOException
36	ErrNotPoolingException
45	ErrTrnchkException
76	ErrServerTimedOutException
77	ErrClientTimedOutException
78	ErrNotTrnExtendException
79	ErrTrnchkExtendException
80	ErrNetDownAtServerException
81	ErrNetDownAtClientException
84	ErrInvalidMessageException
85	ErrBufferOverflowException
86	ErrCollisionMessageException
1001	IOException
1002	UnknownHostException

2.11.5 メソッドトレース

メソッドトレース情報は，setMethodTraceMode メソッドの TrcPath 引数，または TP1/Client/J 環境定義の dcmethodtracepath オペランドで指定したディレクトリに，dcmtd1.trc および dcmtd2.trc とい

うファイル名で出力します。ファイルのサイズは setMethodTraceMode メソッドの size 引数，または TP1/Client/J 環境定義の dcmethodtracesize オペランドで指定します。

TP1Client クラス内では多くの内部メソッドが実行されます。メソッドトレースは，これらの内部メソッドの実行順序や実行時刻をファイルに出力します。Java 環境では障害時に取得できる情報が少ないため，どのような内部処理の延長で障害が発生したかを原因調査の材料にします。

メソッドトレースファイルの出力形式を次に示します。

```
yyyy/mm/dd hh:mm:ss.uuu Location = lll ThreadName = ttt
MethodName = nnnnnnnnnnnnnnnnnnnn
ADDRESS +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +a +b +c +d +e +f 0123456789abcdef
00000000 00 00 00 00 c7 0b 49 17 00 00 00 00 00 06 00 05 ......|.........
```

- yyyy/mm/dd hh:mm:ss.uuu：メソッドトレースを取得した日時
- lll：出入口種別
- ttt：実行スレッド名
- ADDRESS 以降の文字列：呼び出したメソッド名

2.11.6 デバッグトレース

デバッグトレース情報は，通常 TP1/Client/J 内のメモリバッファに取得され，例外を返した場合だけデバッグトレースファイルに出力します。デバッグトレースファイルは，Java VM を実行するユーザのホームディレクトリ下の TP1clientJ ディレクトリに，dcCltXXXXXXXXXXXX.dmp（XXXXXXXXXXXX：タイムスタンプ）というファイル名で出力します。

デバッグトレースファイルを出力する時点で，デバッグトレースファイルの総数が，TP1/Client/J 環境定義の dccltdbgtrcfilecount オペランドで指定したファイル数を超える場合は，更新日時の最も古いファイルを削除します。デバッグトレースファイルの総数が，（dccltdbgtrcfilecount オペランドで指定したファイル数－1）となるまで既存のデバッグトレースファイルを削除します。

CUP を Java アプレットとして動作させる場合，古いデバッグトレースファイルの削除に失敗した場合など，デバッグトレースをファイルに出力できないときは，標準出力に出力します。標準出力に出力された場合，ブラウザなどの機能で Java コンソールを開けば，デバッグトレース情報を参照できます。

デバッグトレースファイルの最大ファイルサイズは 1 メガバイトです。デバッグトレースの出力時にファイルサイズが 1 メガバイトを超える場合は，それまでのデバッグトレース情報を削除してから同じファイルに出力します。そのため，それまでのデバッグトレース情報は出力しません。

デバッグトレースの出力形式を次に示します。

```
(ttt)yyyy/mm/dd hh:mm:ss.uuu MethodName = nnnnnnnnnnnnnnnnnnnn(ii) Location = lll
ADDRESS +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +a +b +c +d +e +f 0123456789abcdef
00000000 00 00 00 00 c7 0b 49 17 00 00 00 00 00 06 00 05 ......|.........
```

- ttt：スレッド ID
- yyyy/mm/dd hh:mm:ss.uuu：デバッグトレースを取得した日時
- nnnnnnnnnnnnnnnnnnnn：実行メソッド名
- ii：インスタンス ID
- lll：出入口種別
- ADDRESS 以降の文字列：デバッグトレース情報

2.11.7　性能解析トレース

性能解析トレース（**PRF トレース**）情報とは，Cosminexus Application Server のトレース情報です。性能解析トレースを取得すると，UAP 実行時の，一連の処理の流れおよび処理に掛かった時間を知ることができ，性能解析に必要な情報を採取できます。また，障害が発生したときに，処理がどこまで到達したかを知ることができます。

性能解析トレースの概要および使用方法の詳細については，マニュアル「Cosminexus 機能解説」，マニュアル「Cosminexus システム運用ガイド」を参照してください。ここでは，TP1/Client/J を Cosminexus Application Server 上で動作させた場合の性能解析トレースの使用方法，性能解析トレースの取得情報，およびトレース取得ポイントについて説明します。

(1)　Cosminexus Application Server での性能解析トレースの出力

TP1/Client/J では，TP1/Client/J を Cosminexus Application Server 上で動作させた場合に，Cosminexus Application Server 上で性能解析トレースを出力できます。性能解析トレースは，TP1/Client/J のトレース取得ポイントで取得されます。性能解析トレースを取得する場合は，TP1/Client/J 環境定義に dccltprftrace=Y を指定します。

Cosminexus Application Server 上で性能解析トレースを出力する場合，TP1/Client/J が取得する情報の分だけ，出力される性能解析トレースのサイズが大きくなることを考慮して，性能解析トレースのサイズを指定してください。

(2)　Cosminexus Application Server と OpenTP1 とのトレースの照合

TP1/Server に対して，ネームサービスを使用した RPC，およびスケジューラダイレクト機能を使用した RPC を行う場合は，スケジューラに送信する RPC 電文中に，TP1/Client/J のインスタンスごとにほぼ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，OpenTP1 の性能検証用トレースで出力します。この OpenTP1 の性能検証用トレースと，Cosminexus Application Server の性能解析トレースとを照合することによって，Cosminexus Application Server と OpenTP1 との間の，一連の処理の流れを知ることができます。また，バージョン 07-02 以降の TP1/Server に対して，リモート API 機能を使用した RPC を行う場合は，スケジューラに送信する RPC 電文中への識別情報（IP アドレスなど）を付加できます。

OpenTP1 の性能検証用トレースに識別情報を付加する場合は，TP1/Client/J 環境定義に dccltprfinfosend=Y を指定します。OpenTP1 の性能検証用トレースの詳細については，マニュアル「OpenTP1 解説」を参照してください。

(3) 性能解析トレースの取得情報

TP1/Client/J で取得した情報は，性能解析トレースの付加情報部分に出力されます。性能解析トレースに取得する情報，およびトレースの出力例を次に示します。

性能解析トレースに取得する情報

ノード ID：_Jaa（4 バイトの英数字）

aa：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））

ルート通信通番部分：bbbbbbbb（4 バイトの 16 進数字）

bbbbbbbb：IP アドレス

RPC 通信通番：ccccdddd（4 バイトの 16 進数字）

cccc：ランダムな 2 バイトの 16 進数字

dddd：通信通番

Cosminexus Application Server で性能解析トレースを出力する場合の出力例

```
...     OPT       ASCII
...     16進表示  _Jaa/0xbbbbbbbb/0xccccdddd...
```

OpenTP1 の性能検証用トレースに識別情報を付加する場合の出力例

```
PRF: Rec Node: smpl Run-ID: 0x416f809d Process: 2612 Trace: 11
Event: 0x2002 Time: 2004/10/15 17:17:07 195.000.000 Server-name: _scd
Rc: ********** Client:  - 0xccccdddd     Server: **** Root: _Jaa - 0xbbbbbbbb
Svc-Grp: tst_svg                        Svc: echo
Trn: *
```

(4) 性能解析トレースのトレース取得ポイント

TP1/Client/J の性能解析トレースの，トレース取得ポイントの詳細を，処理ごとに説明します。

(a) rap リスナーとのコネクション確立，切断処理

rap リスナーとのコネクション確立，切断処理でのイベント ID，トレース取得ポイント，付加情報および性能解析トレース取得レベルについて，次の表に示します。

表 2-50 rap リスナーとのコネクション確立，切断処理でのトレース取得ポイントの詳細

イベント ID	図中の番号※1	トレース取得ポイント	付加情報※2	レベル
0x9180	1	rap リスナーへのコネクション確立処理前	要求先の情報（ホスト名，ポート番）	A

イベント ID	図中の番号※1	トレース取得ポイント	付加情報※2	レベル
0x9181	2	rap リスナーへのコネクション確立処理後（エラー時だけ取得）	内部リターンコード	A
0x9182※3	3	rap リスナーへの接続要求データの送受信後	内部リターンコード	A
0x9183	4	rap リスナーへの切断要求データの送受信前	－	A
0x9184	5	rap リスナーへの切断要求データの送受信後	内部リターンコード	A

（凡例）

A：標準

－：該当なし

注※1

図 2-37 中の番号と対応しています。

注※2

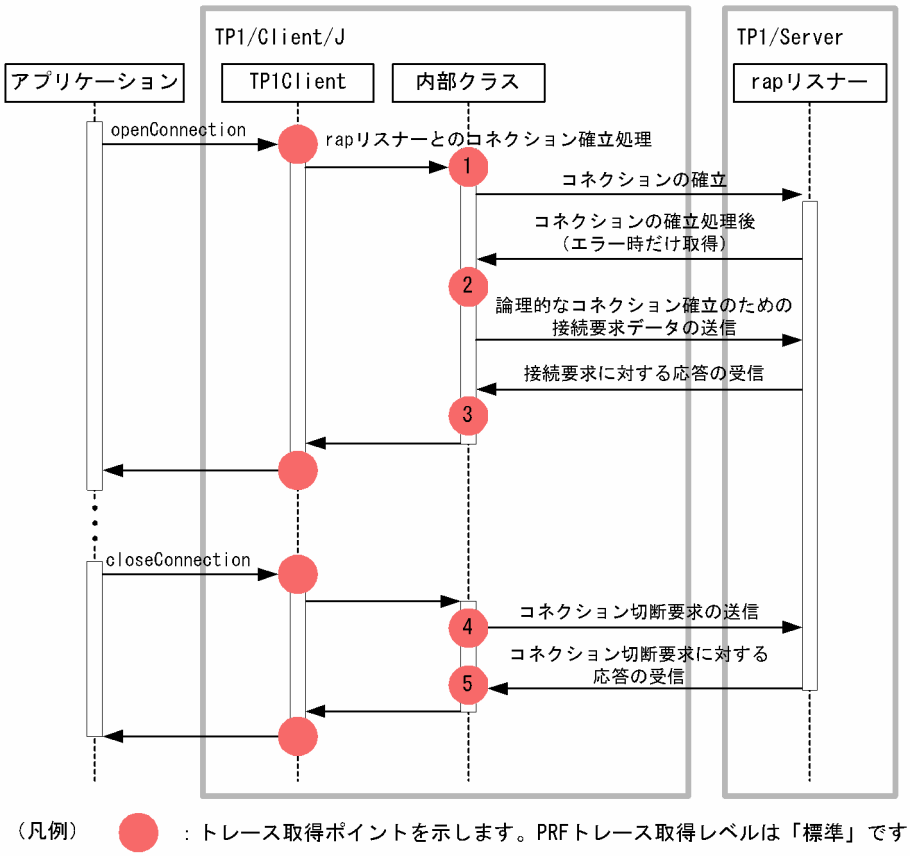
Cosminexus Application Server の性能解析トレースに付加する情報です。

注※3

コネクション確立処理がエラーとなってイベント ID0x9181 が取得された場合，このイベント ID のトレースは取得されません。

rap リスナーとのコネクション確立，切断処理でのトレース取得ポイントを次の図に示します。

図 2-37　rap リスナーとのコネクション確立，切断処理でのトレース取得ポイント



(b) リモート API 機能での API 代理実行要求

リモート API 機能での API 代理実行要求でのイベント ID，トレース取得ポイント，付加情報および性能解析トレース取得レベルについて，次の表に示します。

表 2-51 リモート API 機能での API 代理実行要求でのトレース取得ポイントの詳細

イベント ID	図中の番号※1	トレース取得ポイント	付加情報※2	レベル
0x9190	1	dc_rpc_call の代理実行の要求処理前	－	A
0x9191	2	dc_rpc_call の代理実行の要求処理後	内部リターンコード	A
0x9192	1	dc_trn_begin の代理実行の要求処理前	－	A
0x9193	2	dc_trn_begin の代理実行の要求処理後	内部リターンコード	A
0x9194	1	dc_trn_chained_commit の代理実行の要求処理前	－	A
0x9195	2	dc_trn_chained_commit の代理実行の要求処理後	内部リターンコード	A
0x9196	1	dc_trn_chained_rollback の代理実行の要求処理前	－	A
0x9197	2	dc_trn_chained_rollback の代理実行の要求処理後	内部リターンコード	A
0x9198	1	dc_trn_unchained_commit の代理実行の要求処理前	－	A
0x9199	2	dc_trn_unchained_commit の代理実行の要求処理後	内部リターンコード	A
0x919A	1	dc_trn_unchained_rollback の代理実行の要求処理前	－	A
0x919B	2	dc_trn_unchained_rollback の代理実行の要求処理後	内部リターンコード	A

（凡例）

A：標準

－：該当なし

注※1

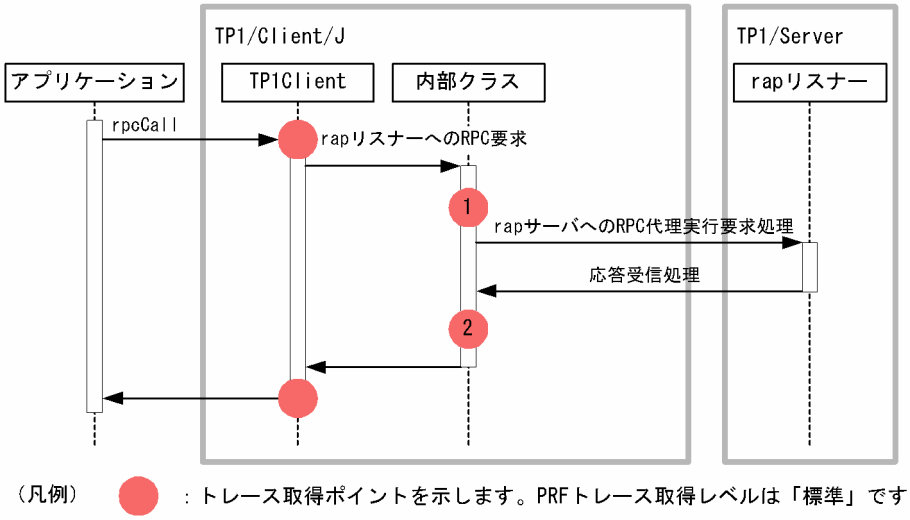
図 2-38 中の番号と対応しています。

注※2

Cosminexus Application Server の性能解析トレースに付加する情報です。

リモート API 機能での API 代理実行要求でのトレース取得ポイントを次の図に示します。

図 2-38　リモート API 機能での API 代理実行要求でのトレース取得ポイント



(c)　スケジュールサーバへの RPC

スケジュールサーバへの RPC でのイベント ID，トレース取得ポイント，付加情報および性能解析トレース取得レベルについて，次の表に示します。

表 2-52　スケジュールサーバへの RPC でのトレース取得ポイントの詳細

イベント ID	図中の番号※1	トレース取得ポイント	付加情報※2	レベル
0x91C0	1	スケジュールサーバへのコネクション確立処理前	サービス要求先ホストの情報（ホスト名，ポート番号）	A
0x91C1	2	スケジュールサーバへの RPC のサービス要求処理の実行後	内部リターンコード	A
0x91C2※3	3	SPP またはスケジュールサーバからの応答データを受け取るコネクションの接続要求を受け付け後	内部リターンコード※4	A
0x91C3	4	ネームサーバへのコネクション確立処理前	サービス要求先ホストの情報（ホスト名，ポート番号）	B
0x91C4	5	ネームサーバへのサービス情報の問い合わせ処理の実行後	内部リターンコード	B
0x91C5※3	6	ネームサーバからの応答データを受け取るコネクションの接続要求受け付け後	内部リターンコード※5	B

（凡例）

A：標準

B：詳細

注※1

図 2-39 および図 2-40 中の番号と対応しています。

注※2

Cosminexus Application Server の性能解析トレースに付加する情報です。

注※3

要求データの送信が正常終了した場合に，このイベント ID のトレースは取得されます。

注※4

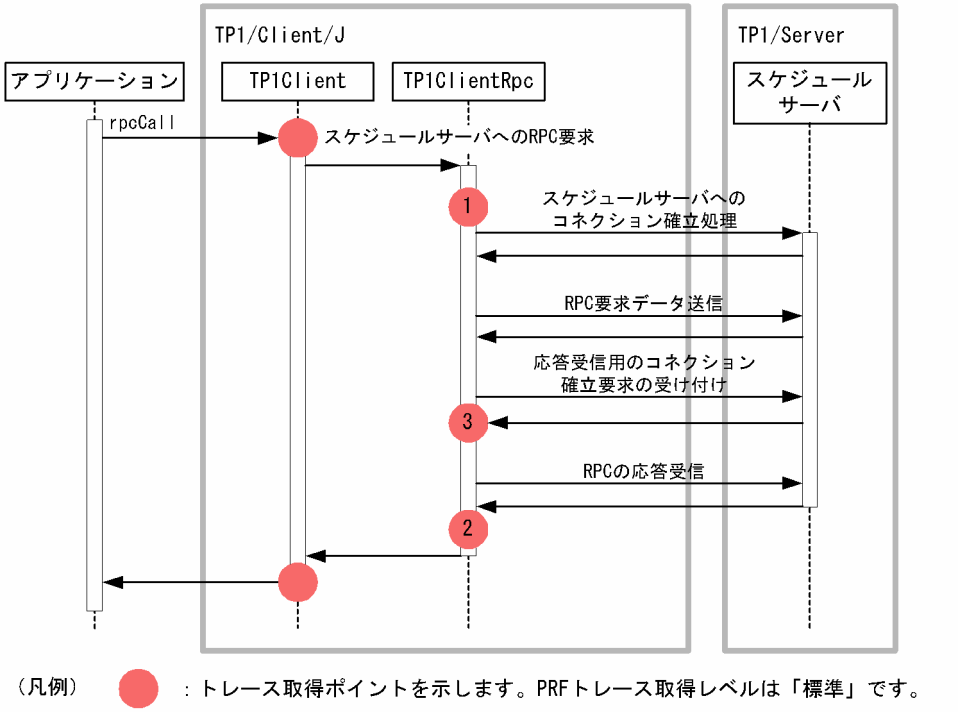
SPP またはスケジュールサーバからの応答データを受け取るコネクションの接続要求の受け付けが正常終了した場合は，接続元情報（IP アドレス，ポート番号）も付加します。

注※5

ネームサーバからの応答データを受け取るコネクションの接続要求の受け付けが正常終了した場合は，接続元情報（IP アドレス，ポート番号）も付加します。

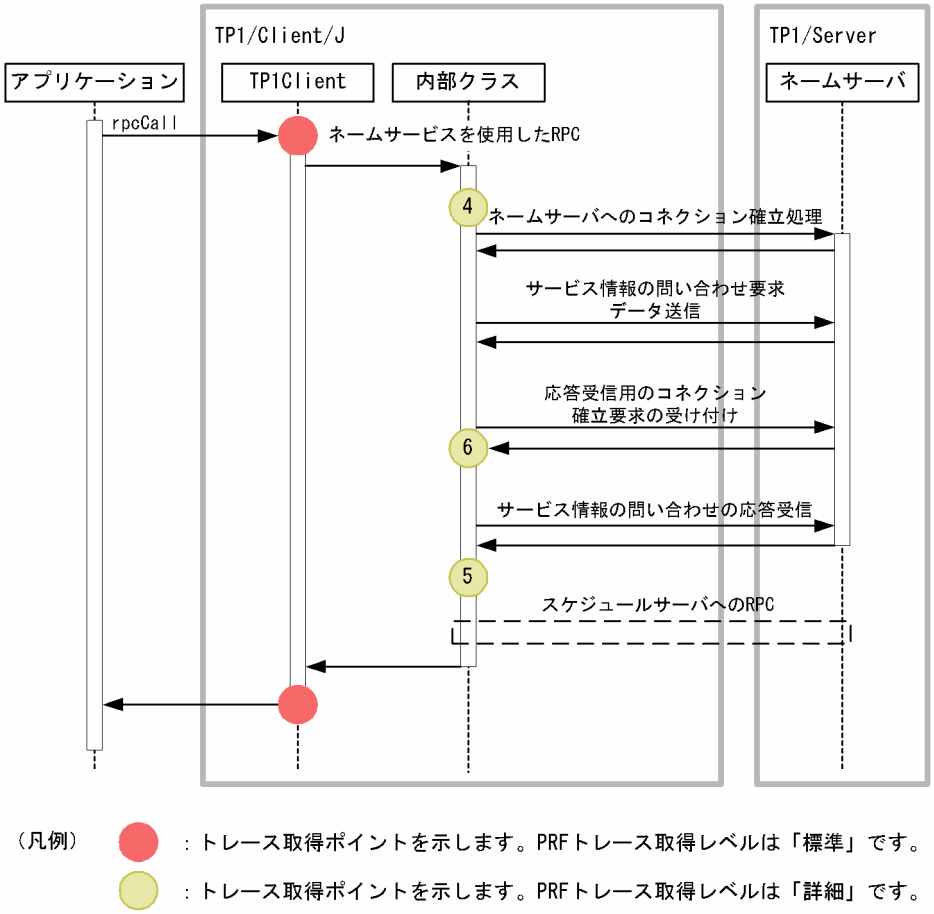
スケジュールサーバへの RPC 要求でのトレース取得ポイントを次の図に示します。

図 2-39　スケジュールサーバへの RPC 要求でのトレース取得ポイント



ネームサーバへのサービス情報の問い合わせでのトレース取得ポイントを次の図に示します。

図 2-40　ネームサーバへのサービス情報の問い合わせでのトレース取得ポイント



(d)　API 実行処理

API 実行処理でのイベント ID，トレース取得ポイント，付加情報および性能解析トレース取得レベルについて，次の表に示します。

表 2-53　API 実行処理でのトレース取得ポイントの詳細

イベント ID	トレース取得ポイント	付加情報※1	レベル
0x91D0	openConnection メソッドの入り口	−	A
0x91D1	openConnection メソッドの出口	内部リターンコード	A
0x91D2	closeConnection メソッドの入り口	−	A
0x91D3	closeConnection メソッドの出口	内部リターンコード	A
0x91D4	rpcCall メソッドの入り口	inlen	A
0x91D5	rpcCall メソッドの出口	outlen，内部リターンコード	A
0x91D6	rpcCallTo メソッドの入り口	inlen	A
0x91D7	rpcCallTo メソッドの出口	outlen，内部リターンコード	A

イベント ID	トレース取得ポイント	付加情報※1	レベル
0x91D8	trnBegin メソッドの入り口	−	A
0x91D9	trnBegin メソッドの出口	内部リターンコード※2	A
0x91DA	trnChainedCommit メソッドの入り口	−	A
0x91DB	trnChainedCommit メソッドの出口	内部リターンコード※2	A
0x91DC	trnChainedRollback メソッドの入り口	−	A
0x91DD	trnChainedRollback メソッドの出口	内部リターンコード※2	A
0x91DE	trnUnchainedCommit メソッドの入り口	−	A
0x91DF	trnUnchainedCommit メソッドの出口	内部リターンコード	A
0x91E0	trnUnchainedRollback メソッドの入り口	−	A
0x91E1	trnUnchainedRollback メソッドの出口	内部リターンコード	A

（凡例）

A：標準

−：該当なし

注※1

Cosminexus Application Server の性能解析トレースに付加する情報です。

注※2

トランザクションの開始時は，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も付加します。

(e) XA リソースサービス機能

XA リソースサービス機能でのイベント ID，トレース取得ポイント，付加情報および PRF トレース取得レベルについて，次の表に示します。

表 2-54 XA リソースサービス機能でのトレース取得ポイントの詳細

イベント ID	トレース取得ポイント	付加情報	レベル
0x919C	TP1/Client/J からの XA トランザクションの start 要求の代理実行要求処理前	内部コード	A
0x919D	TP1/Client/J からの XA トランザクションの start 要求の代理実行要求処理後	内部リターンコード	A
0x919E	TP1/Client/J からの XA トランザクションの end 要求の代理実行要求処理前	−	A
0x919F	TP1/Client/J からの XA トランザクションの end 要求の代理実行要求処理後	内部リターンコード	A
0x91A0	TP1/Client/J からの XA トランザクションの prepare 要求の代理実行要求処理前	−	A
0x91A1	TP1/Client/J からの XA トランザクションの prepare 要求の代理実行要求処理後	内部リターンコード	A

イベント ID	トレース取得ポイント	付加情報	レベル
0x91A2	TP1/Client/J からの XA トランザクションの commit 要求の代理実行要求処理前	−	A
0x91A3	TP1/Client/J からの XA トランザクションの commit 要求の代理実行要求処理後	内部リターンコード	A
0x91A4	TP1/Client/J からの XA トランザクションの rollback 要求の代理実行要求処理前	−	A
0x91A5	TP1/Client/J からの XA トランザクションの rollback 要求の代理実行要求処理後	内部リターンコード	A
0x91A6	TP1/Client/J からの XA トランザクションの forget 要求の代理実行要求処理前	−	A
0x91A7	TP1/Client/J からの XA トランザクションの forget 要求の代理実行要求処理後	内部リターンコード	A
0x91A8	TP1/Client/J からの XA トランザクションの recover 要求の代理実行要求処理前	−	A
0x91A9	TP1/Client/J からの XA トランザクションの recover 要求の代理実行要求処理後	内部リターンコード	A
0x91AA	TP1/Client/J からの XA トランザクションの RPC 要求の代理実行要求処理前	−	A
0x91AB	TP1/Client/J からの XA トランザクションの RPC 要求の代理実行要求処理後	内部リターンコード	A
0x91E2	TP1 Connector からの XA トランザクションの start 要求受け付け直後	内部コード	A
0x91E3	TP1 Connector からの XA トランザクションの start 要求応答直前	内部リターンコード	A
0x91E4	TP1 Connector からの XA トランザクションの end 要求受け付け直後	内部コード	A
0x91E5	TP1 Connector からの XA トランザクションの end 要求応答直前	内部リターンコード	A
0x91E6	TP1 Connector からの XA トランザクションの prepare 要求受け付け直後	内部コード	A
0x91E7	TP1 Connector からの XA トランザクションの prepare 要求応答直前	内部リターンコード	A
0x91E8	TP1 Connector からの XA トランザクションの commit 要求受け付け直後	内部コード	A
0x91E9	TP1 Connector からの XA トランザクションの commit 要求応答直前	内部リターンコード	A
0x91EA	TP1 Connector からの XA トランザクションの rollback 要求受け付け直後	内部コード	A

イベント ID	トレース取得ポイント	付加情報	レベル
0x91EB	TP1 Connector からの XA トランザクションの rollback 要求応答直前	内部リターンコード	A
0x91EC	TP1 Connector からの XA トランザクションの forget 要求受け付け直後	内部コード	A
0x91ED	TP1 Connector からの XA トランザクションの forget 要求応答直前	内部リターンコード	A
0x91EE	TP1 Connector からの XA トランザクションの recover 要求受け付け直後	内部コード	A
0x91EF	TP1 Connector からの XA トランザクションの recover 要求応答直前	内部リターンコード	A
0x91F0	TP1 Connector からの XA トランザクションの RPC 要求受け付け直後	内部コード	A
0x91F1	TP1 Connector からの XA トランザクションの RPC 要求応答直前	内部リターンコード	A

（凡例）

A：標準

－：該当なし

(5) UAP トレースに出力される性能解析トレースの識別情報

TP1/Client/J では，TP1/Server や Cosminexus の性能解析トレースの情報と，TP1/Client/J のトレースの情報を結び付けるために，UAP トレースの出口情報の末尾に，性能解析トレースの識別情報を出力します。

UAP トレースに性能解析トレースの識別情報を出力する，メソッドを次に示します。

- openConnection()
- openConnection(String host, int port)
- closeConnection()
- rpcCall(String group, String service, byte[] in_data, int[] in_len, byte[] out_data, int[] out_len, int flags)
- rpcCall(String group, String service, byte[] in_data, byte[] out_data, int flags)
- rpcCallTo(DCRpcBindTbl direction, String group, String service, byte[] in_data, int[] in_len, byte[] out_data, int[] out_len, int flags)
- trnBegin()
- trnChainedCommit()
- trnUnchainedCommit()
- trnChainedRollback()

- trnUnchainedRollback()

なお，性能解析トレースの識別情報は，TP1/Client/J 環境定義 dccltprfinfosend オペランドの指定値に関係なく出力されます。

性能解析トレースの識別情報を出力した，UAP トレースの出力例を次に示します。太字で示す部分が，性能解析トレースの識別情報です。

```
2008/09/01 16:37:24.698  Location = Out  ThreadName = main
MethodName = TP1Client.rpcCall
Exception  =
ADDRESS   +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +a +b +c +d +e +f  0123456789abcdef
00000000  00 00 00 1c 53 75 7a 75 6b 69 00 00 00 00 00 00  ....Suzuki......
00000010  00 00 00 00 00 00 00 00 00 00 00 00 51 00 00 00  ............Q...
00000020  5f 4a 30 6d 2f 30 78 30 61 64 31 30 66 37 63 2f  _J0m/0x0ad10f7c/
00000030  30 78 32 62 35 64 30 30 30 31                    0x2b5d0001
```

2.12　データ圧縮機能

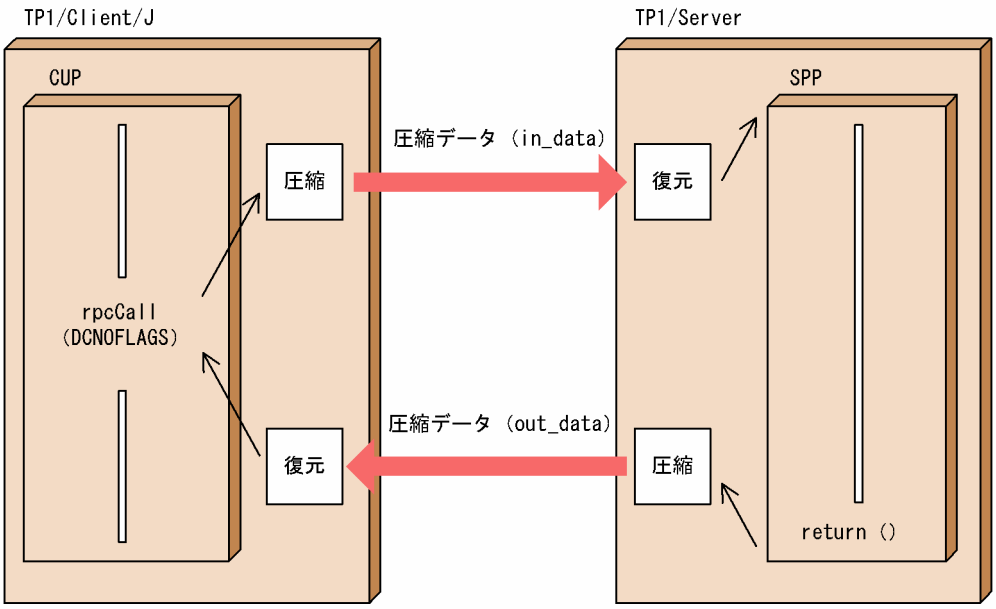
データ圧縮機能を使用すると，RPC によってネットワーク上に送り出されるユーザデータを圧縮できます。これによってネットワーク上に送り出されるパケット数を削減し，ネットワークの混雑を緩和できます。

データ圧縮機能を使用するかどうかは，TP1/Client/J 環境定義の dccltdatacomp オペランドで指定します。

この機能を使用すると，TP1/Client/J は CUP から実行する rpcCall メソッドで設定する入力パラメタ（in_data）の値を，圧縮効果がある場合は圧縮してネットワーク上に送り出します。その問い合わせに対し，SPP から返される応答（out_data）の値も 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/J は，圧縮データを復元して CUP に渡します。

データ圧縮機能の概要を次の図に示します。

図 2-41　データ圧縮機能の概要



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

この機能は，サービス要求先の TP1/Server が 03-03 以降※の場合に使用できます。また，RPC の通信形態，およびすべての RPC インタフェースで使用できます。ただし，サービス要求先の TP1/Server のバージョンによって，次の点が異なります。

- TP1/Server のバージョンが 03-06 未満の場合

  入力パラメタの値が圧縮されていなかった場合，返される応答の値に圧縮効果があっても，応答の値は TP1/Server で圧縮されません。

- TP1/Server のバージョンが 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 で圧縮されます。

注※

リモート API 機能を使用した RPC の場合は，TP1/Server のバージョンが 03-05 以降のときにデータ圧縮機能を使用できます。

また，uCosminexus TP1 Connector または Cosminexus TP1 Connector から XA トランザクション連携機能を使用している場合は，TP1/Server のバージョンが 07-00 以前のとき，データ圧縮機能を使用できません。

2.12.1 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。

また，データ圧縮／復元のためのオーバ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。

2.12.2 データ圧縮機能を使用するときの注意事項

データ圧縮機能を使用する場合の注意事項を次に示します。

- スケジューラダイレクト機能を使用した RPC でサービス要求先の TP1/Server のバージョンが 03-03 未満である場合，RPC を行ったときに，SPP のサービス関数に不正なデータを渡すおそれがあります。したがって，この場合，TP1/Server に対して，データ圧縮機能を使用してサービスを要求しないでください（dccltdatacomp オペランドに N を指定してください）。

  なお，ネームサービス機能を使用した RPC でサービス要求先の TP1/Server のバージョンが 03-03 未満である場合，データ圧縮機能を使用するように設定しても，この設定は無視されるためデータ圧縮機能は使用できません。
- 圧縮前より圧縮後のデータの方がデータが大きくなってしまう場合やサービス要求先の TP1/Server がデータ圧縮機能をサポートしていない場合は，TP1/Client/J のエラートレースファイルにデータを圧縮しなかった旨のメッセージが出力され処理は続行されます。この場合，データは圧縮されません。
- DCCM3 の論理端末に常設コネクションを確立しサービスを要求する場合，データ圧縮機能は使用できません。
- rpcCall メソッドがエラーを返した場合のサービスの応答（out_data）は保証しません。
- RPC メッセージの最大長（rpcCall メソッドで送受信できるユーザメッセージの最大長）は，データ圧縮機能の使用に関係なく，常に dccltrpcmaxmsgsize オペランドで指定した値か，または省略時仮定値（1 メガバイト）です。
- uCosminexus TP1 Connector または Cosminexus TP1 Connector から XA トランザクション連携機能を使用している場合で，通信相手の TP1/Server のバージョンが 07-00 以前のとき，TP1/Client/J ではデータ圧縮機能を使用できません。通信相手の TP1/Server のバージョンが 07-00 以前の場合にデータ圧縮機能を使用すると，KFCA32042-W のメッセージが出力されることがあります。

2.13　送信元ホスト指定機能

TP1/Client/J では，RPC や TCP/IP 通信機能でのコネクション確立要求時に，CUP の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，次のコネクション確立要求時の送信元ホストを固定できます。

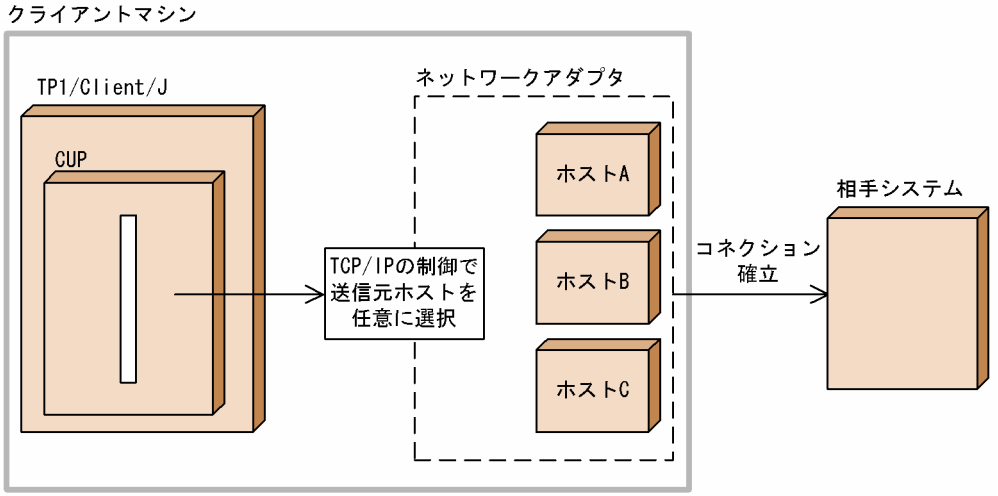
- スケジューラダイレクト機能を使用した RPC
- ネームサービスを使用した RPC
- リモート API 機能を使用した RPC
- 通信先を指定した RPC
- TCP/IP 通信機能

CUP の送信元ホストは，TP1/Client/J 環境定義の dccltcupsndhost オペランドで指定します。

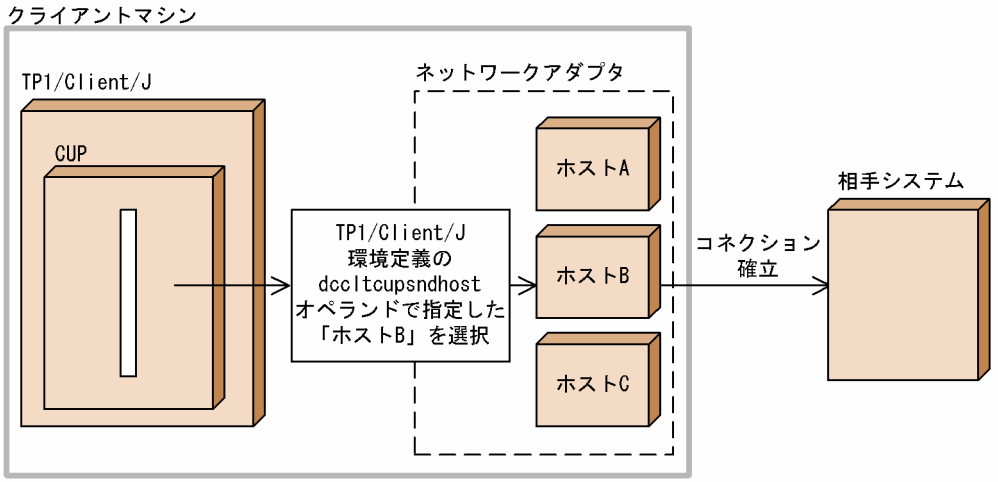
送信元ホスト指定機能を使用しない場合と使用する場合について，次の図に示します。

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

●送信元ホスト指定機能を使用しない場合



●送信元ホスト指定機能を使用する場合



2.14　受信ポート固定機能

TP1/Client/J では，スケジューラダイレクト機能を使用した RPC の受信ポート，およびネームサービスを使用した RPC 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/J への RPC の応答通信をする場合で，TP1/Server と TP1/Client/J との間に設置したファイアウォールで TP1/Client/J 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，TP1/Client/J 環境定義の dccltcuprcvport オペランドを指定します。

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/J 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，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）

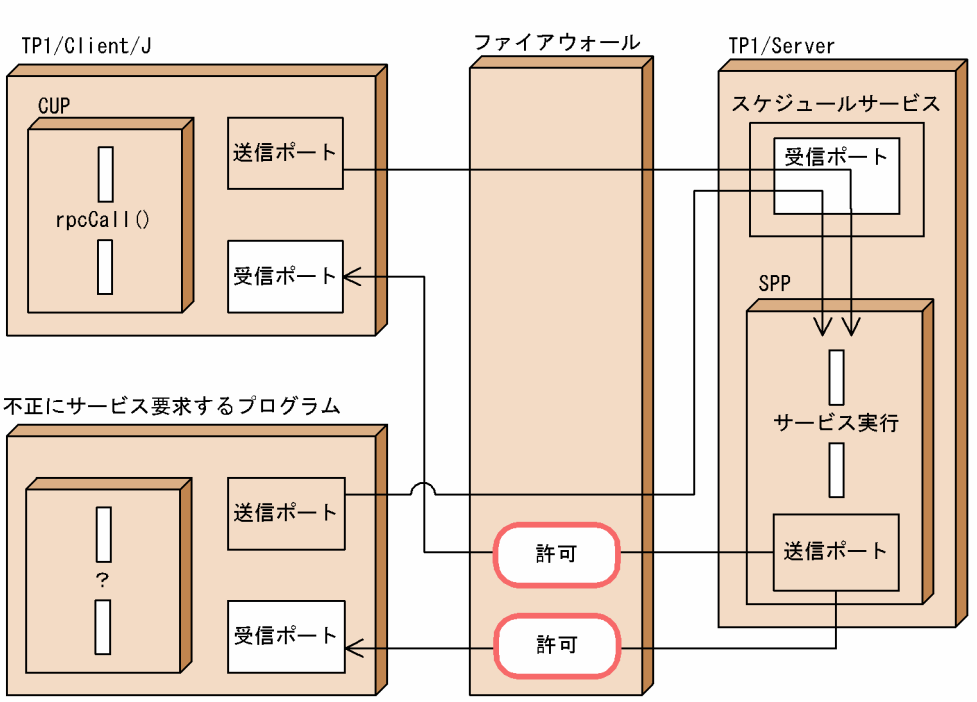
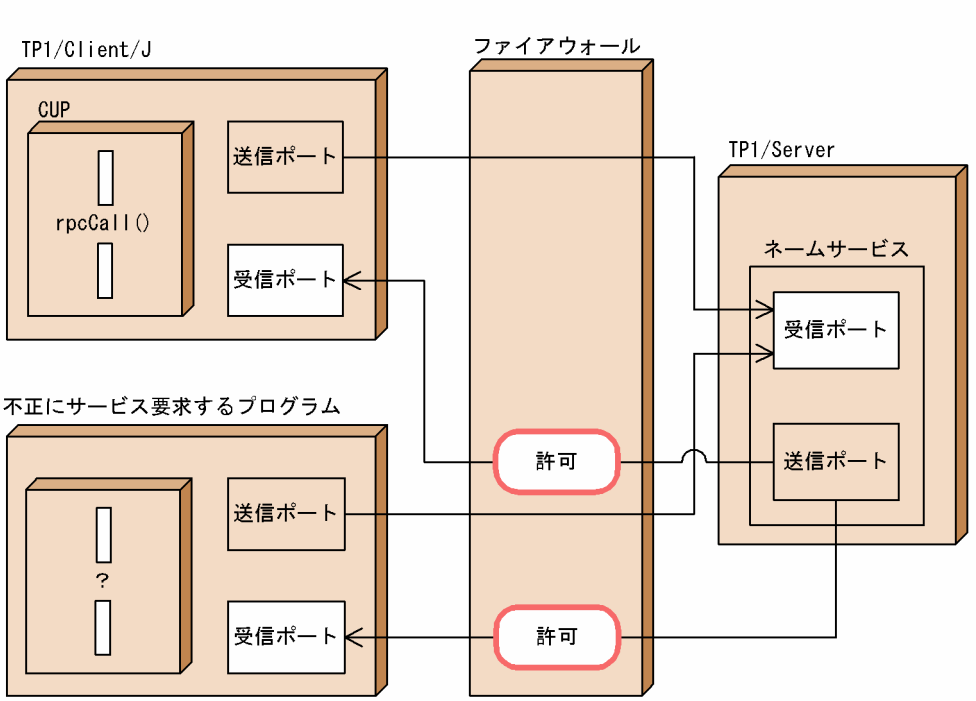


図 2-44　受信ポート固定機能を使用しない場合（ネームサービスを使用した RPC）



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，TP1/Client/J 環境定義の dccltcuprcvport オペランド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，次の図に示します。

図 2-45　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）

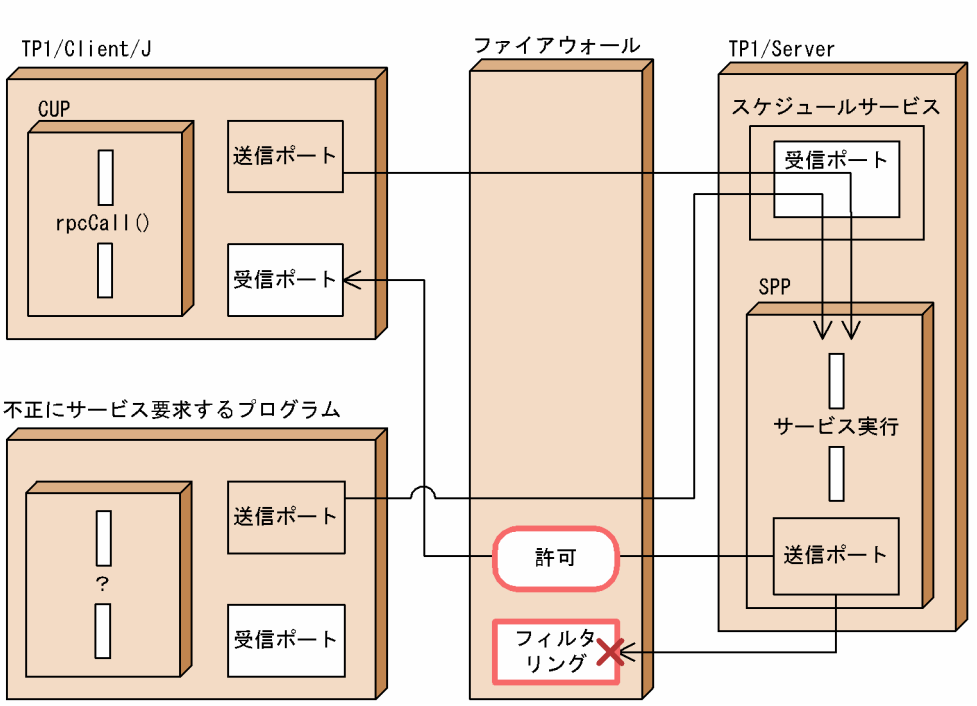
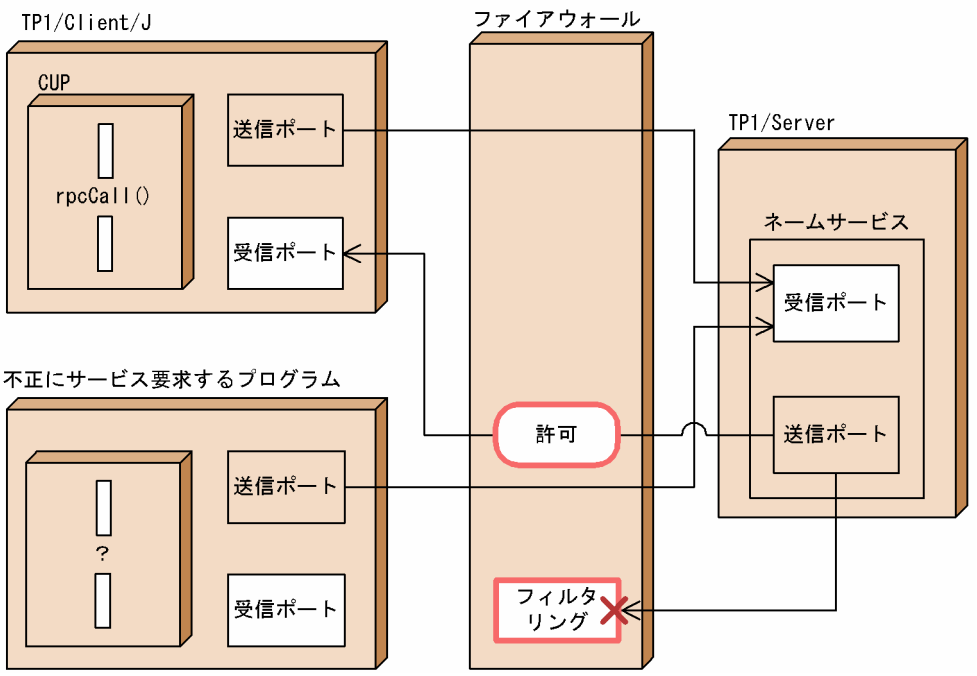


図 2-46　受信ポート固定機能を使用する場合（ネームサービスを使用した RPC）



2.15　ホスト切り替え機能

サーバダウンなどの理由によって，指定したホストとの通信が一時的に不可となった場合，通信先ホストを一つしか指定していないと，TP1/Client/J のメソッドはエラーリターンし，通信先ホストの指定を変更したあと，CUP を再起動する必要があります。

通信先ホストを複数指定しておくと，TP1/Client/J は，ホストを切り替えて，別のホストに通信を試みます。これによって，ユーザは通信先ホストの切り替えを意識することなく業務を継続させることができます。

なお，障害の内容によっては，ホストを切り替えない場合があります。

2.15.1　TP1/Server との通信時

(1)　通信先ホスト複数指定個所

TP1/Client/J 環境定義の dchost オペランドに指定します。

(2)　通信先ホスト選択契機

rpcOpen メソッド実行時に選択します。

接続障害発生時は，rpcCall メソッド実行時にホストを切り替えます。

表 2-55　ホスト切り替えの契機

実行メソッド	ホスト切り替えの契機
rpcCall	ネームサービスとの接続（サービス情報検索）に失敗した場合。 TP1/Client/J 環境定義 dcnamuse=Y のときに該当します。
	スケジュールサービスとの接続に失敗した場合。 TP1/Client/J 環境定義 dcscddirect=Y のときに該当します。

(3)　通信先ホスト選択方法

TP1/Client/J 環境定義 dchostselect オペランドの指定値によって，選択方法が異なります。dchostselect オペランドの指定を省略した場合，または N を指定した場合，dcshost オペランドに指定された順番に通信先ホストを選択します。この指定の場合，TP1/Client/J 環境定義の dchost オペランドに同一の指定を行った複数の CUP から一斉にサービスを要求したとき，一つのホストに負荷が集中してしまいます。dchostselect オペランドに Y を指定すると，初回 rpcCall()メソッド呼び出し時にランダムに通信先ホストを選択するため，サーバの負荷を分散させることができます。

表 2-56　ホスト切り替えの契機

dchostselect	通信先ホストの選択方法	
	初回選択時 （rpcOpen メソッド実行時）	接続障害発生時
N	先頭に指定されたホストを選択します。	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。
Y	複数指定されたホストの中からランダムに選択します。	障害となったホストの次に指定されたホストを選択します。

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。指定されたすべてのホストとの接続に失敗した場合，メソッドはエラーリターンします。

次メソッド発行時，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。

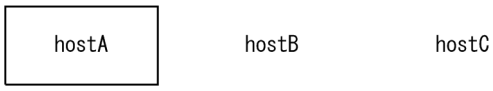
ホスト切り替えによって決定したホストは，次の契機で再びホストの選択，または切り替えが発生するまで，通信先ホストとなります。

- rpcOpen メソッドによるホスト選択
- setDchost メソッドによるホスト選択
- rpcCall メソッドの接続障害で再びホスト切替えが発生

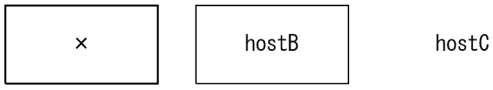
〈例 1〉次の指定の場合

```
dchost=hostA,hostB,hostC
dchostselect=N
```

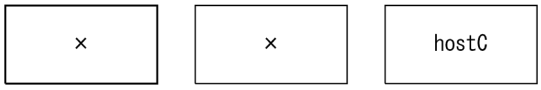
1. rpcOpen メソッド発行
   先頭に指定された hostA を選択



   hostA との接続に成功
2. rpcCall メソッド発行
   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

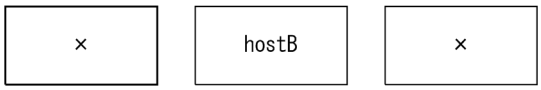
3. rpcCall メソッド発行

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

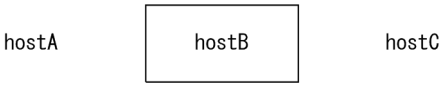


hostB との接続障害発生

選択対象ホストがなくなりメソッドはエラーリターン

4. rpcCall メソッド発行

3.で最後に接続障害が発生したホスト hostB を選択



5. rpcClose メソッド発行

6. rpcOpen メソッド発行

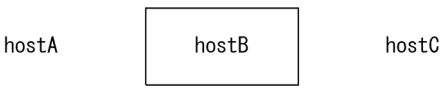
先頭に指定された hostA を選択

〈例 2〉次の指定の場合

```
dchost=hostA,hostB,hostC
dchostselect=Y
```

1. rpcOpen メソッド発行

hostA，hostB，hostC からランダム選択

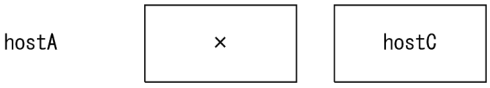


hostB との接続に成功

2. rpcCall メソッド発行

hostB との接続障害発生

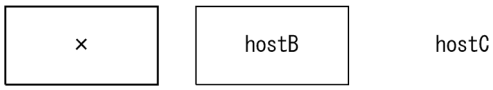
障害となった hostB の次に指定された hostC を選択



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

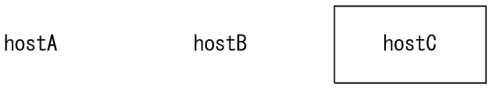
hostA との接続に成功

3. rpcCall メソッド発行
   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
   hostC を選択

   hostC との接続障害発生
   選択対象ホストがなくなりメソッドはエラーリターン
4. rpcCall メソッド発行
   3.で最後に接続障害が発生したホスト hostC を選択



5. rpcClose メソッド発行
6. rpcOpen メソッド発行
   hostA，hostB，hostC からランダム選択

2.15.2　rap リスナー，DCCM3 論理端末との通信時

(1)　通信先ホスト複数指定個所

TP1/Client/J 環境定義の dchost オペランドに指定します。

(2)　通信先ホスト選択契機

rpcOpen メソッド実行時に選択します。

接続障害発生時は，メソッド実行時にホストを切り替えます。

表 2-57　ホスト切り替えの契機

実行メソッド	ホスト切り替えの契機
rpcCall※	rap リスナー，DCCM3 論理端末との接続に失敗した場合。 TP1/Client/J 環境定義 dcrapautoconnect オペランドの指定が Y かつ，dcrapdirect オペランドの指定が Y の時に該当します。
trnBegin openConnection （引数なし）	rap リスナー，DCCM3 論理端末との接続に失敗した場合。 TP1/Client/J 環境定義 dcrapautoconnect オペランドの指定が N かつ，dcrapdirect オペランドの指定が Y の時に該当します。

注※

トランザクションの範囲内で実行した場合は該当しません。

(3)　通信先ホスト選択方法

TP1/Client/J 環境定義 dchostselect オペランドの指定値によって，選択方法が異なります。dchostselect オペランドの指定を省略した場合，または N を指定した場合，dcshost オペランドに指定された順番に通信先ホストを選択します。この指定の場合，TP1/Client/J 環境定義の dchost オペランドに同一の指定を行った複数の CUP から一斉にサービスを要求したとき，一つのホストに負荷が集中してしまいます。dchostselect オペランドに Y を指定すると，rpcOpen 実行時にランダムに通信先ホストを選択するため，サーバの負荷を分散させることができます。

表 2-58　ホスト切り替えの契機

dchostselect	通信先ホストの選択方法	
	初回選択時 （rpcOpen メソッド実行時）	接続障害発生時
N	先頭に指定されたホストを選択します。	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。
Y	複数指定されたホストの中からランダムに選択します。	障害となったホストの次に指定されたホストを選択します。

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。指定されたすべてのホストとの接続に失敗した場合，メソッドはエラーリターンします。

次メソッド発行時，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。

ホスト切り替えによって決定したホストは，次の契機で再びホストの選択，または切り替えが発生するまで，通信先ホストとなります。

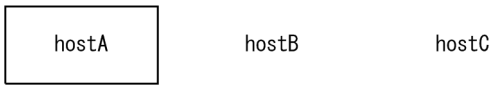
- rpcOpen メソッドによるホスト選択
- setDchost メソッドによるホスト選択
- openConnection，trnBegin，rpcCall メソッドの接続障害で再びホスト切替えが発生

〈例 1〉次の指定の場合

```
dchost=hostA,hostB,hostC
dchostselect=N
```

1. rpcOpen メソッド発行

   先頭に指定された hostA を選択

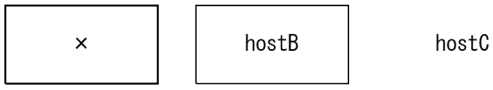


   hostA との接続に成功

2. rpcCall メソッド発行

   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

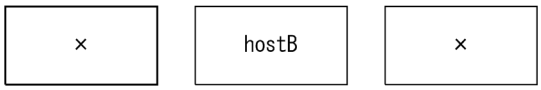
3. rpcCall メソッド発行

   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

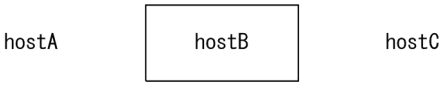


   hostB との接続障害発生

   選択対象ホストがなくなりメソッドはエラーリターン

4. rpcCall メソッド発行

   3.で最後に接続障害が発生したホスト hostB を選択



5. rpcClose メソッド発行

6. rpcOpen メソッド発行

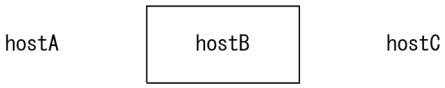
先頭に指定された hostA を選択

〈例 2〉次の指定の場合

```
dchost=hostA,hostB,hostC
dchostselect=Y
```

1. rpcOpen メソッド発行

hostA，hostB，hostC からランダム選択

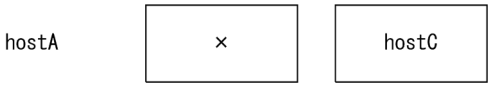


hostB との接続に成功

2. rpcCall メソッド発行

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

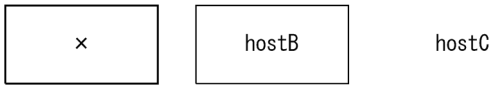
障害となった hostC の次に指定された hostA を選択

hostA との接続に成功

3. rpcCall メソッド発行

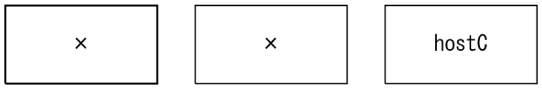
hostA との接続障害発生

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

hostC を選択

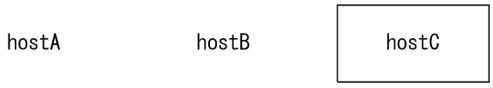


hostC との接続障害発生

選択対象ホストがなくなりメソッドはエラーリターン

4. rpcCall メソッド発行

3.で最後に接続障害が発生したホスト hostC を選択



5. rpcClose メソッド発行
6. rpcOpen メソッド発行
   hostA，hostB，hostC からランダム選択

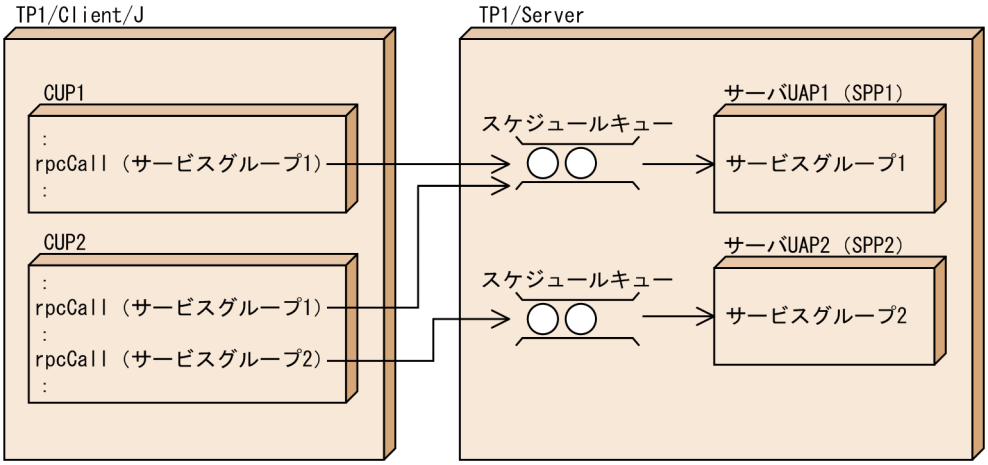
2.16　サービス要求のスケジュールプライオリティ設定機能

CUP からのサービス要求に，優先順位（スケジュールプライオリティ）を付けることができます。一つの処理から呼び出す複数のサービス要求に優先順位を付けたい場合，サービス要求ごとのプライオリティを設定できます。

2.16.1　SPP のスケジュールの方法

OpenTP1 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てスケジュールします。CUP が rpcCall メソッドで指定したサービスグループ名とサービス名を基に，SPP へのサービス要求をスケジュールキューに登録します。登録したサービス要求は，先入れ先出しでスケジュールキューから取り出します。次の図に SPP のスケジュールを示します。

図 2-47　SPP のスケジュール



2.16.2　スケジュールプライオリティの設定

rpcCall メソッドを呼び出す直前に，setRpcServicePrio メソッドを呼び出してサービス要求のプライオリティを設定します。これによって，サービス要求の優先度が，サーバ UAP 側のスケジュールキューを経由してサーバに通知されます。

setRpcServicePrio メソッドを 1 度も使わない場合は，TP1/Server のスケジュールサービスの省略時解釈値である 4 が，サービス要求のプライオリティとして指定されます。設定したスケジュールプライオリティは，getRpcServicePrio メソッドで参照できます。

設定したサービス要求のプライオリティは，サーバ UAP がキュー受信型サーバ（スケジュールサービスでスケジュールされる SPP）であり，かつサーバ UAP のユーザサービス定義に service_priority_control=Y（プライオリティを制御する）を指定している場合だけ有効です。サービスを要求する相手のサーバ UAP でプライオリティを制御していない場合は，このメソッドを呼び出しても

無効になります。OpenTP1 のユーザサービス定義については，マニュアル「OpenTP1 システム定義」を参照してください。

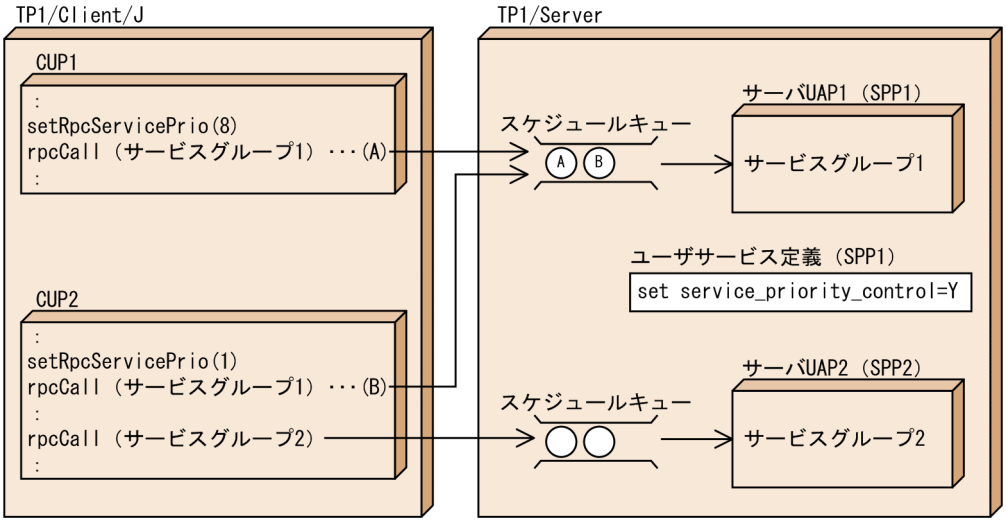
リモート API 機能を使用したサービス要求に対して，setRpcServicePrio メソッドを呼び出しても無効となります。

次の図にプライオリティを設定した場合の SPP のスケジュールを示します。

優先度の低い 8 を設定した(A)のサービス要求と，優先度の高い 1 を設定した(B)のサービス要求が同時にスケジュールキューに存在した場合，(B)のサービス要求を先に取り出します。

(A)のサービス要求と(B)のサービス要求の優先度が同じだった場合は，先入れ先出しでスケジュールキューから取り出します。

図 2-48　プライオリティを設定した場合の SPP のスケジュール



2.16.3　TP1/Server 側に必要な設定

この機能を使用する場合は，TP1/Server のユーザサービス定義を次の手順で変更してください。

(1)　ユーザサーバの停止

（例）dcsvstop ユーザサーバ名

(2)　ユーザサービス定義の変更

次の記述をユーザサービス定義に追加します。

set service_priority_control = Y

(3) ユーザサーバの開始

（例）dcsvstart -u ユーザサーバ名

3 プログラムインタフェース

TP1/Client/J のプログラムインタフェースについて説明します。

3.1　API 一覧

TP1/Client/J が使用する API の一覧を，次の表にそれぞれに示します。

表 3-1　API 一覧（パッケージ，クラス）

分類	名称	意味・機能
パッケージ	JP.co.Hitachi.soft.OpenTP1	TP1/Client/J のクラスのパッケージ名
クラス	TP1Client	TP1/Client/J のクラス名
	DCRpcBindTbl	DCRpcBindTbl クラス

表 3-2　API 一覧（メソッド）

分類	メソッド	意味・機能
実行系	acceptNotification(byte[] inf,int[] inf_len,int port,int timeout,byte[] hostname,byte[] nodeid)	サーバ側の関数（dc_rpc_cltsend 関数）で通知されるメッセージを受信する。
	acceptNotificationChained(byte[] inf,int[] inf_len,int timeout,byte[] hostname,byte[] nodeid)	
	cancelNotification(byte[] inf,int inf_len,String hostname,int port)	サーバからの一方通知受信待ち状態（acceptNotification メソッド，あるいは acceptNotificationChained メソッドの発行）を解除する。
	closeConnection()	CUP と，rap リスナーおよび rap サーバとの間に確立済みの常設コネクションを切断する。
	closeNotification()	一方通知連続受信機能を使用するための環境を削除する。
	cltAssemSend(byte[] buff, int sendleng, String hostname, int portnum, int timeout, int flags)	受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。
	cltAssemReceive(byte[] buff, int[] recvleng, int timeout, int flags)	受信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する。
	cltReceive(byte[] buff, int[] recvleng, int timeout, int flags)	MHP が送信したメッセージを受信する。
	cltSend(byte[] buff, int sendleng, String hostname, int portnum, int flags)	MHP へメッセージを送信する。
	getTrnID(byte[] gid, byte[] bid)	トランザクショングローバル識別子，トランザクションブランチ識別子を取得する。
	openConnection()	rap サーバとの間に常設コネクションを確立するか，または，TP1/Web に接続して擬似セションを開始する。

分類	メソッド	意味・機能
実行系	openConnection(String host, int port)	rap サーバとの間に常設コネクションを確立するか，または，TP1/Web に接続して擬似セションを開始する。
	openConnection(String url, short flags)	
	openNotification(int port)	一方通知連続受信機能を使用するための環境を作成する。
	rpcCall(String group, String service, byte[] in_data, int[] in_len, byte[] out_data, int[] out_len, int flags)	指定された「サービスグループ名＋サービス名」に対して RPC を行う。
	rpcCall(String group, String service, byte[] in_data, byte[] out_data, int flags)	
	rpcCallTo(DCRpcBindTbl direction, String group, String service, byte[] in_data, int[] in_len, byte[] out_data, int[] out_len, int flags)	指定された通信先ノードの「サービスグループ名＋サービス名」に対して RPC を行う。
	rpcClose()	TP1/Server の SPP を呼び出すための環境を解放する。
	rpcOpen()	TP1/Server の SPP を呼び出すための環境を初期化する。
	rpcOpen(String deffilename)	
	trnBegin()	グローバルトランザクションを開始する。
	trnChainedCommit()	トランザクションの同期点を取得する。
	trnChainedRollback()	トランザクションをロールバックする。
	trnInfo()	TP1Client オブジェクトがトランザクションとして稼働しているか確認する。
	trnUnchainedCommit()	トランザクションの同期点を取得する。
	trnUnchainedRollback()	トランザクションをロールバックする。
定義系	setConnectInformation(byte[] inf, short inf_len)	端末識別情報を設定する。
	setDataTraceMode(String TrcPath, int size, int Datasize, boolean flag)※	Java アプリケーションおよび Java サーブレットから RPC を行うとき，データトレースをファイルに出力するかどうかを指定する。
	setDccltdelay(int sec)	CUP と rap リスナーおよび rap サーバ間の通信遅延時間を設定する。
	setDccltextend(int flags)	TP1/Client/J の機能の拡張レベルを設定する。各指定値の論理和を設定すると，複数の機能の拡張レベルを指定できる。
	setDccltinquiretime(int sec)	rap サーバが監視する問い合わせ間隔最大時間（クライアントから一定時間サービス要求が来ない場合に接続を切断する時間）を設定する。

分類	メソッド	意味・機能
定義系	setDchost(String host, int port)	窓口となる TP1/Server のホスト名とポート番号を設定する。
	setDcselint(int msec)	このメソッドは意味を持ちません。 旧バージョンからのソースの互換性を保証するためだけに残しています。
	setDcwatchtim(int sec)	同期応答型 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最大応答待ち時間を設定する。
	setErrorTraceMode(String TrcPath, int size, boolean flag)※	Java アプリケーションおよび Java サーブレットから TP1Client クラスを呼び出すとき，サーバとの接続，切断，およびエラートレースをファイルに出力するかどうかを指定する。
	setMethodTraceMode(String TrcPath, int size, boolean flag)※	Java アプリケーションおよび Java サーブレットから TP1Client クラスを呼び出すとき，メソッドトレースをファイルに出力するかどうかを指定する。
	setRpcextend(int extendoption)	RPC 関連の設定を変更する。
	setScdDirectObject(String scdhost, int scdport, int flags)	通信先スケジューラのホスト名，ポート番号を設定する。
	setTraceArray(String[] array)	Java アプリケーション，Java サーブレットおよび Java アプレットから TP1Client クラスを呼び出すとき，サーバとの接続，切断，およびエラートレースを格納するための配列（メモリトレース）を指定する。障害情報を参照するには，この配列の内容を Java アプリケーション，Java サーブレットまたは Java アプレットで参照する。
	setUapTraceMode(String TrcPath, int size, boolean flag)※	Java アプリケーションおよび Java サーブレットから TP1Client クラスを呼び出すとき，UAP トレースをファイルに出力するかどうかを指定する。

注※

このメソッドを使用できるのは，Java アプリケーションまたは Java サーブレットだけです。Java アプレットで使用すると SecurityException が発生します。

3.2　API の使用方法

TP1/Client/J が提供するクラスライブラリを使用して Java アプレット，Java アプリケーション，または Java サーブレットを作成したあとに，主に五つの API を実行して操作します。

1. TP1/Client/J の使用準備
2. TP1Client クラスのインスタンス作成
3. RPC 環境の初期化
4. TP1/Server の rap サーバに接続（リモート API 機能を使用した RPC を行う場合）
5. 遠隔サービスの呼び出し（RPC）
6. rap リスナー，rap サーバとの常設コネクションの切断（リモート API 機能を使用して RPC を行っている場合）
7. RPC 環境の解放

実行順に各 API の概要と例を示します。クラスの詳細については，「4.　TP1/Client/J で使用するクラス」を参照してください。

注意事項

TP1Client クラスのメソッドはマルチスレッド環境で使用できますが，スレッドセーフではありません。TP1Client クラスのメソッドをマルチスレッド環境で使用する場合は，次のどちらかの方法で実装してください。

- 複数のスレッド間で同一の TP1Client クラスのインスタンスを共有しない場合
  各スレッドに専用の TP1Client クラスのインスタンスを作成してください。各スレッドで実行する TP1Client クラスのメソッドは，この専用のインスタンスを使用してください。
- 複数のスレッド間で同一の TP1Client クラスのインスタンスを共有する場合
  同一の TP1Client クラスのインスタンスのメソッドを同時に実行しないよう，各スレッド間で必ず排他を掛けてください。

複数のスレッドから，同一の TP1Client クラスのインスタンスのメソッドを同時に実行した場合，メソッドが予期しないエラーを返すことがあるため，TP1/Client/J の動作は保証できません。

3.2.1　API の実行順序

(1)　TP1/Client/J の使用準備

プログラムで TP1/Client/J パッケージのインポート宣言をします。

Java の各機能のパッケージをインポートするのと同様に，TP1/Client/J のパッケージもインポートしてください。

```
import java.applet.*;                 // java.appletパッケージのインポート
import java.awt.*;                     // java.awtパッケージのインポート
import JP.co.Hitachi.soft.OpenTP1.*;  // TP1Clientパッケージのインポート
```

(2) TP1Client クラスのインスタンス作成

TP1/Client/J のクラス名は TP1Client です。TP1/Server にアクセスする必要があるときに TP1Client クラスのインスタンスを作成してください。

```
TP1Client zaiko;                    // zaikoというインスタンス変数を宣言
zaiko = new TP1Client();            // TP1Clientクラスのインスタンス作成
                                    // （コンストラクタ呼び出し）
```

(3) RPC 環境の初期化

TP1/Server の SPP を呼び出すために RPC 環境を初期化します。RPC 環境を初期化するには，rpcOpen メソッドを呼び出します。

rpcOpen メソッドは，呼び出し時に何らかの問題が発生すると例外を返します。必要に応じて例外処理を行ってください。

なお，スケジューラダイレクト機能を使用した RPC，ネームサービスを利用した RPC，または通信先を指定した RPC を行う場合は，必ず rpcOpen メソッドを呼び出してください。リモート API 機能を使用した RPC を行う場合は，呼び出さなくてもかまいません。

TP1/Client/J 環境定義をシステムプロパティから取得する場合

```
try {
  zaiko.rpcOpen();                      // RPC環境の初期化
} catch(ErrFatalException e) {          // RPC環境初期化エラー発生
  System.out.println("CUP Initialize error");
} catch (TP1ClientException e) {        // その他のエラー発生
  System.out.println("error occured");
}
```

TP1/Client/J 環境定義をファイルから取得する場合

```
try {
  zaiko.rpcOpen("C:¥¥Clt4J¥¥conf¥¥clt4j.ini");
                                        // RPC環境の初期化
} catch(ErrFatalException e) {          // RPC環境初期化エラー発生
  System.out.println("CUP Initialize error");
} catch (TP1ClientException e) {        // その他のエラー発生
  System.out.println("error occured");
}
```

(4) TP1/Server の rap サーバに接続（リモート API 機能を使用した RPC を行う場合）

リモート API 機能を使用した RPC を行う場合，TP1/Server の rap リスナーおよび rap サーバに接続します。接続を解除するまで，Java アプレット，Java アプリケーション，または Java サーブレットと rap サーバとの間に専用の TCP/IP コネクションが開設されます。

接続するために，TP1Client クラスの openConnection メソッドを呼び出します。openConnection メソッドは，呼び出し時に何らかの問題が発生すると例外を返します。必要に応じて例外処理を行ってください。

なお，スケジューラダイレクト機能を使用した RPC，ネームサービスを利用した RPC，通信先を指定した RPC，またはオートコネクトモードを使用した RPC を行う場合は，openConnection メソッドを呼び出さないでください。

```
try {
  zaiko.openConnection(”zaikosv”,12000);
                          // rapリスナー，rapサーバとのコネクション確立
} catch(ErrTimedOutException e) {       // 接続タイムアウト発生
  System.out.println(”server inactive”);
} catch (TP1ClientException e) {        // その他のエラー発生
  System.out.println(”error occured”);
}
```

rap リスナー，rap サーバと常設コネクションを確立している途中に，rap リスナー，rap サーバ，または OpenTP1 システムが異常終了して常設コネクションが切断された場合でも，CUP 側が常設コネクションの切断を検知できないことがあります。その場合，次に発行するメソッド（rpcCall，closeConnection）が ErrTimedOutException を返すことがあります。

(5) 遠隔サービスの呼び出し（RPC）

rap サーバへの接続が成功したあと，遠隔サービス（SPP）を呼び出せます（RPC）。RPC を行うときは，rpcCall メソッドを呼び出します。rpcCall メソッドの引数には，サービスを示すサービスグループ名，サービス名，問い合わせデータ，問い合わせデータ長，応答データ受信領域，応答データ受信領域長，および RPC の実行形態を指定します。

rpcCall メソッドは，実行時に何らかの問題が発生すると例外を返します。必要に応じて例外処理を行ってください。

```
int in_len[] = new int[1];
int out_len[] = new int[1];
byte in_data[];
byte out_data[];
String in = ”PN=0012-2456-12”;       // 問い合わせデータ(便宜上固定文字列)
in_len[0] = 16;                         // 問い合わせデータ長(便宜上固定)
out_len[0] = 16;                        // 応答データ領域長(便宜上固定)
in_data = new byte[in_len[0]];          // 問い合わせデータ領域作成
in.getBytes(0, in.length(), in_data, 0);// 問い合わせデータ設定
```

```
out_data = new byte[out_len[0]];          // 応答データ領域作成
try {
  zaiko.rpcCall("group", "service",       // RPC実行(問い合わせ・応答型)
      in_data, in_len, out_data, out_len, zaiko.DCNOFLAGS);
} catch (ErrTimedOutException e) {        // RPC要求がタイムアウト
  System.out.println("RPC timedout");
} catch (TP1ClientException e) {          // その他のエラー発生
  System.out.println("error occured");
}
```

なお，RPC には次の表に示す 3 種類があります。flags 引数に指定して選択します。

RPC の種類		機能	指定方法
問い合わせ応答	連鎖なし	サーバに対してサービス要求をしたあと，応答が返る。次回同一サーバへ問い合わせたときは，前回と同一のサーバプロセスで実行するとは限らない。	DCNOFLAGS
	連鎖あり	サーバに対してサービス要求をしたあと，応答が返る。次回同一サーバへ問い合わせたときは，前回と同一のサーバプロセスで実行する（サーバプロセスを占有して使用する）。なお，スケジューラダイレクト機能を使用した RPC，およびネームサービスを使用した RPC では使用できない。	DCRPC_CHAINED
応答なし		サーバに対してサービス要求をしたあと，応答を受信できない。サービスを正常に実行できたかどうか，クライアントではわからない。応答を待たないため，クライアントの実行性能が良い。	DCRPC_NOREPLY

注

DCRPC_CHAINED でサービス呼び出しを実行した場合，最後のサービス呼び出し時に，連鎖型 RPC の終了を示すため，DCNOFLAGS を指定して RPC を実行してください。

(6) rap リスナーおよび rap サーバとの常設コネクションの切断（リモート API 機能を使用して RPC を行っている場合）

TP1/Server とのオンライン業務が完了したあとに，closeConnection メソッドを呼び出して rap リスナー，rap サーバとの常設コネクションを切断します。このメソッドに引数はありません。

closeConnection メソッドは，実行時に何らかの問題が発生すると例外を返します。必要に応じて例外処理を行ってください。

```
try {
  zaiko.closeConnection();
              // rapリスナー，rapサーバとの常設コネクション切断
} catch (TP1ClientException e) {          // エラー発生
  System.out.println("error occured");
}
```

(7) RPC 環境の解放

CUP の最後に，rpcClose メソッドを呼び出して CUP の RPC 環境を解放します。このメソッドに引数はありません。

rpcClose メソッドは，実行時に何らかの問題が発生すると例外を返します。必要に応じて例外処理を行ってください。

```
try {
  zaiko.rpcClose();                    // RPC環境の解放
} catch (TP1ClientException e) {       // エラー発生
  System.out.println("error occured");
}
```

3.2.2 TP1/Client/J 実行時の調整

通常，メソッドの監視時間は TP1/Client/J 環境定義で設定します。しかし，TP1/Client/J 環境定義の設定では，すべてのメソッドで同じ時間を使用するため，TP1/Server 側で時間の掛かる処理がある場合，その時間の最大値を指定する必要があります。そのため，処理に合わせた実行時間の調整ができません。

この問題を解決するため，TP1/Client/J では，動的にメソッドの監視時間を調整するメソッドを提供しています。なお，これらのメソッドを実行すると引数で指定された値が実行時の値として設定され，TP1Client クラスのインスタンスが存在する間（rpcClose メソッドが呼び出されるまで），またはメソッドが新たに指定し直されるまでの間，有効です。メソッドの詳細については「4. TP1/Client/J で使用するクラス」を参照してください。

(1) 時間監視の調整

時間監視の調整には，setDcwatchtim，setDccltinquiretime，および setDccltdelay の三つのメソッドを使用します。それぞれのメソッドで指定できる時間監視を示します。

setDcwatchtim メソッド

応答監視時間を設定します。openConnection，rpcCall，および closeConnection メソッドを呼び出すと，rap リスナーおよび rap サーバとの間で問い合わせ応答を行います。問い合わせを開始してから応答監視時間経過しても応答が返らない場合はタイムアウトになります。サービスを実行するための所要時間や通信速度を考慮して，適当な値を設定してください。

setDccltinquiretime メソッド

openConnection メソッドを呼び出して CUP との間で常設コネクションが確立された rap リスナーおよび rap サーバは，closeConnection メソッドが呼び出されるまでコネクションが確立されたままです。TP1/Client/J に何らかの障害が発生して closeConnection メソッドが呼び出せなくなると，rap リスナーおよび rap サーバはいつまでも解放されません。これを防止するために，TP1/Client/J から一定時間が経過しても問い合わせがないときに，rap リスナーおよび rap サーバ側で強制的に接続を解放するための時間を設定します。

なお，DCCM3 が接続先の rap サーバの場合，setDccltinquiretime メソッドの設定値は無効になります。

setDccltdelay メソッド

rpcCall メソッドを呼び出すと，TP1/Client/J 側では setDcwatchtim メソッドで指定した値で応答時間を監視しています。TP1/Client/J からの RPC 要求を中継する rap サーバでは（setDcwatchtim メソッドで指定した時間 − setDccltdelay メソッドで指定した時間）分だけ応答を監視します。
setDccltdelay メソッドで指定する時間は，TP1/Client/J，TP1/Server 間の通信遅延時間です。TP1/Client/J 側と rap サーバ側が同一時間で応答監視を行うと，TP1/Client/J 側の方が応答監視開始が早い分，先にタイムアウトします。このため rap サーバ側で応答監視満了ぎりぎりの時間に応答を返しても，TP1/Client/J 側がすでにタイムアウトしていて応答を受け取れないことがあります。このような現象を避けるため，このメソッドで rap サーバ側を先にタイムアウトさせます。なお，このメソッドでの指定値を有効にするには，TP1/Client/J 環境定義で dcwatchtiminherit=Y を定義するか，または setRpcextend メソッドで，DRPC_WATCHTIMINHERIT 引数を指定してください。

3.2.3　障害情報の採取および機能の調整

障害情報の採取および機能の調整のために，setTraceArray，setDccltextend，および setRpcextend の三つのメソッドがあります。各メソッドの指定値は，メソッドを呼び出した時点から有効になります。なお，これらのメソッドを呼び出すと引数で指定した値は，実行時の値として設定され，TP1Client クラスのインスタンスが存在する間（rpcClose メソッドが呼び出されるまで），または新たに同一のメソッドが呼び出されるまでの間，有効です。

setTraceArray メソッド

Java アプレットでは，Java のセキュリティ上の制約から実行環境にファイルを作成できません。このため，障害時の情報などをトレース情報としてファイルに取得できません。このような場合，トレース情報を取得するためにメモリトレース機能を使用します。setTraceArray メソッドで String 型の配列を指定しておくと，TP1Client クラスはこの配列に障害情報を格納します。Java アプレットまたは Java アプリケーションでエラー発生時にこの String 配列の内容をブラウザに表示するなどして参照し，障害時の原因調査，対策に利用できます。メモリトレースの詳細については，「2.11.4　エラートレース，メモリトレース」を参照してください。

setDccltextend メソッド

実行環境によって差が出る機能を使用するかどうかを指定します。TP1/Server の SPP 側の dc_rpc_get_callers_address 関数で TP1/Client/J の IP アドレスを参照できるかどうかを指定できます。Microsoft Internet Explorer 3.0J では，TP1/Client/J 側で自 IP を調べられません。また，Windows 95 のマルチホームドホスト形態でこの操作を行うと OS 内部で異常が発生するため，デフォルトでは自 IP を調べる操作は行いません。

setRpcextend メソッド

TP1/Client/J から発行する RPC の機能拡張オプションを指定します。RPC 機能の拡張レベルを複数指定する場合，それぞれの指定値の論理和を指定します。

3.2.4　トレース出力の指示

各種トレースを取得するには，TP1/Client/J 環境定義で設定するか，または実行時にメソッドで指定します。

ただし，Java アプレットでは，セキュリティの制約からトレースを取得できません。また，Java サーブレットは，使用するアプリケーションサーバによってセキュリティの制約に違いがあります。詳細は使用するアプリケーションサーバを確認してください。実行時にトレースの取得の有無を変更できるのは，setUapTraceMode，setDataTraceMode，setErrorTraceMode，および setMethodTraceMode の四つのメソッドです。

各トレースファイルが出力する内容については，「2.11.1　トレースファイルの出力内容」を参照してください。これらのトレースは次のメソッドを実行した時点から出力されます。

setUapTraceMode メソッド
　TP1Client クラスの各メソッドの呼び出し情報をファイルに出力することを指定します。

setDataTraceMode メソッド
　TP1/Server との通信データをファイルに出力することを指定します。

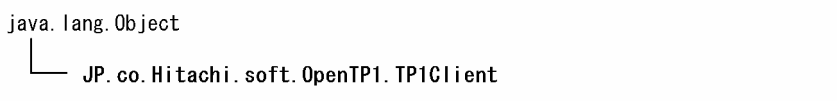
setErrorTraceMode メソッド
　エラー発生時の情報をファイルに出力することを指定します。

setMethodTraceMode メソッド
　TP1Client クラスの内部メソッドの走行履歴をファイルに出力することを指定します。Java 環境では障害情報が少ないため，どの処理で障害が発生したかを調査するために採取します。

4 TP1/Client/J で使用するクラス

TP1/Client/J で使用するクラスについて説明します。

クラス TP1Client




```
public final class TP1Client
extends java.lang.Object
```

TP1/Server のサーバに対してサービス要求を行うクラスです。

関連項目

TP1ClientException

フィールド

●DCNOFLAGS

```
public static final int DCNOFLAGS = 0x00000000
```

RPC の形態に同期応答型 RPC を指定します。

●DCRPC_NOREPLY

```
public static final int DCRPC_NOREPLY = 0x00000001
```

RPC の形態に非応答型 RPC を指定します。

●DCRPC_CHAINED

```
public static final int DCRPC_CHAINED = 0x00000004
```

RPC の形態に連鎖型 RPC を指定します。

●DCRPC_SCD_LOAD_PRIORITY

```
public static final int DCRPC_SCD_LOAD_PRIORITY = 0x00000008
```

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。

●DCRPC_WATCHTIMINHERIT

```
public static final int DCRPC_WATCHTIMINHERIT = 0x00000010
```

リモート API 機能を使用した RPC を行う場合に，CUP の最大応答待ち時間を rap サーバ側に引き継ぐかどうかを指定します。

●DCRPC_RAP_AUTOCONNECT

```
public static final int DCRPC_RAP_AUTOCONNECT = 0x00000020
```

リモート API 機能を利用した RPC を行う場合に，TP1/Client/J が自動的にコネクションを確立するかどうかを指定します。

●DCRPC_TPNOTRAN

```
public static final int DCRPC_TPNOTRAN = 0x00000020
```

RPC の形態にトランザクションを引き継がない RPC を指定します。

●DCRPC_WATCHTIMRPCINHERIT

```
public static final int DCRPC_WATCHTIMRPCINHERIT = 0x00000040
```

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。

●DCRPC_MAX_MESSAGE_SIZE

```
public static final int DCRPC_MAX_MESSAGE_SIZE = 1048576
```

RPC 電文の最大長を指定します。指定できる最大値は 1048576 バイトです。ただし，
dccltrpcmaxmsgsize オペランドを使用した場合，RPC 電文の最大長は，
DCRPC_MAX_MESSAGE_SIZE の値ではなく，dccltrpcmaxmsgsize オペランドに指定した値になります。

コンストラクタ

●TP1Client

```
public TP1Client()
```

TP1Client クラスのインスタンスを作成します。CUP を Java アプリケーション，または Java サーブレットで作成する場合に使用します。

●TP1Client

```
public TP1Client(Applet app)
```

TP1Client クラスのインスタンスを作成します。CUP を Java アプレットとして作成する場合に使用します。

パラメタ

app
Java アプレットのインスタンスオブジェクトを指定します。

メソッド

●setDccltinquiretime

```
public void setDccltinquiretime(int sec)
                        throws ErrInvalidArgsException
```

CUP がサーバに対して問い合わせを行ってから，次の問い合わせをするまでの間隔の最大時間を設定します。この値は，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，rap サーバは強制的に常設コネクションを切断します。このメソッドは TP1Client クラスのインスタンスが存在している間，呼び出せます。

パラメタ

sec

rap サーバが監視する問い合わせ間隔最大時間を 0 から 1048575（単位：秒）までの範囲で指定します。0 を指定した場合は，rap サーバは CUP からの問い合わせを無限に待ちます。

戻り値

なし。

例外

ErrInvalidArgsException

sec 引数に指定された値が，0 から 1048575 までの範囲にありません。

注意事項

常設コネクションの確立前に「問い合わせ間隔最大時間」を変更できます。

常設コネクションの確立後に「問い合わせ間隔最大時間」を変更しても，rap サーバ側の監視タイマには反映されません。この場合，確立中の常設コネクションを切断し，次回常設コネクションを確立するタイミングで rap サーバ側の監視タイマに反映されます。

●setDccltdelay

```
public void setDccltdelay(int sec)
                  throws ErrInvalidArgsException
```

rap サーバと，CUP 間の通信遅延時間を設定します。このメソッドを設定すると，rap サーバ側の最大応答待ち時間の時間監視を指定した値の分だけ早く終了させ，CUP 側の最大応答待ち時間の監視時間タイムアウトによる電文のすれ違いを防止できます。

setDcwatchtim メソッドで指定した最大応答待ち時間が 0 の場合，このメソッドで指定した通信遅延時間は無視されます。また，最大応答待ち時間から rap サーバ通信時間を引いた値が 0 または負の値になるときは，rap サーバ側での最大応答待ち時間を 1 と仮定して rap サーバが動作します。

なお，このメソッドの指定を有効にするには，TP1/Client/J 環境定義で dcwatchtiminherit=Y を指定するか，または setRpcextend メソッドの DRPC_WATCHTIMINHERIT 引数を指定してください。

パラメタ

sec

rap サーバと，CUP 間の通信遅延時間を 0 から 65535（単位：秒）までの範囲で指定します。

戻り値

なし。

例外

ErrInvalidArgsException

sec 引数に指定された値が，0 から 65535 までの範囲にありません。

●setDcwatchtim

```
public void setDcwatchtim(int sec)
                  throws ErrInvalidArgsException
```

同期応答型 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最大応答待ち時間を設定します。このメソッドで設定する値は，TP1/Client/J の内部通信での最大応答待ち時間としても使用されます。

パラメタ

sec

最大応答待ち時間を 0 から 65535（単位：秒）までの範囲で指定します。0 を指定した場合は，応答を受信するまで無限に待ちます。使用する Java 環境によっては，サーバがダウンしても TCP/IP コネクションの切断を検出できない場合があります。この場合，待ち時間を無限にすると存在しないサーバからの応答を永久に待ち続けることになるため，適当な待ち時間を設定してください。

戻り値

なし。

例外

ErrInvalidArgsException

sec 引数に指定された値が，0 から 65535 までの範囲にありません。

●setDcselint

```
public void setDcselint(int msec)
                throws ErrInvalidArgsException
```

推奨されていません。旧バージョンからのソースの互換性を保証するためだけに残しているメソッドです。現在は，このメソッドは意味を持ちません。

●setDccltextend

```
public void setDccltextend(int flags)
```

TP1/Client/J の機能の拡張レベルを指定します。機能の拡張レベルを複数指定する場合，それぞれの指定値の論理和を指定します。

パラメタ

flags

TP1/Client/J の機能の拡張レベルを指定します。

0x00000000：TP1/Client/J の機能を拡張しません。

0x00000001：TP1/Client/J の機能を拡張します。rpcCall メソッド呼び出し時，自 CUP の IP アドレスをサービスに連絡します。呼び出したサービスで dc_rpc_get_callers_address()関数を実行し，CUP のアドレスを求める必要がある場合，指定してください。

戻り値

なし。

●setRpcextend

```
public void setRpcextend(int extendoption)
                 throws ErrInvalidArgsException
```

TP1/Client/J から発行する RPC の機能拡張オプションを指定します。RPC 機能の拡張レベルを複数指定する場合，それぞれの指定値の論理和を指定します。

パラメタ

extendoption

RPC 機能の拡張レベルを指定します。

DCRPC_SCD_LOAD_PRIORITY：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。このオプションが真の場合，TP1/Client/J 環境定義に dcscdloadpriority=Y を指定したときと同じ動作をします。このオプションが偽の場合，TP1/Client/J 環境定義に dcscdloadpriority=N を指定したときと同じ動作をします。

DCRPC_WATCHTIMINHERIT：リモート API 機能を使用した RPC を行う場合に，CUP の最大応答待ち時間を rap サーバ側に引き継ぐかどうかを指定します。このオプションが真の場合，TP1/Client/J 環境定義に dcwatchtiminherit=Y を指定したときと同じ動作をします。このオプションが偽の場合，TP1/Client/J 環境定義に dcwatchtiminherit=N を指定したときと同じ動作をします。

DCRPC_RAP_AUTOCONNECT：リモート API 機能を使用した RPC を行う場合に，TP1/Client/J が自動的にコネクションを確立するかどうかを指定します。このオプションが真の場合，TP1/Client/J 環境定義に dcrapautoconnect=Y を指定したときと同じ動作をします。このオプションが偽の場合，TP1/Client/J 環境定義に dcrapautoconnect=N を指定したときと同じ動作をします。また，すでに非オートコネクトモードを使用し，openConnection メソッドを呼び出してコネクションを確立している場合は，このオプションは無視されます。

DCRPC_WATCHTIMRPCINHERIT：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。このオプションが真の場合，TP1/Client/J 環境定義に dcwatchtimrpcinherit=Y を指定したときと同じ動作をします。このオプションが偽の場合，TP1/Client/J 環境定義に dcwatchtimrpcinherit=N を指定したときと同じ動作をします。

戻り値

なし。

例外

ErrInvalidArgsException

extendoption 引数の指定に誤りがあります。

●setRpcServicePrio

```
public void setRpcServicePrio(int prio)
                  throws ErrInvalidArgsException
```

サービス要求のプライオリティを設定します。サービス要求単位でスケジュールプライオリティを制御する場合に呼び出します。このメソッドで設定したプライオリティは，このメソッドを再び呼び出すまで更新されません。したがって，同じプライオリティでまとめてサービス要求する場合は，このメソッドを 1 回だけ呼び出します。

このメソッドで指定したプライオリティは，直後に呼び出す rpcCall メソッドで，スケジュールキューを経由してサーバに通知されます。

このメソッドを一度も呼び出さない場合の処理は，省略時解釈値である 4 が，サービス要求のプライオリティとして指定されます。

rpcClose メソッドを呼び出した場合は，サービス要求のプライオリティを省略時解釈値である 4 にリセットします。

パラメタ

prio

サービス要求のスケジュールプライオリティを，0 または 1 から 8 の範囲で設定します。prio の設定は省略できません。

最も高いプライオリティの値は 1 で，最も低いプライオリティの値は 8 です。0 を設定した場合は，省略時解釈値である 4 が，サービス要求のプライオリティとして指定されます。上記以外の値を設定した場合，setRpcServicePrio メソッドは例外を返します。

戻り値

なし。

例外

ErrInvalidArgsException

メソッドの引数の指定に誤りがあります。

注意事項

- 設定したサービス要求のプライオリティは，サーバ UAP がキュー受信型サーバであり，かつサーバ UAP のユーザサービス定義に service_priority_control=Y（プライオリティを制御する）を指

定している場合だけ有効です。サービス要求する相手のサーバ UAP でプライオリティを制御していない場合は，このメソッドを呼び出しても無効になります。

- リモート API 機能を使用した RPC では，このメソッドでサービス要求のプライオリティを設定しても無効になります。
- rpcCall メソッドは，サービス要求のプライオリティを省略値にリセットしません。サービス要求のプライオリティをリセットする場合は，引数 prio に 0 を設定した setRpcServicePrio メソッドを呼び出し直してください。

●setDchost

```
public void setDchost(String host, int port)
               throws ErrHostUndefException,
                      ErrInvalidPortException,
                      ErrProtoException
```

窓口となる TP1/Server のホスト名とポート番号を設定します。

パラメタ

host

窓口となる TP1/Server のホスト名を指定します。リモート API 機能を使用した RPC を行う場合は，通信先の rap リスナーのホスト名を指定します。CUP と TP1/Server の間にファイアウォールがある場合は，ファイアウォールのホスト名を指定します。

port

窓口となる TP1/Server 上で動作しているスケジュールサーバのポート番号，ネームサーバのポート番号を指定します。リモート API 機能を使用した RPC を行う場合は，通信先の rap リスナーのポート番号を指定します。ポート番号は，5001 から 65535 までの範囲で指定します。

戻り値

なし。

例外

ErrHostUndefException

host 引数の指定に誤りがあります。

ErrInvalidPortException

port 引数の指定に誤りがあります。

ErrProtoException

メソッドの発行順序に誤りがあります。リモート API 機能を使用した RPC を行う場合で，すでに rap サーバとの間に常設コネクションが確立されている状態でこのメソッドが呼び出されました。

●setTraceArray

```
public void setTraceArray(String[] array)
```

引数に指定された配列にエラートレースを取得するかどうかを指定します。CUP が Java アプレット，Java アプリケーション，および Java サーブレットのどの場合でも，このメソッドを呼び出せます。

パラメタ

array

エラートレースを格納する String 配列を指定します。null を指定した場合，エラートレースを取得しません。

戻り値

なし。

●setUapTraceMode

```
public void setUapTraceMode(String TrcPath,
                            int size,
                            boolean flag)
                     throws ErrInvalidArgsException
```

CUP が Java アプリケーションまたは Java サーブレットの場合，UAP トレースを取得するかどうかを指定します。CUP が Java アプレットの場合，このメソッドを呼び出さないでください。

パラメタ

TrcPath

UAP トレースを出力するディレクトリを指定します。flag 引数に false を指定した場合は無視されます。

size

出力する UAP トレースファイルのサイズを 4096 から 1048576 までの範囲で指定します。flag 引数に false を指定した場合は無視されます。

flag

UAP トレースを取得するかどうかを指定します。

true：UAP トレースを取得します。

false：UAP トレースを取得しません。

戻り値

なし。

例外

ErrInvalidArgsException

引数の指定に誤りがあります。

注意事項

このメソッドは Java アプレットでは使用できません。Java アプレットで使用した場合の動作は保証できません。

●setMethodTraceMode

```
public void setMethodTraceMode(String TrcPath,
                               int size,
                               boolean flag)
                        throws ErrInvalidArgsException
```

CUP が Java アプリケーションまたは Java サーブレットの場合，メソッドトレースを取得するかどうかを指定します。CUP が Java アプレットの場合，このメソッドを呼び出さないでください。

パラメタ

TrcPath

メソッドトレースを出力するディレクトリを指定します。flag 引数に false を指定した場合は無視されます。

size

出力するメソッドトレースファイルのサイズを 4096 から 1048576 までの範囲で指定します。flag 引数に false を指定した場合は無視されます。

flag

メソッドトレースを取得するかどうかを指定します。

true：メソッドトレースを取得します。

false：メソッドトレースを取得しません。

戻り値

なし。

例外

ErrInvalidArgsException

引数の指定に誤りがあります。

注意事項

このメソッドは Java アプレットでは使用できません。Java アプレットで使用した場合の動作は保証できません。

●setDataTraceMode

```
public void setDataTraceMode(String TrcPath,
                             int size,
                             int DataSize,
                             boolean flag)
                      throws ErrInvalidArgsException
```

CUP が Java アプリケーションまたは Java サーブレットの場合，データトレースを取得するかどうかを指定します。CUP が Java アプレットの場合，このメソッドを呼び出さないでください。

パラメタ

TrcPath

データトレースを出力するディレクトリを指定します。flag 引数に false を指定した場合は無視されます。

size

出力するデータトレースファイルのサイズを 4096 から 1048576 までの範囲で指定します。flag 引数に false を指定した場合は無視されます。

DataSize

出力するデータサイズ 16 から 1048576 までの範囲で指定します。flag 引数に false を指定した場合は無視されます。

flag

データトレースを取得するかどうかを指定します。

true：データトレースを取得します。

false：データトレースを取得しません。

戻り値

なし。

例外

ErrInvalidArgsException

引数の指定に誤りがあります。

注意事項

このメソッドは Java アプレットでは使用できません。Java アプレットで使用した場合の動作は保証できません。

●setErrorTraceMode

```
public void setErrorTraceMode(String TrcPath,
                              int size,
                              boolean flag)
                       throws ErrInvalidArgsException
```

CUP が Java アプリケーションまたは Java サーブレットの場合，エラートレースを取得するかどうかを指定します。CUP が Java アプレットの場合，このメソッドを呼び出さないでください。

パラメタ

TrcPath

エラートレースを出力するディレクトリを指定します。flag 引数に false を指定した場合は無視されます。

size

出力するエラートレースファイルのサイズを 4096 から 1048576 までの範囲で指定します。flag 引数に false を指定した場合は無視されます。

flag

エラートレースを取得するかどうかを指定します。

true：エラートレースを取得します。

false：エラートレースを取得しません。

戻り値

なし。

例外

ErrInvalidArgsException

引数の指定に誤りがあります。

注意事項

このメソッドは Java アプレットでは使用できません。Java アプレットで使用した場合の動作は保証できません。

●rpcOpen

```
public void rpcOpen()
             throws ErrIOErrException,
                    ErrProtoException,
                    ErrFatalException,
                    ErrSecurityException,
                    ErrSyserrException
```

TP1/Client/J の各機能を使用する環境を初期化します。CUP を実行するとき，最初にこのメソッドを呼び出します。

パラメタ

なし。

戻り値

なし。

例外

ErrIOErrException

何らかの入出力例外が発生しました。

ErrProtoException

メソッドの発行順序に誤りがあります。rpcClose メソッドが呼び出されないで再度 rpcOpen メソッドが呼び出されました。

ErrFatalException

TP1/Client/J 環境定義の指定に誤りがあります。または，TP1/Server との通信環境の初期化に失敗しました。

ErrSyserrException

システムエラーが発生しました。

ErrSecurityException

セキュリティ例外が発生しました。

注意事項

- このメソッドを Java アプリケーション，または Java サーブレットで使用する場合は，TP1/Client/J 環境定義をシステムプロパティから取得します。
- このメソッドを Java アプレットで使用する場合は，TP1/Client/J 環境定義をアプレットの param タグから取得します。
- このメソッドを Java アプレットで使用する場合は，TP1Client クラスをインスタンス化するコンストラクタに，TP1Client(Applet app)を使用してください。

●rpcOpen

```
public void rpcOpen(String deffilename)
             throws ErrIOErrException,
                    ErrProtoException,
                    ErrFatalException,
                    ErrSyserrException,
                    ErrInvalidArgsException,
                    ErrSecurityException
```

TP1/Client/J の各機能を使用する環境を初期化します。CUP を実行するとき，最初にこのメソッドを呼び出します。

パラメタ

deffilename

TP1/Client/J 環境定義を格納したファイルのパス名を完全パス名または相対パス名で指定します。

戻り値

なし。

例外

ErrIOErrException

何らかの入出力例外が発生しました。

ErrProtoException

メソッドの発行順序に誤りがあります。rpcClose メソッドを呼び出さないで再度 rpcOpen メソッドが呼び出されました。

ErrFatalException

TP1/Client/J 環境定義に誤りがあります。または，TP1/Server との通信環境の初期化に失敗しました。

ErrSyserrException

システムエラーが発生しました。

ErrInvalidArgsException

引数の指定に誤りがあります。

ErrSecurityException

セキュリティ例外が発生しました。

注意事項

このメソッドは，Java アプレットでは使用できません。Java アプレットで使用する場合は，このメソッドの前に説明した rpcOpen メソッドを使用してください。

●rpcClose

```
public void rpcClose()
              throws ErrIOErrException,
                     ErrSyserrException,
                     ErrNetDownException
```

TP1/Client/J の各機能を使用する環境を解放します。TP1/Client/J の各機能を使用する場合は，rpcOpen メソッドを呼び出します。CUP の実行の最後にこのメソッドを呼び出します。

パラメタ

なし。

戻り値

なし。

例外

ErrIOErrException

何らかの入出力例外が発生しました。

ErrSyserrException

システムエラーが発生しました。

ErrNetDownException

ネットワーク障害が発生しました。

●openConnection

```
public void openConnection()
                    throws ErrIOErrException,
                           ErrHostUndefException,
                           ErrTimedOutException,
                           ErrNetDownException,
```

```
                    ErrNoBufsException,
                    ErrNotUpException,
                    ErrSyserrException,
                    ErrProtoException,
                    ErrInvalidPortException
```

TP1/Client/J 環境定義の dchost オペランドおよび dcrapport オペランドで指定された rap サーバとの間に常設コネクションを確立するか，または dcweburl オペランドで指定された OpenTP1 の WWW セション管理機能を利用したサーバ（TP1/Web）に接続して擬似セションを開始します。

なお，TP1Client クラスの 1 つのインスタンスに対して，同時に複数の常設コネクションは確立できません。

ただし，CUP と接続要求先 rap リスナーの間にファイアウォールがある場合，接続要求先 rap リスナーにはファイアウォールのホスト名とポート番号を指定してください。

パラメタ

なし。

戻り値

なし。例外を返さない場合，正常終了したか，またはすでに常設コネクションが確立されています。

例外

ErrIOErrException

何らかの入出力例外が発生しました。

ErrHostUndefException

rap リスナーのホスト名が TP1/Client/J 環境定義の dchost オペランドに指定されていません。

または，TP1/Client/J 環境定義の dcweburl オペランドに指定された URL（プロトコル，WWW サーバ，プロンプターの CGI 名称，TP1/Web のサービス名などの情報）に誤りがあります。

ErrTimedOutException

rap リスナーとのコネクション確立中にタイムアウトが発生しました。

または，TP1/Web の擬似セション開始中にタイムアウトが発生しました。

ErrNetDownException

rap リスナー，または TP1/Web との通信でネットワーク障害が発生しました。

または，通信先の TP1/Server が稼働していません。

ErrNoBufsException

rap リスナー，rap サーバでメモリ不足が発生しました。

または，TP1/Web でメモリ不足が発生しました。

ErrNotUpException

rap リスナー，rap サーバが稼働していません。

または，TP1/Web が稼働していません。

ErrSyserrException

システムエラーが発生しました。

ErrProtoException

メソッドの発行順序に誤りがあります。すでにコネクションを確立している間に再度 openConnection メソッドが呼び出されました。

ErrInvalidPortException

TP1/Client/J 環境定義の dcrapport オペランドに指定されたポート番号が不正，または未指定です。

注意事項

接続先の rap リスナーが起動していない場合，ErrIOErrException または ErrNetDownException が返されます。

このメソッドが例外を返した場合，常設コネクションは確立されていません。

●openConnection

```
public void openConnection(String host,
                           int port)
                    throws ErrIOErrException,
                           ErrHostUndefException,
                           ErrTimedOutException,
                           ErrNoBufsException,
                           ErrNotUpException,
                           ErrSyserrException,
                           ErrInvalidPortException,
                           ErrProtoException,
                           ErrInvalidArgsException,
                           ErrNetDownException
```

リモート API 機能を使用した RPC を行うために，CUP と rap リスナー，rap サーバとの間に常設コネクションを確立します。常設コネクションの確立先はパラメタで指定された値を使用します。

パラメタ

host

rap リスナー，またはファイアウォールのホスト名を指定します。

port

rap リスナー，またはファイアウォールのポート番号を 5001 から 65535 までの範囲で指定します。

戻り値

なし。例外を返さない場合，正常終了したか，またはすでに常設コネクションが確立されています。

例外

ErrIOErrException

何らかの入出力例外が発生しました。

ErrHostUndefException

host 引数の指定に誤りがあります。

ErrTimedOutException

rap リスナーとのコネクションの確立中にタイムアウトが発生しました。

ErrNoBufsException

rap リスナー，rap サーバでメモリ不足が発生しました。

ErrNotUpException

rap リスナー，rap サーバが稼働していません。

ErrSyserrException

システムエラーが発生しました。

ErrInvalidPortException

port 引数の指定に誤りがあります。

ErrProtoException

メソッドの発行順序に誤りがあります。すでにコネクションを確立している間に再度 openConnection メソッドが呼び出されました。

ErrInvalidArgsException

引数の指定に誤りがあります。

ErrNetDownException

rap リスナーとの通信でネットワーク障害が発生しました。または，通信先の TP1/Server が稼働していません。

注意事項

接続先の rap リスナーが起動していない場合，ErrIOErrException または ErrNetDownException が返されます。このメソッドが例外を返した場合，常設コネクションは確立されていません。

●openConnection

```
public void openConnection(String   url,
                           short   flags)
                    throws ErrIOErrException,
                           ErrHostUndefException,
                           ErrTimedOutException,
                           ErrNetDownException,
                           ErrNoBufsException,
                           ErrNotUpException,
                           ErrSyserrException,
                           ErrProtoException,
                           ErrInvalidArgsException
```

OpenTP1 の WWW セション管理機能を利用したサーバ（TP1/Web）に接続し，擬似セションを開始します。以降，HTTP プロトコルに基づいた通信サービスを行えます。

パラメタ

url

プロトコル，WWW サーバ，プロンプターの CGI 名称，TP1/Web のサービス名などの情報を URL 形式で指定します。

flags

DCSESSION を指定します。

戻り値

なし。

例外

ErrIOErrException

何らかの入出力例外が発生しました。

ErrHostUndefException

url 引数で指定された URL（プロトコル，WWW サーバ，プロンプターの CGI 名称，TP1/Web のサービス名などの情報）に誤りがあります。

ErrTimedOutException

TP1/Web の擬似セション開始中にタイムアウトが発生しました。

ErrNetDownException

ネットワーク障害が発生しました。

ErrNoBufsException

TP1/Web でメモリ不足が発生しました。

ErrNotUpException

TP1/Web が稼働していません。

ErrSyserrException

システムエラーが発生しました。

ErrProtoException

メソッドの発行順序に誤りがあります。

すでに rap サーバとの間に常設コネクションが確立されている状態で，TP1/Client/J 環境定義で dcrapautoconnect=Y または dcrapdirect=N を指定しました。

ErrInvalidArgsException

引数の指定に誤りがあります。

●closeConnection

```
public void closeConnection()
                    throws ErrIOErrException,
                           ErrSyserrException,
                           ErrProtoException,
```

```
                                ErrTimedOutException,
                                ErrNetDownException
```

CUP と rap リスナー，rap サーバとの間で確立されている常設コネクションを切断します。

パラメタ

なし。

戻り値

なし。

例外

ErrIOErrException

何らかの入出力例外が発生しました。

ErrSyserrException

システムエラーが発生しました。

ErrProtoException

メソッドの発行順序に誤りがあります。すでにコネクションが切断されている間に再度 closeConnection メソッドが呼び出されました。

ErrTimedOutException

TP1/Web との擬似セション切断中または rap リスナー，rap サーバとのコネクション切断中に，タイムアウトが発生しました。

ErrNetDownException

rap リスナーとの通信でネットワーク障害が発生しました。

注意事項

このメソッドが例外を返した場合でも，常設コネクションは切断されています。

●rpcCall

```
public void rpcCall(String group,
                    String service,
                    byte[] in_data,
                    int[] in_len,
                    byte[] out_data,
                    int[] out_len,
                    int flags)
             throws ErrInvalidArgsException,
                    ErrProtoException,
                    ErrNoBufsException,
                    ErrNetDownException,
                    ErrTimedOutException,
                    ErrMessageTooBigException,
                    ErrReplyTooBigException,
                    ErrNoSuchServiceGroupException,
                    ErrNoSuchServiceException,
                    ErrServiceClosedException,
                    ErrServiceTerminatingException,
```

```
                    ErrServiceNotUpException,
                    ErrNotUpException,
                    ErrSyserrAtServerException,
                    ErrSyserrException,
                    ErrNoBufsAtServerException,
                    ErrInvalidReplyException,
                    ErrInitializingException,
                    ErrTrnchkException,
                    ErrServerBusyException,
                    ErrSecchkException,
                    ErrServiceTerminatedException,
                    ErrIOErrException,
                    ErrTestmodeException,
                    ErrConnfreeException,
                    ErrHostUndefException,
                    ErrInvalidPortException
```

SPP のサービスを要求します。サービスグループ名とサービス名の組み合わせで識別されるサービス関数を呼び出し，その応答を受け取ります。このメソッドを呼び出したときに，目的のサービスグループが閉塞されていると ErrServiceClosedException が返されます。

flags 引数に DCRPC_TPNOTRAN を指定すると，トランザクションの処理からの RPC をトランザクションの処理ではないサービス要求にできます。

このメソッドを呼び出したときに，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，ErrServiceTerminatingException，ErrServiceClosedException，または ErrNoSuchServiceGroupException のどれかが返されます。どの例外が返されるかは，このメソッドを呼び出したタイミングによって異なります。

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 および max_socket_msglen オペランドの指定で電文の輻輳制御を行っています。そのためサーバがサービス要求を受信できない場合があります。この場合，メソッドを呼び出すと ErrServerBusyException が返されます。この例外が返されたあと，適当な時間を置いてから再度このメソッドを呼び出すと，サービス要求できることがあります。

パラメタ

group

サービスグループ名を 31 文字以内の識別子で指定します。

service

サービス名を 31 文字以内の識別子で指定します。

in_data

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 1 から DCRPC_MAX_MESSAGE_SIZE※の指定値までの範囲で指定します。入力パラメタ長は in_len[0]に格納してください。

注※

dccltrpcmaxmsgsize オペランドを使用した場合，DCRPC_MAX_MESSAGE_SIZE の値ではなく，dccltrpcmaxmsgsize オペランドに指定した値になります。

out_data

サービス関数で指定した応答が返される領域を指定します。

非応答型 RPC の場合は null を指定します。非応答型 RPC の場合，null 以外の値を指定しても何も格納されません。

out_len

サービスの応答の長さを 1 から DCRPC_MAX_MESSAGE_SIZE※の指定値までの範囲で指定します。応答の長さは out_len[0]に格納してください。

サービス要求終了時，SPP のサービス関数で指定した応答の長さが out_len[0]に設定されます。

非応答型 RPC の場合，応答の長さを指定しても無視され，サービス要求終了時，out_len[0]には何も設定されません。

注※

dccltrpcmaxmsgsize オペランドを使用した場合，DCRPC_MAX_MESSAGE_SIZE の値ではなく，dccltrpcmaxmsgsize オペランド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：連鎖型 RPC

DCRPC_TPNOTRAN：トランザクションを引き継がない RPC

DCNOFLAGS または DCRPC_CHAINED を指定すると，サーバから応答が返されるか，TP1/Client/J 環境定義の dcwatchtim オペランドで指定した最大応答待ち時間が経過してタイムアウトになるまで，このメソッドは制御を戻しません。ただし，サービス要求先の SPP が異常終了した場合は，すぐに例外を返します。この場合，TP1/Client/J 環境定義の dcwatchtim オペランドで指定した値によって，次の例外を返します。

- 1～65535 の値を指定した場合
  ErrTimedOutException
- 0 を指定した場合
  ErrServiceNotUpException

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，このメソッドはサービスの実行終了を待たないですぐに制御を戻します。このフラグを指定した場合，応答（out_data 引数の指定値）と応答の長さ（out_len 引数の指定値）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 側からはわかりません。

DCRPC_CHAINED を指定すると，同じサービスグループに属するサービスを複数回要求する場合は，最初の要求時と同一のプロセスで実行されます。

連鎖型 RPC を使用する場合，次に示す制限があります。

- 2 回目以降のこのメソッドの呼び出しは，ユーザサーバおよびサービスの閉塞を検出できません。
- 2 回目以降のこのメソッドの呼び出しで，サービス関数の処理に異常が発生した場合，サービス単位ではなくユーザサーバ全体が閉塞します。
- リモート API 機能を使用する場合に限り，flags 引数に DCRPC_CHAINED を指定できます。リモート API 機能を使用しない場合，flags 引数に DCRPC_CHAINED を指定すると ErrInvalidArgsException が返されます。
- 連鎖型 RPC は，次のどちらかの方法で終了させてください。

  ・連鎖型 RPC を実行しているサービスグループに対して，flags 引数に DCNOFLAGS を指定した rpcCall メソッド（同期応答型 RPC）を実行する。※

  ・連鎖型 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。

  注※

  同期応答型 RPC を実行しないで closeConnection メソッドまたは rpcClose メソッドを呼び出すと，TP1/Client/J の API は正常終了しますが，次の状態となります。

  ・グローバルトランザクションの範囲外の場合

  サービスを実行していたプロセスは連鎖型 RPC タイムアウトになるまで占有されます。

  ・グローバルトランザクションの範囲内の場合

  暗黙的にコミットされ，連鎖型 RPC を終了します。

DCRPC_TPNOTRAN を指定すると，トランザクションの処理からの RPC は，トランザクションの処理にしないサービス要求になります。トランザクションの処理からだけ DCRPC_TPNOTRAN を指定できます。DCRPC_TPNOTRAN は，DCNOFLAGS，DCRPC_NOREPLY，または DCRPC_CHAINED とあわせて指定できます。

（例）`flags = TP1Client.DCNOFLAGS|TP1Client.DCRPC_TPNOTRAN;`

戻り値

なし。

例外

ErrInvalidArgsException

引数の指定に誤りがあります。この場合，詳細メッセージに誤った引数名が設定されます。

ErrProtoException

メソッドの発行順序に誤りがあります。openConnection メソッドが呼び出されていません。

ErrNoBufsException

メモリ不足が発生しました。

ErrNetDownException

ネットワーク障害が発生しました。または，通信先の TP1/Server が稼働していません。

ErrTimedOutException

次の要因が考えられます。

- このメソッドの処理でタイムアウトが発生しました。
- サービス要求先 SPP が処理を完了する前に異常終了しました。
- dcresponsehost オペランドに指定したホスト名または IP アドレスが誤っているため，応答電文の受信ができず時間切れ（タイムアウト）が発生しました。

ErrMessageTooBigException

in_len 引数に指定した入力パラメタ長が最大値を超えています。

ErrReplyTooBigException

サーバから返された応答の長さが，CUP で用意した領域（out_data 引数の指定値）の長さを超えています。

ErrNoSuchServiceGroupException

group 引数に指定したサービスグループ名は定義されていません。

ErrNoSuchServiceException

service 引数に指定したサービス名は定義されていません。

ErrServiceClosedException

service 引数に指定したサービス名が存在するサービスグループは閉塞されています。

ErrServiceTerminatingException

service 引数に指定したサービスは終了処理中です。

ErrServiceNotUpException

サービス要求した SPP は稼働していません。または，サービス要求した SPP が処理を完了する前に異常終了しました。この例外は TP1/Client/J 環境定義に dcwatchtim=0 を指定（応答を無限に待つ）した場合に返されます。

ErrNotUpException

指定したサービスが存在するノードの TP1/Server が稼働していません。この場合，異常終了，停止中，終了処理中，およびネットワーク障害の発生が考えられます。

ErrSyserrAtServerException

指定したサービスでシステムエラーが発生しました。

ErrSyserrException

システムエラーが発生しました。

ErrNoBufsAtServerException

指定したサービスでメモリ不足が発生しました。

ErrInvalidReplyException

サービス関数が返した応答の長さが 1 から DCRPC_MAX_MESSAGE_SIZE※で指定した値までの範囲にありません。

注※

dccltrpcmaxmsgsize オペランドを使用した場合，DCRPC_MAX_MESSAGE_SIZE の値ではなく，dccltrpcmaxmsgsize オペランドに指定した値になります。

ErrInitializingException

サービス要求したノードにある TP1/Server は開始処理中です。

ErrTrnchkException

ノード間負荷バランス機能を使用している環境で，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 TP1/Server のバージョンが，TP1/Client/J のバージョンよりも古いため，ノード間負荷バランス機能を実行できません。この例外は，ノード間負荷バランス機能を使用している SPP にサービス要求した場合にだけ返されます。

ErrServerBusyException

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。

ErrSecchkException

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。rpcCall メソッドを呼び出した CUP には，SPP へのアクセス権限がありません。

ErrServiceTerminatedException

サービス要求した SPP が，処理を完了する前に異常終了しました。この例外は rap リスナーサービス定義の rpc_extend_function オペランドに 00000001 を指定した場合にだけ返されます。rpc_extend_function オペランドに 00000000 を指定，またはオペランドを省略した場合は，ErrTimedOutException または ErrServiceNotUpException が返されます。

ErrIOErrException

何らかの入出力例外が発生しました。rap サーバが問い合わせ間隔の時間監視でタイムアウトし，コネクションを切断したことも考えられます。

ErrTestmodeException

テストモードの SPP に対してサービス要求を行いました。

ErrConnfreeException

rap サーバとの常設コネクションが切断されました。または，TP1/Web との擬似セションが切断されました。

ErrHostUndefException

次の要因が考えられます。

- 通信先となる TP1/Server のホスト名が TP1/Client/J 環境定義の dchost オペランドに指定されていないか，または指定に誤りがあります。
- TP1/Client/J 環境定義の dcweburl オペランドに指定された URL（プロトコル，WWW サーバ，プロンプターの CGI 名称，TP1/Web のサービス名などの情報）に誤りがあります。

ErrInvalidPortException

次の要因が考えられます。

- リモート API 機能を使用した RPC を行う場合，TP1/Client/J 環境定義の dcrapport オペランドが指定されていない
- スケジューラダイレクト機能を使用した RPC を行う場合，TP1/Client/J 環境定義の dcscdport オペランドが指定されていない

注意事項

- 入力パラメタ（in_data 引数の指定値）と，サービス関数の応答（out_data 引数の指定値）に同じ領域を指定しないでください。
- flags 引数に DCRPC_NOREPLY を指定した場合，次の例外は返されません。

発生しない例外

- ErrReplyTooBigException
- ErrInvalidReplyException

発生しても検出できない例外

- ErrNoSuchServiceException
- ErrServiceClosedException
- ErrServiceTerminatingException
- ErrSyserrAtServerException
- ErrNoBufsAtServerException
- ErrNotUpException

- ErrTimedOutException が返される場合，次に示す要因が考えられます。

  ・TP1/Client/J 環境定義または TP1/Server の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が異常終了した

  ・サービス要求先の SPP が存在するノードに障害が発生した

  ・サービス要求先の SPP の処理完了前に異常終了した

  ・ネットワーク障害が発生した

  ・dcresponsehost オペランドに指定したホスト名または IP アドレスが誤っているため，応答電文の受信ができず時間切れ（タイムアウト）が発生した

  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。

- このメソッドの呼び出し時，何らかの障害が発生した場合には例外が返されますが，その障害によって rap リスナー，rap サーバとの常設コネクションが切断されたかどうかは，その時点では判断できません。この場合，再びこのメソッドを呼び出すと，rap リスナー，rap サーバとの常設コネクションが切断されていれば ErrConnfreeException が返されます。
- このメソッドの呼び出し時，何らかの障害で rap リスナー，rap サーバとの常設コネクションが切断された場合，再び openConnection メソッドを呼び出して rap リスナー，rap サーバとの常設コネクションを確立する必要があります。

- オートコネクトモードを使用した常設コネクションの場合，サービス要求送信時に rap サーバとのネットワーク障害が発生すると，一度だけリトライ処理を行います。リトライしてもコネクションが確立されない場合は，ErrNetDownException が返されます。
- スケジューラダイレクト機能を使用した RPC，およびネームサービスを使用した RPC では，flags 引数に DCRPC_CHAINED を指定できません。リモート API 機能を使用する場合は，flags 引数に DCRPC_CHAINED を指定できます。

●rpcCall

```
public void rpcCall(String group,
                    String service,
                    byte[] in_data,
                    byte[] out_data,
                    int flags)
             throws ErrInvalidArgsException,
                    ErrProtoException,
                    ErrNoBufsException,
                    ErrNetDownException,
                    ErrTimedOutException,
                    ErrMessageTooBigException,
                    ErrReplyTooBigException,
                    ErrNoSuchServiceGroupException,
                    ErrNoSuchServiceException,
                    ErrServiceClosedException,
                    ErrServiceTerminatingException,
                    ErrServiceNotUpException,
                    ErrNotUpException,
                    ErrSyserrAtServerException,
                    ErrSyserrException,
                    ErrNoBufsAtServerException,
                    ErrInvalidReplyException,
                    ErrInitializingException,
                    ErrTrnchkException,
                    ErrServerBusyException,
                    ErrSecchkException,
                    ErrServiceTerminatedException,
                    ErrIOErrException,
                    ErrTestmodeException,
                    ErrConnfreeException,
                    ErrHostUndefException,
                    ErrInvalidPortException
```

SPP のサービスを要求します。サービスグループ名とサービス名の組み合わせで識別されるサービス関数を呼び出し，その応答を受け取ります。このメソッドを呼び出したときに，目的のサービスグループが閉塞されていると ErrServiceClosedException が返されます。

flags 引数に DCRPC_TPNOTRAN を指定すると，トランザクションの処理からの RPC をトランザクションの処理ではないサービス要求にできます。

このメソッドを呼び出したときに，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，ErrServiceTerminatingException，ErrServiceClosedException，または

ErrNoSuchServiceGroupException のどれかが返されます。どの例外が返されるかは，このメソッドを呼び出したタイミングによって異なります。

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 および max_socket_msglen オペランドの指定で電文の輻輳制御を行っています。そのためサーバがサービス要求を受信できない場合があります。この場合，このメソッドを呼び出すと ErrServerBusyException が返されます。この例外が返されたあと，適当な時間を置いてから再度このメソッドを呼び出すと，サービス要求できることがあります。

パラメタ

group

サービスグループ名を 31 文字以内の識別子で指定します。

service

サービス名を 31 文字以内の識別子で指定します。

in_data

サービスの入力パラメタを指定します。

out_data

サービス関数で指定した応答が返される領域を指定します。

非応答型 RPC の場合は null を指定します。非応答型 RPC の場合，null 以外の値を指定しても何も格納されません。

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：連鎖型 RPC

DCRPC_TPNOTRAN：トランザクションを引き継がない RPC

DCNOFLAGS または DCRPC_CHAINED を指定すると，サーバから応答が返されるか，TP1/Client/J 環境定義の dcwatchtim オペランドで指定した最大応答待ち時間が経過してタイムアウトになるまで，このメソッドは制御を戻しません。ただし，サービス要求先の SPP が異常終了した場合は，すぐに例外を返します。この場合，TP1/Client/J 環境定義の dcwatchtim オペランドで指定された最大応答待ち時間によって，次の例外を返します。

- 1～65535 の値を指定した場合
  ErrTimedOutException
- 0 を指定した場合
  ErrServiceNotUpException

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，このメソッドはサービスの実行終了を待たないですぐに制御を戻します。このフラグを指定した場合，応答（out_data 引数の指定値）と応答の長さ（out_len 引数の指定値）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 側からはわかりません。

DCRPC_CHAINED を指定すると，同じサービスグループに属するサービスを複数回要求する場合は，最初の要求時と同一のプロセスで実行されます。

連鎖型 RPC を使用する場合，次に示す制限があります。

- 2 回目以降のこのメソッドの呼び出しは，ユーザサーバおよびサービスの閉塞を検出できません。
- 2 回目以降のこのメソッドの呼び出しでサービス関数の処理に異常が発生した場合，サービス単位ではなくユーザサーバ全体が閉塞します。
- flags 引数に DCRPC_CHAINED を指定して呼び出したサービスの場合，最後のサービス要求では DCNOFLAGS を指定して RPC を実行してください。最後のサービス要求で DCNOFLAGS を指定しないで closeConnection メソッドを呼び出すと，ErrTimedOutException，またはほかの例外が返されます。また，サービスを実行していたプロセスは連鎖型 RPC タイムアウトになるまでプロセスを占有されます。
- リモート API 機能を使用する場合に限り，flags 引数に DCRPC_CHAINED を指定できます。
- 連鎖型 RPC は，次のどちらかの方法で終了させてください。

  ・連鎖型 RPC を実行しているサービスグループに対して，flags 引数に DCNOFLAGS を指定した rpcCall メソッド（同期応答型 RPC）を実行する。※

  ・連鎖型 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。

  注※

  同期応答型 RPC を実行しないで closeConnection メソッド，または rpcClose メソッドを呼び出すと，TP1/Client/J の API は正常終了しますが，次の状態となります。

  ・グローバルトランザクションの範囲外の場合

  サービスを実行していたプロセスは連鎖型 RPC タイムアウトになるまで占有されます。

  ・グローバルトランザクションの範囲内の場合

  暗黙的にコミットされ，連鎖型 RPC を終了します。

DCRPC_TPNOTRAN を指定すると，トランザクションの処理からの RPC は，トランザクションの処理にしないサービス要求になります。トランザクション外の処理からは DCRPC_TPNOTRAN を指定できません。DCRPC_TPNOTRAN は，DCNOFLAGS，DCRPC_NOREPLY，または DCRPC_CHAINED とあわせて指定できます。

（例）`flags = TP1Client.DCNOFLAGS|TP1Client.DCRPC_TPNOTRAN;`

戻り値

なし。

例外

ErrInvalidArgsException

引数の指定に誤りがあります。この場合，詳細メッセージに誤った引数名が設定されます。

ErrProtoException

メソッドの発行順序に誤りがあります。openConnection メソッドが呼び出されていません。

ErrNoBufsException

メモリ不足が発生しました。

ErrNetDownException

ネットワーク障害が発生しました。または，通信先の TP1/Server が稼働していません。

ErrTimedOutException

次の要因が考えられます。

- このメソッドの処理でタイムアウトが発生しました。
- サービス要求先 SPP が処理を完了する前に異常終了しました。
- dcresponsehost オペランドに指定したホスト名または IP アドレスが誤っているため，応答電文の受信ができず時間切れ（タイムアウト）が発生しました。

ErrMessageTooBigException

in_len 引数に指定した入力パラメタ長が最大値を超えています。

ErrReplyTooBigException

サーバから返された応答の長さが，CUP で用意した領域（out_data 引数の指定値）の長さを超えています。

ErrNoSuchServiceGroupException

group 引数に指定したサービスグループ名は定義されていません。

ErrNoSuchServiceException

service 引数に指定したサービス名は定義されていません。

ErrServiceClosedException

service 引数に指定したサービス名が存在するサービスグループは閉塞されています。

ErrServiceTerminatingException

service 引数に指定したサービスは終了処理中です。

ErrServiceNotUpException

サービス要求した SPP は稼働していません。または，サービス要求した SPP が処理を完了する前に異常終了しました。この例外は TP1/Client/J 環境定義に dcwatchtim=0 を指定（応答を無限に待つ）した場合に返されます。

ErrNotUpException

指定したサービスが存在するノードの TP1/Server が稼働していません。この場合，異常終了，停止中，終了処理中，およびネットワーク障害の発生が考えられます。

ErrSyserrAtServerException

指定したサービスでシステムエラーが発生しました。

ErrSyserrException

システムエラーが発生しました。

ErrNoBufsAtServerException

指定したサービスでメモリ不足が発生しました。

ErrInvalidReplyException

サービス関数が返した応答の長さが 1 から DCRPC_MAX_MESSAGE_SIZE※で指定した値までの範囲にありません。

注※

dccltrpcmaxmsgsize オペランドを使用した場合，DCRPC_MAX_MESSAGE_SIZE の値ではなく，dccltrpcmaxmsgsize オペランドに指定した値になります。

ErrInitializingException

サービス要求したノードにある TP1/Server は開始処理中です。

ErrTrnchkException

ノード間負荷バランス機能を使用している環境で，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 TP1/Server のバージョンが，TP1/Client/J のバージョンよりも古いため，ノード間負荷バランス機能を実行できません。この例外は，ノード間負荷バランス機能を使用している SPP にサービス要求した場合にだけ返されます。

ErrServerBusyException

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。

ErrSecchkException

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。rpcCall メソッドを呼び出した CUP には，SPP へのアクセス権限がありません。

ErrServiceTerminatedException

サービス要求した SPP が，処理を完了する前に異常終了しました。この例外は rap リスナーサービス定義の rpc_extend_function オペランドに 00000001 を指定した場合にだけ返されます。rpc_extend_function オペランドに 00000000 を指定，またはオペランドを省略した場合は，ErrTimedOutException または ErrServiceNotUpException が返されます。

ErrIOErrException

何らかの入出力例外が発生しました。rap サーバが問い合わせ間隔の時間監視でタイムアウトし，コネクションを切断したことも考えられます。

ErrTestmodeException

テストモードの SPP に対してサービス要求を行いました。

ErrConnfreeException

rap サーバとの常設コネクションが切断されました。または，TP1/Web との擬似セションが切断されました。

ErrHostUndefException

次の要因が考えられます。

- 通信先となる TP1/Server のホスト名が TP1/Client/J 環境定義の dchost オペランドに指定されていないか，または指定に誤りがあります。
- TP1/Client/J 環境定義の dcweburl オペランドに指定された URL（プロトコル，WWW サーバ，プロンプターの CGI 名称，TP1/Web のサービス名などの情報）に誤りがあります。

ErrInvalidPortException

次の要因が考えられます。

- リモート API 機能を使用した RPC を行う場合，TP1/Client/J 環境定義の dcrapport オペランドが指定されていない
- スケジューラダイレクト機能を使用した RPC を行う場合，TP1/Client/J 環境定義の dcscdport オペランドが指定されていない

注意事項

- 入力パラメタ（in_data 引数の指定値）と，サービス関数の応答（out_data 引数の指定値）に同じ領域を指定しないでください。
- flags 引数に DCRPC_NOREPLY を指定した場合，次の例外は返されません。

発生しない例外

- ErrReplyTooBigException
- ErrInvalidReplyException

発生しても検出できない例外

- ErrNoSuchServiceException
- ErrServiceClosedException
- ErrServiceTerminatingException
- ErrSyserrAtServerException
- ErrNoBufsAtServerException
- ErrNotUpException

- ErrTimedOutException が返される場合，次に示す要因が考えられます。

・TP1/Client/J 環境定義または TP1/Server の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が異常終了した

・サービス要求先の SPP が存在するノードに障害が発生した

・サービス要求先の SPP の処理完了前に異常終了した

・ネットワーク障害が発生した

・dcresponsehost オペランドに指定したホスト名または IP アドレスが誤っているため，応答電文の受信ができず時間切れ（タイムアウト）が発生した

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。

- このメソッドの呼び出し時，何らかの障害が発生した場合には例外が返されますが，その障害によって rap リスナー，rap サーバとの常設コネクションが切断されたかどうかは，その時点では判断できません。この場合，再びこのメソッドを呼び出すと，rap リスナー，rap サーバとの常設コネクションが切断されていれば ErrConnfreeException が返されます。
- このメソッドの呼び出し時，何らかの障害で rap リスナー，rap サーバとの常設コネクションが切断された場合，再び openConnection メソッドを呼び出して rap リスナー，rap サーバとの常設コネクションを確立する必要があります。
- オートコネクトモードを使用した常設コネクションの場合，サービス要求送信時に rap サーバとのネットワーク障害が発生すると，一度だけリトライ処理を行います。リトライしてもコネクションが確立されない場合は，ErrNetDownException が返されます。
- スケジューラダイレクト機能を使用した RPC，およびネームサービスを使用した RPC では，flags 引数に DCRPC_CHAINED を指定できません。リモート API 機能を使用する場合は，flags 引数に DCRPC_CHAINED を指定できます。

●rpcCallTo

```
public void rpcCallTo(DCRpcBindTbl direction,
                      String group,
                      String service,
                      byte[] in_data,
                      int[] in_len,
                      byte[] out_data,
                      int[] out_len,
                      int flags)
              throws ErrInvalidArgsException,
                     ErrProtoException,
                     ErrNoBufsException,
                     ErrNetDownException,
                     ErrTimedOutException,
                     ErrMessageTooBigException,
                     ErrReplyTooBigException,
                     ErrNoSuchServiceGroupException,
                     ErrNoSuchServiceException,
                     ErrServiceClosedException,
                     ErrServiceTerminatingException,
                     ErrServiceNotUpException,
                     ErrNotUpException,
                     ErrSyserrAtServerException,
                     ErrSyserrException,
                     ErrNoBufsAtServerException,
                     ErrInvalidReplyException,
                     ErrInitializingException,
                     ErrTrnchkException,
                     ErrServerBusyException,
                     ErrSecchkException,
                     ErrServiceTerminatedException,
                     ErrIOErrException,
                     ErrTestmodeException,
                     ErrConnfreeException,
                     ErrHostUndefException,
                     ErrInvalidPortException
```

rpcCall メソッドと同様に，SPP のサービスを要求します。rpcCallTo メソッドでは，サービスグループ名，サービス名，およびホスト名を検索のキーにして，該当するサービス関数をサービスの要求先に限定します。

このメソッドを呼び出す前に setScdDirectObject メソッドを呼び出し，DCRpcBindTbl インスタンスを作成しておく必要があります。それ以外のインタフェースは，rpcCall メソッドと同じです。

パラメタ

direction

DCRpcBindTbl オブジェクトを指定します。このメソッドを呼び出す前に setScdDirectObject メソッドを呼び出し，通信先情報を設定しておきます。

group

サービスグループ名を 31 文字以内の識別子で指定します。

service

サービス名を 31 文字以内の識別子で指定します。

in_data

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 1 から DCRPC_MAX_MESSAGE_SIZE※の指定値までの範囲で指定します。入力パラメタ長は in_len[0]に格納してください。

注※

dccltrpcmaxmsgsize オペランドを使用した場合，DCRPC_MAX_MESSAGE_SIZE の値ではなく，dccltrpcmaxmsgsize オペランドに指定した値になります。

out_data

サービス関数で指定した応答が返される領域を指定します。

非応答型 RPC の場合は null を指定します。非応答型 RPC の場合，null 以外の値を指定しても何も格納されません。

out_len

サービスの応答の長さを 1 から DCRPC_MAX_MESSAGE_SIZE※の指定値までの範囲で指定します。応答の長さは out_len[0]に格納してください。

注※

dccltrpcmaxmsgsize オペランドを使用した場合，DCRPC_MAX_MESSAGE_SIZE の値ではなく，dccltrpcmaxmsgsize オペランド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

戻り値

なし。

例外

rpcCall メソッドの例外と同じ。

注意事項

- ソケット受信型のユーザサーバにサービス要求を行った場合，このメソッドは，ErrServiceNotUpException を返します。
- サービス要求先の TP1/Server のバージョンは，03-03 以降でなければなりません。これ以前のバージョンの TP1/Server をサービス要求先に指定した場合は，動作の保証はできません。
- このメソッドは，リモート API 機能を使用した RPC では使用できません。リモート API 機能を使用しているときにこのメソッドを発行した場合，このメソッドは，ErrProtoException を返します。
- このメソッドは，連鎖型 RPC では使用できません。flags 引数に DCRPC_CHAINED を指定した場合，このメソッドは ErrInvalidArgsException を返します。
- rpcCallTo メソッドの呼び出し時は，TP1/Client/J 環境定義の次のオペランドは参照されません。
  ・dchost
  ・dcscdport
  ・dcnamport
  ・dcscdloadpriority
  また，setDchost メソッドで指定されたホスト名，ポート番号も参照されません。これらの指定値は，次回 rpcCall メソッドの呼び出し時に有効になります。
- サービス要求先のホスト名に誤りがあった場合，このメソッドは ErrHostUndefException を返します。
- サービス要求先のポート番号に誤りがあった場合，このメソッドは ErrInvalidPortException を返します。

●setScdDirectObject

```
public DCRpcBindTbl setScdDirectObject(String scdhost,
                                       int scdport,
                                       int flags)
```

通信先スケジューラのホスト名，ポート番号を設定します。

パラメタ

scdhost

通信先スケジュールサーバのホスト名を指定します。不正なホスト名，および null が指定された場合，このメソッドのあとに呼び出す rpcCallTo メソッドは，ErrHostUndefException を返します。

scdport

通信先スケジュールサーバのポート番号を指定します。ポート番号は，5001 から 65535 までの範囲で指定します。不正なポート番号が指定された場合，このメソッドのあとに呼び出す rpcCallTo メソッドは，ErrInvalidPortException を返します。

flags

DCNOFLAGS を指定します。

戻り値

DCRpcBindTbl オブジェクト。

●trnBegin

```
public void trnBegin()
              throws ErrProtoException,
                     ErrTMException,
                     ErrRMException,
                     ErrNoBufsException,
                     ErrNotUpException,
                     ErrTimedOutException,
                     ErrIOErrException,
                     ErrConnfreeException,
                     ErrSyserrException,
                     ErrNetDownException,
                     ErrHostUndefException,
                     ErrInvalidPortException
```

グローバルトランザクションを，trnBegin メソッドを呼び出す TP1Client オブジェクトから開始します。

このメソッドは，リモート API 機能を使用する場合で，要求先 TP1/Server Base のバージョンが 05-00 以降のときだけ使用できます。

パラメタ

なし。

戻り値

なし。

例外

ErrProtoException

trnBegin メソッドを不正なコンテキストから呼び出しています。

ErrTMException

トランザクションサービスでエラーが発生したため，トランザクションは開始できませんでした。この例外が返されたあと，適当な時間を置いてから再度このメソッドを呼び出すと，トランザクションを開始できることがあります。

ErrRMException

リソースマネジャでエラーが発生しました。トランザクションは開始できませんでした。

ErrNoBufsException

メモリ不足が発生しました。

ErrNotUpException

TP1/Server が稼働していません。

ErrTimedOutException

タイムアウトが発生しました。

ErrIOErrException

何らかの入出力例外が発生しました。

ErrConnfreeException

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。

ErrSyserrException

システムエラーが発生しました。

ErrNetDownException

ネットワーク障害が発生しました。

ErrHostUndefException

通信先となる TP1/Server のホスト名が TP1/Client/J 環境定義の dchost オペランドに指定されていないか，または指定に誤りがあります。

ErrInvalidPortException

TP1/Client/J 環境定義の dcrapport オペランドに指定されたポート番号が不正，または未指定です。

●trnChainedCommit

```
public void trnChainedCommit()
                        throws ErrProtoException,
                               ErrRollbackException,
                               ErrHeuristicException,
                               ErrHazardException,
                               ErrNoBeginException,
                               ErrRollbackNoBeginException,
                               ErrHeuristicNoBeginException,
                               ErrHazardNoBeginException,
                               ErrNoBufsException,
                               ErrNotUpException,
                               ErrTimedOutException,
                               ErrIOErrException,
                               ErrConnfreeException,
                               ErrNetDownException,
                               ErrSyserrException
```

トランザクションの同期点を取得します。

trnChainedCommit メソッド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，以降実行するメソッド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。

パラメタ

なし。

戻り値

なし。

例外

ErrProtoException

trnChainedCommit メソッドを不正なコンテキストから呼び出しています。

ErrRollbackException

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。この例外が返されたあとも，このプロセスはトランザクション下にあり，グローバルトランザクションの範囲内です。

ErrHeuristicException

dc_trn_chained_commit 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，あるトランザクションブランチはコミット，あるトランザクションブランチはロールバックしました。この例外は，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い場合に返されます。この例外が返される原因になった UAP，リソースマネジャ，およびグローバルトランザクションの同期点の結果は，TP1/Server のメッセージログファイルを参照してください。この例外が返されたあとも，このプロセスはトランザクション下にあり，グローバルトランザクションの範囲内です。

ErrHazardException

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わかりません。この例外が返される原因になった UAP，リソースマネジャ，およびグローバルトランザクションの同期点の結果は，TP1/Server のメッセージログファイルを参照してください。この例外が返されたあとも，このプロセスはトランザクション下にあり，グローバルトランザクションの範囲内です。

ErrNoBeginException

コミットまたはロールバック処理は正常に終了しましたが，新しいトランザクションは開始できませんでした。この例外が返されたあと，このプロセスはトランザクション下にありません。

ErrRollbackNoBeginException

コミットしようとした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。新しいトランザクションは開始できませんでした。この例外が返されたあと，このプロセスはトランザクション下にありません。

ErrHeuristicNoBeginException

dc_trn_chained_commit 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，あるトランザクションブランチはコミット，あるトランザクションブランチはロールバックしました。この例外は，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い場合に返されます。この例外が返される原因になった UAP，リソースマネジャ，およびグローバルトランザクションの同期点の結果は，TP1/Server のメッセージログファ

イルを参照してください。新しいトランザクションは開始できませんでした。この例外が返されたあと，このプロセスはトランザクション下にありません。

ErrHazardNoBeginException

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わかりません。この例外が返される原因になったUAP，リソースマネジャ，およびグローバルトランザクションの同期点の結果は，TP1/Serverのメッセージログファイルの内容を参照してください。新しいトランザクションは開始できませんでした。この例外が返されたあと，このプロセスはトランザクション下にありません。

ErrNoBufsException

メモリ不足が発生しました。

ErrNotUpException

TP1/Serverが稼働していません。

ErrTimedOutException

タイムアウトが発生しました。

ErrIOErrException

何らかの入出力例外が発生しました。

ErrConnfreeException

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。

ErrNetDownException

ネットワーク障害が発生しました。

ErrSyserrException

システムエラーが発生しました。

●trnChainedRollback

```
public void trnChainedRollback()
                        throws ErrProtoException,
                               ErrHeuristicException,
                               ErrHazardException,
                               ErrNoBeginException,
                               ErrHeuristicNoBeginException,
                               ErrHazardNoBeginException,
                               ErrNoBufsException,
                               ErrNotUpException,
                               ErrTimedOutException,
                               ErrIOErrException,
                               ErrConnfreeException,
                               ErrNetDownException,
                               ErrSyserrException
```

トランザクションをロールバックします。

trnChainedRollback メソッドが正常終了すると，新しいグローバルトランザクションが発生し，以降呼び出すメソッド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。

パラメタ

なし。

戻り値

なし。

例外

ErrProtoException

trnChainedRollback メソッドを不正なコンテキストから呼び出しています。

ErrHeuristicException

dc_trn_chained_rollback 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，あるトランザクションブランチはコミット，あるトランザクションブランチはロールバックしました。この例外は，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い場合に返されます。この例外が返される原因になった UAP，リソースマネジャ，およびグローバルトランザクションの同期点の結果は，TP1/Server のメッセージログファイルを参照してください。この例外が返されたあとも，このプロセスはトランザクション下にあり，グローバルトランザクションの範囲内です。

ErrHazardException

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わかりません。この例外が返される原因になった UAP，リソースマネジャ，およびグローバルトランザクションの同期点の結果は，TP1/Server のメッセージログファイルを参照してください。この例外が返されたあとも，このプロセスはトランザクション下にあり，グローバルトランザクションの範囲内です。

ErrNoBeginException

コミットまたはロールバック処理は正常に終了しましたが，新しいトランザクションは開始できませんでした。この例外が返されたあと，このプロセスはトランザクション下にありません。

ErrHeuristicNoBeginException

dc_trn_chained_rollback 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，あるトランザクションブランチはコミット，あるトランザクションブランチはロールバックしました。この例外は，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い場合に返されます。この例外が返される原因になった UAP，リソースマネジャ，およびグローバルトランザクションの同期点の結果は，TP1/Server のメッセージログファイルの内容を参照してください。新しいトランザクションは開始できませんでした。この例外が返されたあと，このプロセスはトランザクション下にありません。

ErrHazardNoBeginException

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

わかりません。この例外が返される原因となったUAP，リソースマネジャ，およびグローバルトランザクションの同期点の結果は，TP1/Serverのメッセージログファイルの内容を参照してください。新しいトランザクションは開始できませんでした。この例外が返されたあと，このプロセスはトランザクション下にありません。

ErrNoBufsException

メモリ不足が発生しました。

ErrNotUpException

TP1/Serverが稼働していません。

ErrTimedOutException

タイムアウトが発生しました。

ErrIOErrException

何らかの入出力例外が発生しました。

ErrConnfreeException

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。

ErrNetDownException

ネットワーク障害が発生しました。

ErrSyserrException

システムエラーが発生しました。

●trnUnchainedCommit

```
public void trnUnchainedCommit()
                        throws ErrProtoException,
                               ErrRollbackException,
                               ErrHeuristicException,
                               ErrHazardException,
                               ErrNoBufsException,
                               ErrNotUpException,
                               ErrTimedOutException,
                               ErrIOErrException,
                               ErrConnfreeException,
                               ErrNetDownException,
                               ErrSyserrException
```

トランザクションの同期点を取得します。

trnUnchainedCommitメソッド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。

パラメタ

なし。

戻り値

なし。

例外

ErrProtoException

trnUnchainedCommit メソッドを不正なコンテキストから呼び出しています。

ErrRollbackException

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。プロセスはトランザクションの範囲外です。

ErrHeuristicException

dc_trn_unchained_commit 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，あるトランザクションブランチはコミット，あるトランザクションブランチはロールバックしました。この例外は，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い場合に返されます。この例外が返される原因になった UAP，リソースマネジャ，およびグローバルトランザクションの同期点の結果は，TP1/Server のメッセージログファイルを参照してください。この例外が返されたあと，このプロセスはトランザクション下にありません。プロセスはグローバルトランザクションの範囲外です。

ErrHazardException

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わかりません。この例外が返される原因になった UAP，リソースマネジャ，およびグローバルトランザクションの同期点の結果は，TP1/Server のメッセージログファイルを参照してください。この例外が返されたあと，このプロセスはトランザクション下にありません。プロセスはグローバルトランザクションの範囲外です。

ErrNoBufsException

メモリ不足が発生しました。

ErrNotUpException

TP1/Server が稼働していません。

ErrTimedOutException

タイムアウトが発生しました。

ErrIOErrException

何らかの入出力例外が発生しました。

ErrConnfreeException

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。

ErrNetDownException

ネットワーク障害が発生しました。

ErrSyserrException

システムエラーが発生しました。

●trnUnchainedRollback

```
public void trnUnchainedRollback()
                          throws ErrProtoException,
                                 ErrHeuristicException,
                                 ErrHazardException,
                                 ErrNoBufsException,
                                 ErrNotUpException,
                                 ErrTimedOutException,
                                 ErrIOErrException,
                                 ErrConnfreeException,
                                 ErrNetDownException,
                                 ErrSyserrException
```

トランザクションをロールバックします。

trnUnchainedRollback メソッドを呼び出すと，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。

パラメタ

なし。

戻り値

なし。

例外

ErrProtoException

trnUnchainedRollback メソッドを不正なコンテキストから呼び出しています。

ErrHeuristicException

dc_trn_unchained_rollback 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，あるトランザクションブランチはコミット，あるトランザクションブランチはロールバックしました。この例外は，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い場合に返されます。この例外が返される原因になった UAP，リソースマネジャ，およびグローバルトランザクションの同期点の結果は，TP1/Server のメッセージログファイルを参照してください。この例外が返されたあと，このプロセスはトランザクション下にありません。プロセスはグローバルトランザクションの範囲外です。

ErrHazardException

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わかりません。この例外が返される原因になった UAP，リソースマネジャ，およびグローバルトランザクションの同期点の結果は，TP1/Server のメッセージログファイルを参照してください。この例外が返されたあと，このプロセスはトランザクション下にありません。プロセスはグローバルトランザクションの範囲外です。

ErrNoBufsException

メモリ不足が発生しました。

ErrNotUpException

TP1/Server が稼働していません。

ErrTimedOutException

タイムアウトが発生しました。

ErrIOErrException

何らかの入出力例外が発生しました。

ErrConnfreeException

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。

ErrNetDownException

ネットワーク障害が発生しました。

ErrSyserrException

システムエラーが発生しました。

●trnInfo

```
public boolean trnInfo()
```

trnInfo メソッドを呼び出した TP1Client オブジェクトが，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。このメソッドは内部の変数を参照するだけで，rap サーバとは通信しません。

パラメタ

なし。

戻り値

true

trnInfo メソッドを呼び出した TP1Client オブジェクトは，トランザクションの範囲内にあります。

false

trnInfo メソッドを呼び出した TP1Client オブジェクトは，トランザクションの範囲外にあります。

例外

なし。

●getTrnID

```
public void getTrnID(byte[] gid,byte[] bid)
              throws ErrInvalidArgsException,
                     ErrProtoException
```

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。このメソッドは内部の変数を参照するだけで，rap サーバとは通信しません。

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は，次に示すメソッドを呼び出してトランザクションが起動したときに，TP1/Server が割り当てたものです。

- trnBegin メソッド
- trnChainedCommit メソッド
- trnChainedRollback メソッド

パラメタ

gid

トランザクショングローバル識別子を格納する 16 バイト以上の byte 型配列を指定します。

bid

トランザクションブランチ識別子を格納する 16 バイト以上の byte 型配列を指定します。

戻り値

なし。

例外

ErrInvalidArgsException

引数の指定に誤りがあります。

ErrProtoException

getTrnID メソッドを不正なコンテキストから呼び出しています。

●getRpcServicePrio

```
public int getRpcServicePrio()
```

setRpcServicePrio メソッドで設定した，サービス要求のスケジュールプライオリティを取得します。

このメソッドが取得するスケジュールプライオリティの値は，CUP が再び setRpcServicePrio メソッドを呼び出すか，または rpcClose メソッドを呼び出すまで変わりません。

次に示す場合，getRpcServicePrio メソッドは省略時解釈値である 4 を取得します。

- CUP で setRpcServicePrio メソッドを呼び出していない場合
- setRpcServicePrio メソッドの引数 prio に 0 を設定して呼び出した場合
- rpcClose メソッドを呼び出した場合

パラメタ

なし。

戻り値

setRpcServicePrio メソッドで設定したスケジュールプライオリティを，1～8 までの整数で取得します。

例外

なし。

●cltReceive

```
public void cltReceive(byte[]  buff,
                       int[]   recvleng,
                       int     timeout,
                       int     flags)
                throws ErrInvalidArgsException,
                       ErrProtoException,
                       ErrNetDownException,
                       ErrTimedOutException,
                       ErrSyserrException,
                       ErrInvalidPortException,
                       ErrConnfreeException
```

MHP が送信したメッセージを受信します。

cltReceive メソッドを実行する場合，TP1/Client/J 環境定義の dcsndrcvtype オペランドに DCCLT_ONEWAY_RCV または DCCLT_SNDRCV を指定して，rpcOpen メソッドをあらかじめ実行しておく必要があります。

パラメタ

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。メソッド実行後は受信したメッセージが返されます。

recvleng

recvleng[0]に受信するメッセージの長さを指定します。メソッド実行後は受信したメッセージの長さが返されます。

timeout

メッセージ受信時の最大待ち時間を，-1 から 65535 までの整数（単位：秒）で指定します。

-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，ErrTimedOutException が返されます。

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，ErrTimedOutException が返されます。

flags

メッセージ受信後に，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。

DCNOFLAGS を指定した場合は，障害時を除き，rpcClose メソッドを実行するまで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。

戻り値

なし。

例外

ErrInvalidArgsException

引数の指定に誤りがあります。

ErrProtoException

rpcOpen メソッドが実行されていません。または，rpcOpen メソッドは実行されていますが，TP1/Client/J 環境定義の dcsndrcvtype オペランドに DCCLT_ONEWAY_RCV もしくは DCCLT_SNDRCV を指定していません。

ErrNetDownException

ネットワーク障害が発生しました。

ErrTimedOutException

メッセージ受信時にタイムアウトしました。

ErrSyserrException

システムエラーが発生しました。

ErrInvalidPortException

この例外は，TP1/Client/J 環境定義の dcsockopenatrcv オペランドに Y を指定し，dcrcvport オペランドに指定したポートがすでに使用中の場合に発生します。dcsockopenatrcv オペランドに N を指定した場合は，rpcOpen メソッドが ErrFatalException を返します。

ErrConnfreeException

MHP からコネクションが解放されました。

注意事項

cltReceive メソッドは，次に示す場合に CUP に制御を戻します。

- MHP から recvleng 引数で指定した長さ分のメッセージを受信した場合
- MHP からのメッセージ受信時に，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

cltReceive メソッドの発行時に，MHP からコネクションが解放された場合，ErrConnfreeException でエラーリターンします。

●cltSend

```
public void cltSend(byte[]  buff,
                    int     sendleng,
                    String  hostname,
                    int     portnum,
                    int     flags)
             throws ErrInvalidArgsException,
                    ErrProtoException,
                    ErrNetDownException,
                    ErrSyserrException,
                    ErrHostUndefException,
```

```
                    ErrInvalidPortException,
                    ErrConnRefusedException
```

MHP へメッセージを送信します。

cltSend メソッドを実行する場合，TP1/Client/J 環境定義の dcsndrcvtype オペランドに DCCLT_ONEWAY_SND または DCCLT_SNDRCV を指定して，rpcOpen メソッドをあらかじめ実行しておく必要があります。

パラメタ

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 MHP が存在するノードのホスト名を指定します。

null を指定した場合は，rpcOpen メソッドを実行したときに取得した TP1/Client/J 環境定義の dcsndhost オペランドの内容を参照します。

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。

コネクションが確立されている場合，この引数は無視されます。

portnum

コネクションが確立されていない場合，接続する MHP のポート番号を，0 から 65535 までの整数で指定します。

0 を指定すると，rpcOpen メソッドを実行したときに取得した TP1/Client/J 環境定義の dcsndport オペランドの内容を参照します。

コネクションが確立されている場合，この引数は無視されます。

flags

メッセージ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：メッセージ送信後，コネクションを解放しません。

DCNOFLAGS を指定した場合は，障害時を除き，rpcClose メソッドを実行するまでコネクションを解放しません。

DCCLT_SND_CLOSE：メッセージ送信後，コネクションを解放します。

戻り値

なし。

例外

ErrInvalidArgsException

引数の指定に誤りがあります。

ErrProtoException

rpcOpen メソッドが実行されていません。または，rpcOpen メソッドは実行されていますが，TP1/Client/J 環境定義の dcsndrcvtype オペランドに DCCLT_ONEWAY_SND もしくは DCCLT_SNDRCV を指定していません。

ErrNetDownException

ネットワーク障害が発生しました。

ErrSyserrException

システムエラーが発生しました。

ErrHostUndefException

hostname 引数の指定に誤りがあります。または，hostname 引数および TP1/Client/J 環境定義の dcsndhost オペランドの両方にホスト名が指定されていません。

ErrInvalidPortException

portnum 引数の指定に誤りがあります。

ErrConnRefusedException

MHP に対するコネクションの確立要求が拒否されました（接続を待ち受けていないポートに対してコネクションを確立しようとしました）。

注意事項

cltSend メソッドを実行してメッセージを送信しているときに，MHP からコネクションが解放されると，次に実行する cltSend メソッドは正常終了または異常終了します。正常終了した場合は，その次に実行する cltSend メソッドで初めて異常終了します。そのため，CUP を作成するときは注意してください。

●cltAssemSend

```
public void cltAssemSend(byte[] buff,
                         int sendleng,
                         String hostname,
                         int portnum,
                         int timeout,
                         int flags)
                  throws ErrInvalidArgsException,
                         ErrProtoException,
                         ErrNetDownException,
                         ErrSyserrException,
                         ErrHostUndefException,
                         ErrInvalidPortException,
                         ErrConnRefusedException,
                         ErrTimedOutException,
                         ErrConnfreeException,
                         ErrInvalidMessageException,
                         ErrCollisionMessageException
```

受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します。

cltAssemSend メソッドを実行する場合，TP1/Client/J 環境定義の dcsndrcvtype オペランドに DCCLT_ONEWAY_SND または DCCLT_SNDRCV を指定して，rpcOpen メソッドをあらかじめ実行しておく必要があります。

受信メッセージの組み立て機能を使用した場合，メッセージの先頭 4 バイトにメッセージ長（4 ＋ sendleng 引数の指定値）を付けて，buff[0]～buff[sendleng－1]の長さのメッセージを送信します。

相手システムとのコネクションが確立されていない場合，hostname 引数および portnum 引数の指定に従ってコネクションを確立し，メッセージを送信します。

パラメタ

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 引数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，接続する相手システムのホスト名を指定します。

null を指定した場合は，rpcOpen メソッドを実行したときに取得した TP1/Client/J 環境定義の dcsndhost オペランド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。コネクションが確立されている場合，この引数は無視されます。

portnum

コネクションが確立されていない場合，接続する相手システムのポート番号を，0 から 65535 までの整数で指定します。

0 を指定すると，rpcOpen メソッドを実行したときに取得した TP1/Client/J 環境定義の dcsndport オペランドの内容を参照します。コネクションが確立されている場合，この引数は無視されます。

timeout

予約引数です。指定した値は無視されます。

flags

メッセージ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：メッセージ送信後，コネクションを解放しません。

DCNOFLAGS を指定した場合は，障害時を除き，rpcClose メソッドを実行するまでコネクションを解放しません。

DCCLT_SND_CLOSE：メッセージ送信後，コネクションを解放します。

戻り値

なし。

例外

ErrInvalidArgsException

引数の指定に誤りがあります。

ErrProtoException

rpcOpen メソッドが実行されていません。または，rpcOpen メソッドは実行されていますが，TP1/Client/J 環境定義の dcsndrcvtype オペランドで DCCLT_ONEWAY_SND または DCCLT_SNDRCV を指定していません。

ErrNetDownException

ネットワーク障害が発生しました。

ErrSyserrException

システムエラーが発生しました。

ErrHostUndefException

接続する相手システムのホスト名に誤りがあります。または，hostname 引数および TP1/Client/J 環境定義の dcsndhost オペランドの両方にホスト名が指定されていません。

ErrInvalidPortException

portnum 引数の指定に誤りがあります。

ErrConnRefusedException

相手システムに対するコネクションの確立要求が拒否されました（接続を待ち受けていないポートに対してコネクションを確立しようとしました）。

ErrTimedOutException

応答メッセージの受信時に，タイムアウトが発生しました。コネクションは解放されます。

ErrConnfreeException

相手システムからコネクションが解放されました。

ErrInvalidMessageException

不正なメッセージを受信しました。

ErrCollisionMessageException

送受信メッセージの衝突が発生しました。

注意事項

cltAssemSend メソッドを実行してメッセージを送信するときに，相手システムからコネクションが解放されると，次に実行する cltAssemSend メソッドは正常終了または異常終了します。正常終了した場合は，その次に実行する cltAssemSend メソッドで初めて異常終了します。そのため，CUP を作成するときは注意してください。

●cltAssemReceive

```
public void cltAssemReceive(byte[] buff,
                            int[] recvleng,
                            int timeout,
```

```
                int flags)
        throws ErrInvalidArgsException,
               ErrProtoException,
               ErrNetDownException,
               ErrTimedOutException,
               ErrSyserrException,
               ErrInvalidPortException,
               ErrConnfreeException,
               ErrInvalidMessageException,
               ErrBufferOverflowException
```

受信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。

cltAssemReceive メソッドを実行する場合，TP1/Client/J 環境定義の dcsndrcvtype オペランドに DCCLT_ONEWAY_RCV または DCCLT_SNDRCV を指定して，rpcOpen メソッドをあらかじめ実行しておく必要があります。

受信メッセージの組み立て機能を使用した場合，メッセージの先頭 4 バイトは，buff 引数に指定されたバッファに格納されません。

このメソッドが正常終了した場合，recvleng[0]の長さのユーザデータを含むメッセージを受信し，ユーザデータが buff[0]～buff[recvleng[0]－1]に格納されます。

パラメタ

buff

受信したメッセージを格納する領域を指定します。受信するメッセージの長さ以上の領域を指定してください。

recvleng

メソッド実行後に，受信したメッセージの長さが recvleng[0]に返されます。

timeout

メッセージ受信時の最大待ち時間を，-1 から 65535 までの整数（単位：秒）で指定します。

-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，ErrTimedOutException が返されます。

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，ErrTimedOutException が返されます。

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。

DCNOFLAGS を指定した場合は，障害時を除き，rpcClose メソッドを実行するまで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。

戻り値

なし。

例外

ErrInvalidArgsException

引数の指定に誤りがあります。

ErrProtoException

rpcOpen メソッドが実行されていません。または，rpcOpen メソッドは実行されていますが，TP1/Client/J 環境定義の dcsndrcvtype オペランドで DCCLT_ONEWAY_RCV または DCCLT_SNDRCV を指定していません。

ErrNetDownException

ネットワーク障害が発生しました。

ErrTimedOutException

メッセージの受信時にタイムアウトが発生しました。

ErrSyserrException

システムエラーが発生しました。

ErrInvalidPortException

この例外は，TP1/Client/J 環境定義の dcsockopenatrcv オペランドに Y を指定し，dcrcvport オペランドに指定したポートがすでに使用中の場合に発生します。dcsockopenatrcv オペランドに N を指定した場合は，rpcOpen メソッドが ErrFatalException を返します。

ErrConnfreeException

相手システムからコネクションが解放されました。

ErrInvalidMessageException

不正なメッセージを受信しました。

ErrBufferOverflowException

buff 引数の指定値を超える長さのメッセージを受信しました。

注意事項

cltAssemReceive メソッドは，次に示す場合に CUP に制御を戻します。

- メッセージ受信が完了した場合
- buff 引数の指定値を超える長さ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
- 不正なメッセージ長のメッセージを受信した場合
- 不正なセグメント情報のメッセージを受信した場合

cltAssemReceive メソッドの発行時に，相手システムからコネクションが解放された場合，ErrConnfreeException でエラーリターンします。

●setConnectInformation

```
public void setConnectInformation(byte[] inf,
                                  short inf_len)
                           throws ErrInvalidArgsException
```

端末識別情報を設定します。

このメソッドに指定した端末識別情報は，TP1/Client/J 環境定義の dchost オペランドに DCCM3 論理端末のホスト名を，dchost オペランドまたは dcrapport オペランドに DCCM3 論理端末のポート番号を指定し，次のどちらかの方法で DCCM3 論理端末との常設コネクションを確立した場合に有効となります。

- openConnection メソッドを呼び出します。引数有りの openConnection メソッドの場合，引数 host に DCCM3 論理端末のホスト名，引数 port に DCCM3 論理端末のポート番号を指定します。
- TP1/Client/J 環境定義の dcrapautoconnect オペランドに Y を指定し，rpcCall メソッドを呼び出します。

このメソッドを呼び出した場合，TP1/Client/J 環境定義の dccltconnectinf オペランドに指定した端末識別情報は，再び rpcOpen メソッドを呼び出すまで無視されます。

なお，このメソッドに指定した端末識別情報は，DCCM3 論理端末との常設コネクション確立時に認識されます。このメソッドを複数回呼び出した場合は，DCCM3 論理端末との常設コネクションを確立する直前に指定した端末識別情報が有効となります。

また，ネームサービスを使用した RPC，またはスケジューラダイレクト機能を使用した RPC の場合は，このメソッドで指定した端末識別情報は無視されます。

パラメタ

inf

端末識別情報として，DCCM3 論理端末の論理端末名称を 64 バイト以内の 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，9 バイト目以降に指定し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 64 までの範囲のバイト長を指定します。

戻り値

なし。

例外

ErrInvalidArgsException

引数の指定に誤りがあります。

●acceptNotification

```
public void acceptNotification(
                                byte[] inf,
                                int[] inf_len,
                                int port,
                                int timeout,
                                byte[] hostname,
                                byte[] nodeid)
                        throws  ErrInvalidArgsException,
                                ErrProtoException,
                                ErrIOErrException,
                                ErrSecurityException,
                                ErrInvalidPortException,
                                ErrTimedOutException,
                                ErrNetDownException,
                                ErrInvalidMessageException,
                                ErrAcceptCanceledException,
                                ErrReplyTooBigException,
                                ErrVersionException,
                                ErrSyserrException
```

サーバ側の関数（dc_rpc_cltsend 関数）によって通知されるメッセージを，timeout 引数に指定した時間まで待ち続けます。メッセージを受信した時点で CUP に制御を戻し，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，および通知元サーバのノード識別子を返します。

パラメタ

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。

メソッドが正常終了した場合，サーバからの通知メッセージが格納されます。

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（inf 引数の長さ）を指定します。0 から DCRPC_MAX_MESSAGE_SIZE の範囲で指定します。

ただし，TP1/Client/J 環境定義 dccltrpcmaxmsgsize に 2 以上を指定した場合，指定できる値の最大値は，DCRPC_MAX_MESSAGE_SIZE の値（1 メガバイト）ではなく，TP1/Client/J 環境定義 dccltrpcmaxmsgsize に指定した値になります。

メソッドが正常終了した場合，サーバからの通知メッセージ長が格納されます。

port

サーバからの通知メッセージを受信するポート番号を，5001 から 65535 の範囲で指定します。

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，port 引数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，port 引数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，ご利用の OS の関連ドキュメントを参照してください。

timeout

タイムアウト値を，0 から 65535（単位：秒）の範囲で指定します。0 を指定した場合，無限に待ち続けます。

hostname

通知してきたサーバのホスト名を格納する領域を指定します。256 バイト以上を指定してください。

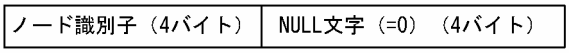
メソッドが正常終了した場合，通知元サーバのホスト名が格納されます。TP1/Client/J は，java.net.InetAddress クラスの getHostName メソッドで通知元サーバの IP アドレスからホスト名を取得し，プラットフォームのデフォルトの文字セットを使用して，取得したホスト名をバイト配列に変換します。この結果が，hostname に格納されます。通知元サーバの IP アドレスからホスト名への変換に失敗した場合，10 進ドット記法（例：10.209.15.124）の IP アドレスを hostname に格納します。

null を指定した場合，通知してきたサーバのホスト名を格納しません。

nodeid

通知してきたサーバのノード識別子を格納する領域を指定します。8 バイト以上を指定してください。

メソッドが正常終了した場合，通知元サーバのノード識別子が格納されます。ノード識別子のフォーマットは，次のとおりです。



戻り値

なし。

例外

ErrInvalidArgsException

メソッドの引数の指定に誤りがあります。

ErrProtoException

rpcOpen メソッドが実行されていません。

ErrIOErrException

何らかの入出力例外が発生しました。

ErrSecurityException

セキュリティ例外が発生しました。

ErrInvalidPortException

port 引数で指定したポート番号は使用中です。

ErrClientTimedOutException

TP1/Client/J でタイムアウトが発生しました。

ErrNetDownAtClientException

TP1/Server と CUP の間でネットワーク障害が発生しました。

ErrInvalidMessageException

不正なメッセージを受信しました。

ErrAcceptCanceledException

一方通知受信待ち状態が cancelNotification メソッドで解除されました。このとき，inf 引数，inf_len 引数，および hostname 引数にはすでに値が設定されています。nodeid 引数には，先頭から 8 バイト目まで 0 で初期化された値が設定されます。

ErrReplyTooBigException

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を切り捨てました。このとき，hostname 引数および nodeid 引数には，すでに値を設定されています。

ErrVersionException

通知元サーバのバージョンが不正です。

ErrSyserrException

システムエラーが発生しました。

●cancelNotification

```
public void cancelNotification(
                                byte[] inf,
                                int inf_len,
                                String hostname,
                                int port)
                        throws  ErrInvalidArgsException,
                                ErrProtoException,
                                ErrIOErrException,
                                ErrInvalidPortException,
                                ErrTimedOutException,
                                ErrNetDownException,
                                ErrHostUndefException,
                                ErrSyserrException
```

サーバからの一方通知受信待ち状態（acceptNotification メソッド，または acceptNotificationChained メソッドの発行）を解除します。解除するときに，inf 引数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。

パラメタ

inf

CUP に通知するメッセージを指定します。

inf_len

CUP に通知するメッセージ長（inf 引数の長さ）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。

ただし，TP1/Client/J 環境定義 dccltrpcmaxmsgsize に 2 以上を指定した場合，指定できる値の最大値は DCRPC_MAX_MESSAGE_SIZE の値（1 メガバイト）ではなく，TP1/Client/J 環境定義 dccltrpcmaxmsgsize に指定した値になります。

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，10 進ドット記法（例：10.209.15.124）の IP アドレスを指定することもできます。

port

一方通知受信待ち状態の CUP のポート番号を，5001 から 65535 の範囲で指定します

戻り値

なし。

例外

ErrInvalidArgsException

メソッドの引数の指定に誤りがあります。

ErrProtoException

rpcOpen メソッドが実行されていません。

ErrIOErrException

何らかの入出力例外が発生しました。

ErrInvalidPortException

port 引数で指定したポート番号が不正です。

ErrClientTimedOutException

TP1/Client/J でタイムアウトが発生しました。

ErrNetDownAtClientException

CUP 間でネットワーク障害が発生しました。

ErrHostUndefException

hostname 引数で指定したホスト名が不正です。

ErrSyserrException

システムエラーが発生しました。

●openNotification

```
public void openNotification(int port)
                     throws  ErrInvalidArgsException,
                             ErrProtoException,
                             ErrInvalidPortException,
                             ErrNetDownException,
                             ErrSyserrException
```

一方通知連続受信機能を使用するための環境を作成します。このメソッドは，closeNotification メソッドと対で発行します。このメソッドが正常終了した場合は，必ず closeNotification メソッドを発行してください。

パラメタ

port

サーバからの通知メッセージを受信するポート番号を，5001 から 65535 の範囲で指定します。

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，port 引数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，port 引数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，ご利用の OS の関連ドキュメントを参照してください。

戻り値

なし。

例外

ErrInvalidArgsException

メソッドの引数の指定に誤りがあります。

ErrProtoException

rpcOpen メソッドが実行されていない，または openNotification メソッドがすでに実行されています。

ErrInvalidPortException

port 引数で指定したポート番号は，すでに使用されています。

ErrNetDownAtClientException

TP1/Server と CUP の間でネットワーク障害が発生しました。

ErrSyserrException

システムエラーが発生しました。

●acceptNotificationChained

```
public void acceptNotificationChained(
                                      byte[] inf,
                                      int[] inf_len,
                                      int timeout,
                                      byte[] hostname,
                                      byte[] nodeid)
                               throws ErrInvalidArgsException,
                                      ErrProtoException,
                                      ErrIOErrException,
                                      ErrSecurityException,
                                      ErrTimedOutException,
                                      ErrNetDownException,
                                      ErrInvalidMessageException,
                                      ErrAcceptCanceledException,
```

```
                                    ErrReplyTooBigException,
                                    ErrVersionException,
                                    ErrSyserrException
```

サーバ側の関数（dc_rpc_cltsend 関数）によって通知されるメッセージを，timeout 引数で指定した時間まで待ち続けます。メッセージを受信した時点で CUP に制御を戻し，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，および通知元サーバのノード識別子を返します。このメソッドを発行するときは，あらかじめ openNotification メソッドを発行しておく必要があります。このメソッドは，openNotification メソッドから closeNotification メソッドまでの間で発行できます。

パラメタ

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。

メソッドが正常終了した場合，サーバからの通知メッセージが格納されます。

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（inf 引数の長さ）を指定します。0 から DCRPC_MAX_MESSAGE_SIZE の範囲で指定します。

ただし，TP1/Client/J 環境定義 dccltrpcmaxmsgsize に 2 以上を指定した場合，指定できる値の最大値は DCRPC_MAX_MESSAGE_SIZE の値（1 メガバイト）ではなく，TP1/Client/J 環境定義 dccltrpcmaxmsgsize に指定した値になります。

メソッドが正常終了した場合，サーバからの通知メッセージ長が格納されます。

timeout

タイムアウト値を，0 から 65535（単位：秒）の範囲で指定します。0 を指定した場合，無限に待ち続けます。

hostname

通知してきたサーバのホスト名を格納する領域を指定します。256 バイト以上を指定してください。

メソッドが正常終了した場合，通知元サーバのホスト名が格納されます。

TP1/Client/J は，java.net.InetAddress クラスの getHostName メソッドで通知元サーバの IP アドレスからホスト名を取得し，プラットフォームのデフォルトの文字セットを使用して，取得したホスト名をバイト配列に変換します。この結果が，hostname に格納されます。通知元サーバの IP アドレスからホスト名への変換に失敗した場合，10 進ドット記法（例：10.209.15.124)の IP アドレスを hostname に格納します。

null を指定した場合，通知してきたサーバのホスト名を格納しません。

nodeid

通知してきたサーバのノード識別子を格納する領域を指定します。8 バイト以上を指定してください。

メソッドが正常終了した場合，通知元サーバのノード識別子が格納されます。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（=0）（4バイト）

戻り値

なし。

例外

ErrInvalidArgsException

メソッドの引数の指定に誤りがあります。

ErrProtoException

openNotification メソッドが実行されていません。

ErrIOErrException

何らかの入出力例外が発生しました。

ErrSecurityException

セキュリティ例外が発生しました。

ErrClientTimedOutException

TP1/Client/J でタイムアウトが発生しました。

ErrNetDownAtClientException

TP1/Server と CUP の間でネットワーク障害が発生しました。

ErrInvalidMessageException

不正なメッセージを受信しました。

ErrAcceptCanceledException

一方通知受信待ち状態が cancelNotification メソッドによって解除されました。このとき，inf 引数，inf_len 引数，および hostname 引数には，すでに値が設定されています。nodeid 引数は，先頭から 8 バイト目まで 0 で初期化された値が設定されます。

ErrReplyTooBigException

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。このとき，hostname 引数および nodeid 引数にはすでに値が設定されています。

ErrVersionException

通知元サーバのバージョンが不正です。

ErrSyserrException

システムエラーが発生しました。

●closeNotification

```
public void closeNotification()
                    throws  ErrNetDownException,
                            ErrSyserrException
```

一方通知連続受信機能を使用するための環境を削除します。このメソッドは，openNotification メソッドと対で発行します。openNotification メソッドが正常終了した場合は，必ずこのメソッドを発行してください。

パラメタ

なし。

戻り値

なし。

例外

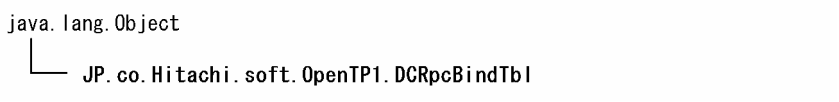
ErrNetDownAtClientException

TP1/Server と CUP の間でネットワーク障害が発生しました。

ErrSyserrException

システムエラーが発生しました。

クラス DCRpcBindTbl




```
public class DCRpcBindTbl
extends java.lang.Object
```

通信先情報を管理するクラスです。rpcCallTo メソッドで使用します。

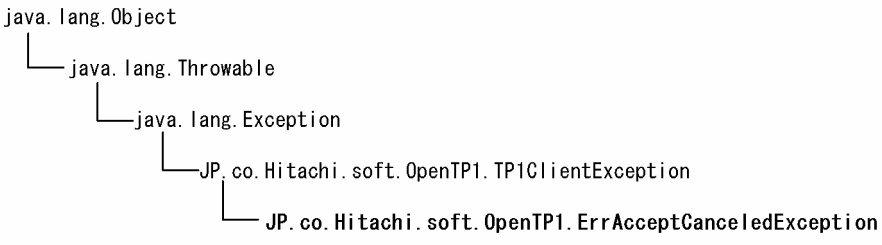
コンストラクタ

●DCRpcBindTbl

```
public DCRpcBindTbl()
```

通信先情報を管理するクラスのインスタンスを作成します。

クラス ErrAcceptCanceledException




```
public class ErrAcceptCanceledException
extends TP1ClientException
```

TP1/Client/J が返す例外のクラスです。次の事象を表します。

一方通知受信待ち状態が解除されました。

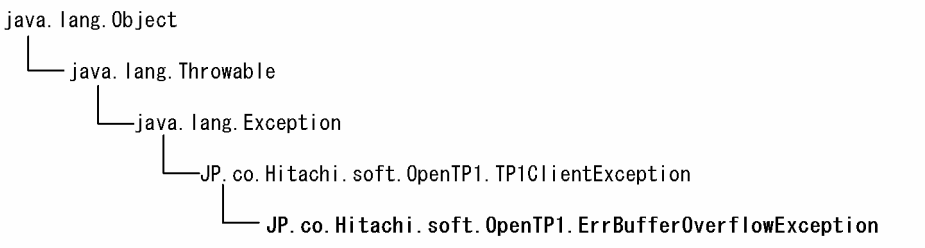
コンストラクタ

●ErrAcceptCanceledException

```
public ErrAcceptCanceledException()
```

詳細メッセージなしで ErrAcceptCanceledException クラスのインスタンスを作成します。

クラス ErrBufferOverflowException




```
public class ErrBufferOverflowException
extends TP1ClientException
```

TP1/Client/J が返す例外のクラスです。次の事象を表します。

cltAssemReceive メソッドに指定した受信バッファの長さを超えるメッセージを受信したため，受信バッファオーバーフローが発生しました。

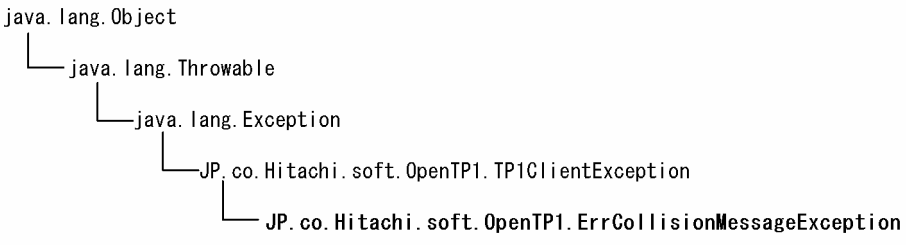
コンストラクタ

●ErrBufferOverflowException

```
public ErrBufferOverflowException()
```

詳細メッセージなしで ErrBufferOverflowException クラスのインスタンスを作成します。

クラス ErrCollisionMessageException




```
public class ErrCollisionMessageException
extends TP1ClientException
```

TP1/Client/J が返す例外のクラスです。次の事象を表します。

送受信メッセージの衝突が発生しました。

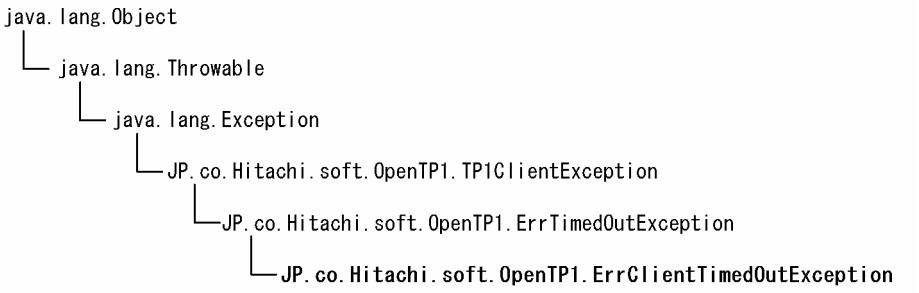
コンストラクタ

●ErrCollisionMessageException

```
public ErrCollisionMessageException()
```

詳細メッセージなしで ErrCollisionMessageException のインスタンスを作成します。

クラス ErrClientTimedOutException




```
public class ErrClientTimedOutException
extends ErrTimedOutException
```

TP1/Client/J が返す例外のクラスです。次の事象を表します。

TP1/Client/J 側でタイムアウトが発生しました。

関連項目

TP1Client，TP1ClientException，ErrTimedOutException

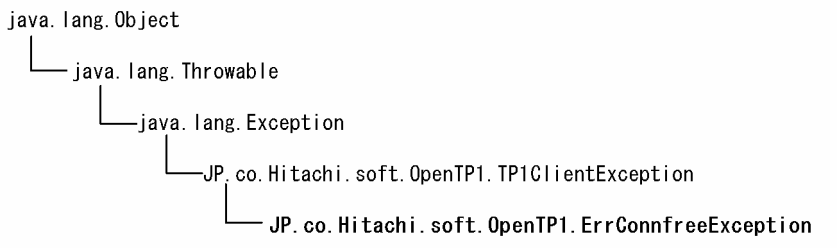
コンストラクタ

●ErrClientTimedOutException

```
public ErrClientTimedOutException()
```

詳細メッセージなしで ErrClientTimedOutException のインスタンスを作成します。

クラス ErrConnfreeException




```
public class ErrConnfreeException
extends TP1ClientException
```

TP1/Client/J が返す例外のクラスです。次の事象を表します。

rpcCall メソッドがこの例外を返した場合

rap サーバとの常設コネクションが切断されました。または，TP1/Web との擬似セションが切断されました。

trnBegin メソッド，trnChainedCommit メソッド，trnChainedRollback メソッド，trnUnchainedCommit メソッド，または trnUnchainedRollback メソッドがこの例外を返した場合

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。

cltReceive メソッドがこの例外を返した場合

MHP からコネクションが解放されました。

cltAssemSend メソッドまたは cltAssemReceive メソッドがこの例外を返した場合

相手システムからコネクションが解放されました。

関連項目

TP1Client，TP1ClientException

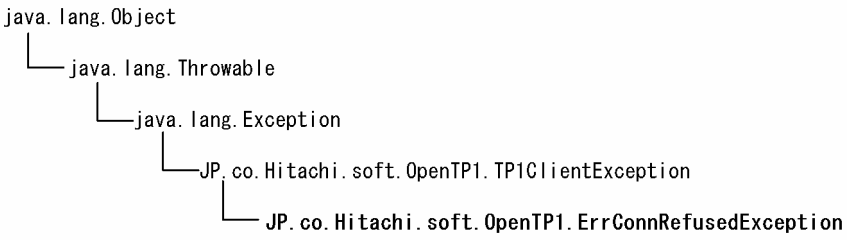
コンストラクタ

●ErrConnfreeException

```
public ErrConnfreeException()
```

詳細メッセージなしで ErrConnfreeException のインスタンスを作成します。

クラス ErrConnRefusedException




```
public class ErrConnRefusedException
extends Exception
```

TP1/Client/J が返す例外のクラスです。次の事象を表します。

リモートなアドレスおよびポートに対するソケットの接続の試行中にエラーが発生したことを通知します。

一般的には，接続がリモートで拒否された（リモートのアドレスとポートで待機中のプロセスがない）ことが原因です。

関連項目

TP1Client，TP1ClientException

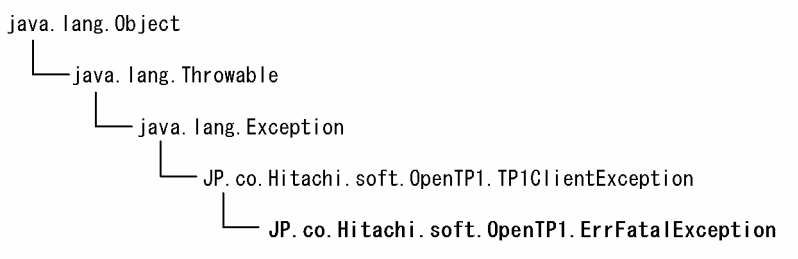
コンストラクタ

●ErrConnRefusedException

```
public ErrConnRefusedException()
```

詳細メッセージなしで ErrConnRefusedException のインスタンスを作成します。

クラス ErrFatalException




```
public class ErrFatalException
extends TP1ClientException
```

TP1/Client/J が返す例外のクラスです。次の事象を表します。

- TP1/Client/J 環境定義の指定に誤りがあります。
- TP1/Server との通信環境の初期化に失敗しました。
- TP1/Client/J 環境定義の次のオペランドに指定したポートが他プロセスで使用中です。
  dccltcuprcvport オペランド
  dcrcvport オペランド

関連項目

TP1Client，TP1ClientException

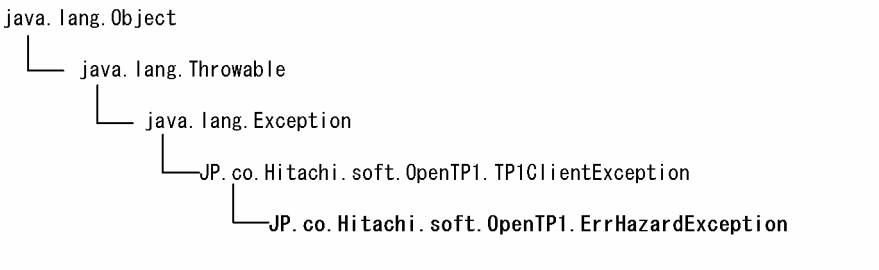
コンストラクタ

●ErrFatalException

```
public ErrFatalException()
```

詳細メッセージなしで ErrFatalException のインスタンスを作成します。

クラス ErrHazardException




```
public class ErrHazardException
extends TP1ClientException
```

TP1/Client/J が返す例外のクラスです。次の事象を表します。

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わかりません。この例外が返される原因になったUAP，リソースマネジャ，およびグローバルトランザクションの同期点の結果は，TP1/Server のメッセージログファイルを参照してください。この例外が返されたあとも，このプロセスはトランザクション下にあり，グローバルトランザクションの範囲内です。

関連項目

TP1Client，TP1ClientException

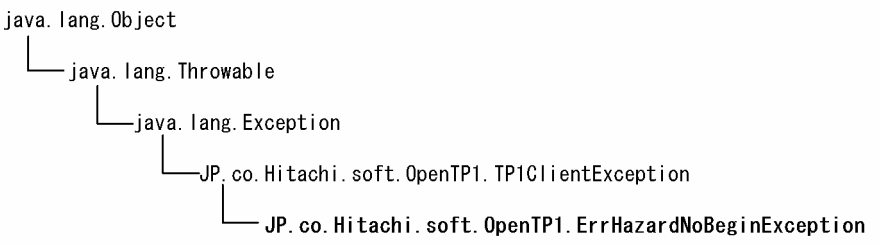
コンストラクタ

●ErrHazardException

```
public ErrHazardException()
```

詳細メッセージなしで ErrHazardException のインスタンスを作成します。

クラス ErrHazardNoBeginException




```
public class ErrHazardNoBeginException
extends TP1ClientException
```

TP1/Client/J が返す例外のクラスです。次の事象を表します。

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わかりません。この例外が返される原因になった UAP，リソースマネジャ，およびグローバルトランザクションの同期点の結果は，TP1/Server のメッセージログファイルの内容を参照してください。新しいトランザクションは開始できませんでした。この例外が返されたあと，このプロセスはトランザクション下にありません。

関連項目

TP1Client，TP1ClientException

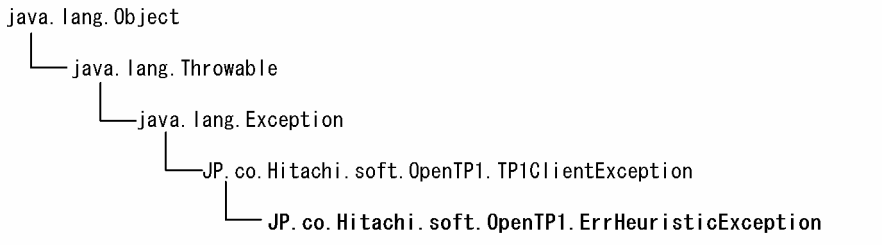
コンストラクタ

●ErrHazardNoBeginException

```
public ErrHazardNoBeginException()
```

詳細メッセージなしで ErrHazardNoBeginException のインスタンスを作成します。

クラス ErrHeuristicException




```
public class ErrHeuristicException
extends TP1ClientException
```

TP1/Client/J が返す例外のクラスです。次の事象を表します。

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，あるトランザクションブランチはコミット，あるトランザクションブランチはロールバックしました。この例外は，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い場合に返されます。この例外が返される原因になった UAP，リソースマネジャ，およびグローバルトランザクションの同期点の結果は，TP1/Server のメッセージログファイルを参照してください。この例外が返されたあとも，このプロセスはトランザクション下にあり，グローバルトランザクションの範囲内です。

関連項目

TP1Client，TP1ClientException

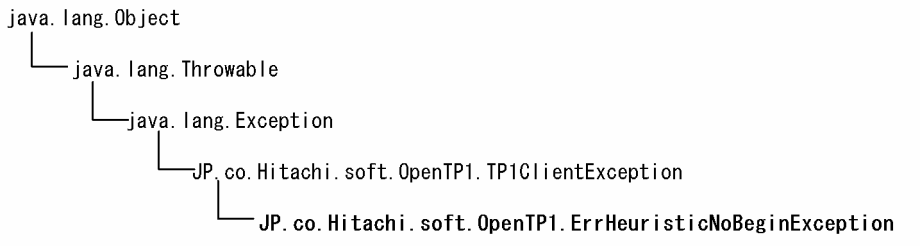
コンストラクタ

●ErrHeuristicException

```
public ErrHeuristicException()
```

詳細メッセージなしで ErrHeuristicException のインスタンスを作成します。

クラス ErrHeuristicNoBeginException




```
public class ErrHeuristicNoBeginException
extends TP1ClientException
```

TP1/Client/J が返す例外のクラスです。次の事象を表します。

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，あるトランザクションブランチはコミット，あるトランザクションブランチはロールバックしました。この例外は，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い場合に返されます。この例外が返される原因になった UAP，リソースマネジャ，およびグローバルトランザクションの同期点の結果は，TP1/Server のメッセージログファイルの内容を参照してください。新しいトランザクションは開始できませんでした。この例外が返されたあと，このプロセスはトランザクション下にありません。

関連項目

TP1Client，TP1ClientException

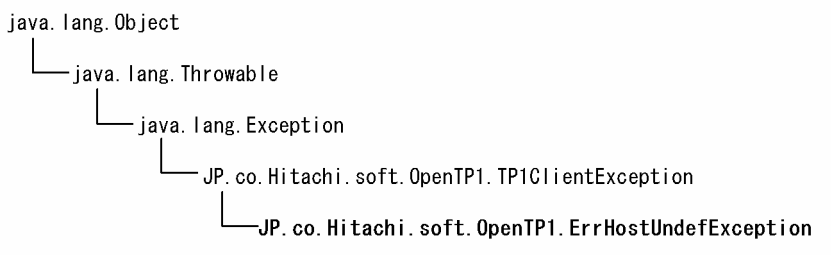
コンストラクタ

●ErrHeuristicNoBeginException

```
public ErrHeuristicNoBeginException()
```

詳細メッセージなしで ErrHeuristicNoBeginException のインスタンスを作成します。

クラス ErrHostUndefException




```
public class ErrHostUndefException
extends TP1ClientException
```

TP1/Client/J が返す例外のクラスです。次の事象を表します。

setDchost メソッドがこの例外を返した場合

host 引数の指定に誤りがあります。

openConnection メソッドがこの例外を返した場合

- rap リスナーのホスト名が TP1/Client/J 環境定義の dchost オペランドに指定されていません。
- url 引数または TP1/Client/J 環境定義の dcweburl オペランドに指定された URL（プロトコル，WWW サーバ，プロンプターの CGI 名称，TP1/Web のサービス名などの情報）に誤りがあります。
- host 引数の指定に誤りがあります。

rpcCall メソッドがこの例外を返した場合

- 通信先となる TP1/Server のホスト名が TP1/Client/J 環境定義の dchost オペランドに指定されていないか，または指定に誤りがあります。
- TP1/Client/J 環境定義の dcweburl オペランドに指定された URL（プロトコル，WWW サーバ，プロンプターの CGI 名称，TP1/Web のサービス名などの情報）に誤りがあります。

trnBegin メソッドがこの例外を返した場合

通信先となる TP1/Server のホスト名が TP1/Client/J 環境定義の dchost オペランドに指定されていないか，または指定に誤りがあります。

cltSend メソッド，または cltAssemSend メソッドがこの例外を返した場合

hostname 引数の指定に誤りがあるか，または，hostname 引数および TP1/Client/J 環境定義の dcsndhost オペランドの両方にホスト名が指定されていません。

関連項目

TP1Client，TP1ClientException

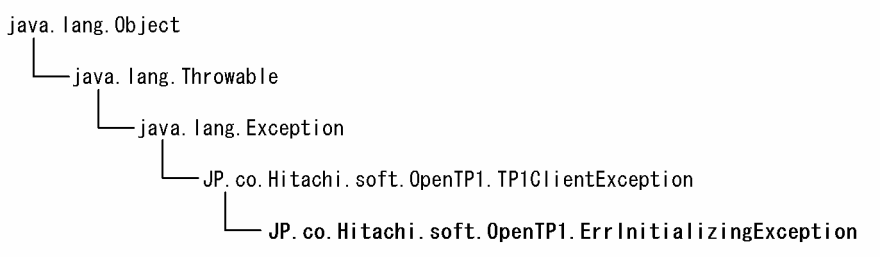
コンストラクタ

●ErrHostUndefException

```
public ErrHostUndefException()
```

詳細メッセージなしで ErrHostUndefException のインスタンスを作成します。

クラス ErrInitializingException




```
public class ErrInitializingException
extends TP1ClientException
```

TP1/Client/J が返す例外のクラスです。次の事象を表します。

サービス要求したノードにある TP1/Server は開始処理中です。

関連項目

TP1Client, TP1ClientException

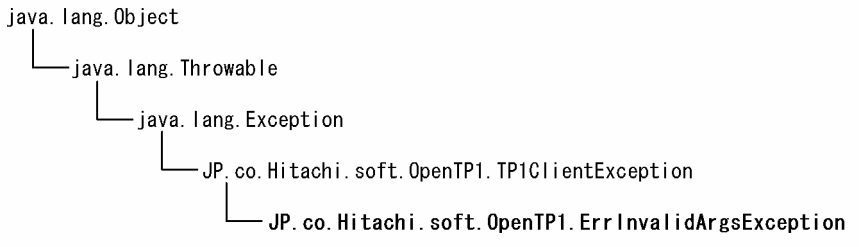
コンストラクタ

●ErrInitializingException

```
public ErrInitializingException()
```

詳細メッセージなしで ErrInitializingException のインスタンスを作成します。

クラス ErrInvalidArgsException




```
public class ErrInvalidArgsException
extends TP1ClientException
```

TP1/Client/J が返す例外のクラスです。次の事象を表します。

メソッドの引数の指定に誤りがあります。

関連項目

TP1Client，TP1ClientException

コンストラクタ

●ErrInvalidArgsException

```
public ErrInvalidArgsException()
```

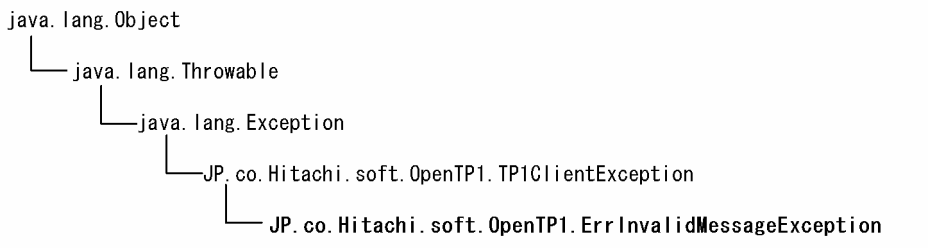
詳細メッセージなしで ErrInvalidArgsException のインスタンスを作成します。

●ErrInvalidArgsException

```
public ErrInvalidArgsException(String msg)
```

詳細メッセージ付きで TP1ClientException のインスタンスを作成します。詳細メッセージは getMessage メソッドで取り出せます。

クラス ErrInvalidMessageException




```
public class ErrInvalidMessageException
extends TP1ClientException
```

TP1/Client/J が返す例外のクラスです。次の事象を表します。

cltAssemReceive メソッドがこの例外を返した場合

受信メッセージのメッセージ長が 0x00000005～0x7FFFFFFF にありません。

セグメント情報が不正です。

cltAssemSend メソッドがこの例外を返した場合

応答受信時に受信したメッセージのメッセージ長が 0x0000000B～0x7FFFFFFF か，または，セグメント情報が不正です。

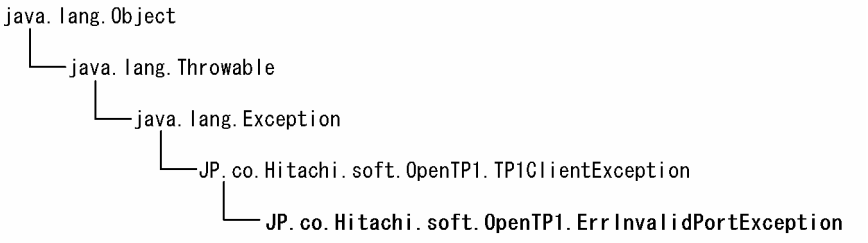
コンストラクタ

●ErrInvalidMessageException

```
public ErrInvalidMessageException()
```

詳細メッセージなしで ErrInvalidMessageException クラスのインスタンスを作成します。

クラス ErrInvalidPortException




```
public class ErrInvalidPortException
extends TP1ClientException
```

TP1/Client/J が返す例外のクラスです。次の事象を表します。

setDchost メソッド，openConnection メソッド，または trnBegin メソッドがこの例外を返した場合

port 引数の指定に誤りがあります。

rpcCall メソッドがこの例外を返した場合

- リモート API 機能を使用した RPC を行う場合，TP1/Client/J 環境定義の dcrapport オペランドが指定されていません。
- スケジューラダイレクト機能を使用した RPC を行う場合，TP1/Client/J 環境定義の dcscdport オペランドが指定されていません。

cltSend メソッドまたは cltAssemSend メソッドがこの例外を返した場合

portnum 引数の指定に誤りがあります。

cltReceive メソッドまたは cltAssemReceive メソッドがこの例外を返した場合

TP1/Client/J 環境定義の dcsockopenatrcv オペランドに Y を指定し，dcrcvport オペランドに指定したポートがすでに使用中の場合に発生します。dcsockopenatrcv オペランドに N を指定した場合は，rpcOpen メソッドが ErrFatalException を返します。

関連項目

TP1Client，TP1ClientException

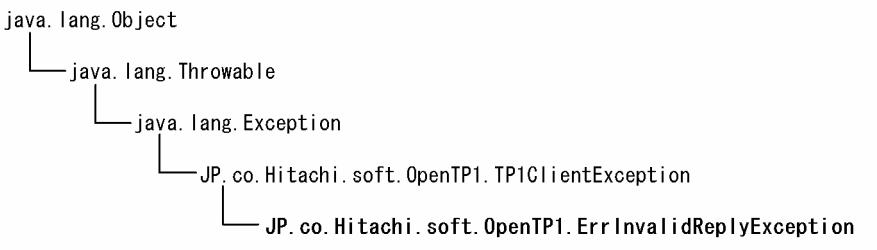
コンストラクタ

●ErrInvalidPortException

```
public ErrInvalidPortException()
```

詳細メッセージなしで ErrInvalidPortException のインスタンスを作成します。

クラス ErrInvalidReplyException




```
public class ErrInvalidReplyException
extends TP1ClientException
```

TP1/Client/J が返す例外のクラスです。次の事象を表します。

サービス関数が返した応答の長さが 1 から DCRPC_MAX_MESSAGE_SIZE※で指定した値までの範囲にありません。

注※

dccltrpcmaxmsgsize オペランドを使用した場合，DCRPC_MAX_MESSAGE_SIZE の値ではなく，dccltrpcmaxmsgsize オペランドに指定した値になります。

関連項目

TP1Client，TP1ClientException

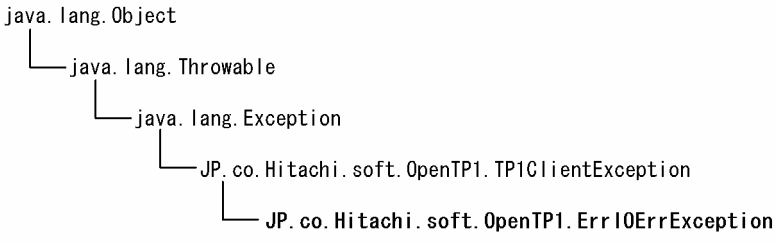
コンストラクタ

●ErrInvalidReplyException

```
public ErrInvalidReplyException()
```

詳細メッセージなしで ErrInvalidReplyException のインスタンスを作成します。

クラス ErrIOErrException




```
public class ErrIOErrException
extends TP1ClientException
```

TP1/Client/J が返す例外のクラスです。次の事象を表します。

何らかの入出力例外が発生しました。詳細は各メソッドの例外および注意事項を参照してください。

関連項目

TP1Client, TP1ClientException

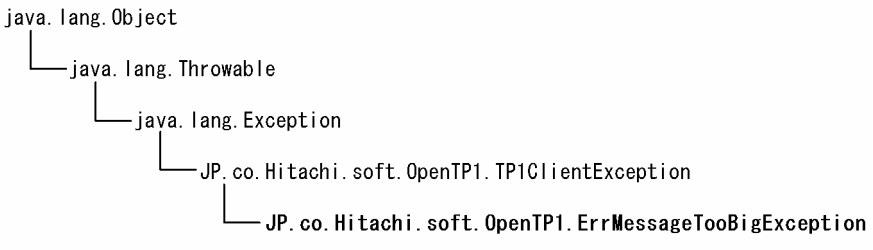
コンストラクタ

●ErrIOErrException

```
public ErrIOErrException()
```

詳細メッセージなしで ErrIOErrException のインスタンスを作成します。

クラス ErrMessageTooBigException




```
public class ErrMessageTooBigException
extends TP1ClientException
```

TP1/Client/J が返す例外のクラスです。次の事象を表します。

rpcCall メソッドの in_len 引数に指定した入力パラメタ長が最大値を超えています。

関連項目

TP1Client，TP1ClientException

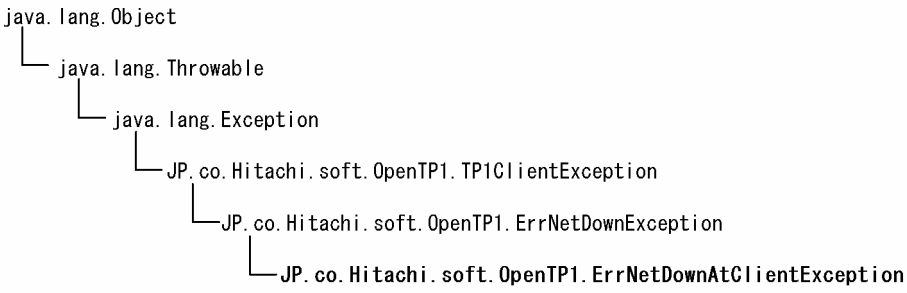
コンストラクタ

●ErrMessageTooBigException

```
public ErrMessageTooBigException()
```

詳細メッセージなしで ErrMessageTooBigException のインスタンスを作成します。

クラス ErrNetDownAtClientException




```
public class ErrNetDownAtClientException
extends ErrNetDownException
```

TP1/Client/J が返す例外のクラスです。次の事象を表します。

TP1/Server と CUP の間でネットワーク障害が発生しました。

関連項目

TP1Client，TP1ClientException，ErrNetDownException

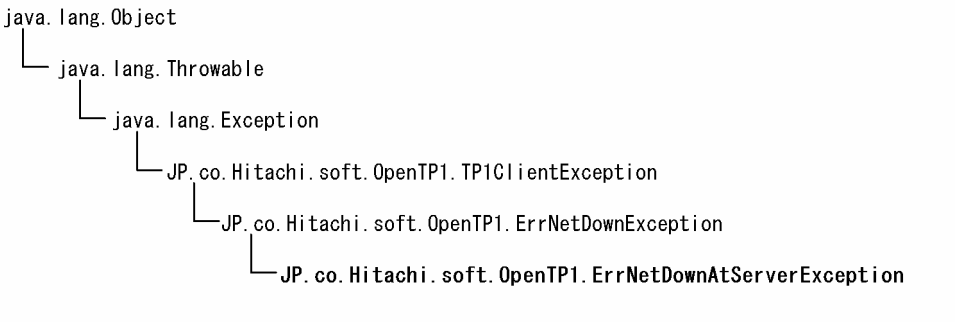
コンストラクタ

●ErrNetDownAtClientException

```
public ErrNetDownAtClientException()
```

詳細メッセージなしで ErrNetDownAtClientException のインスタンスを作成します。

クラス ErrNetDownAtServerException




```
public class ErrNetDownAtServerException
extends ErrNetDownException
```

TP1/Client/J が返す例外のクラスです。次の事象を表します。

TP1/Server と SPP の間でネットワーク障害が発生しました。

関連項目

TP1Client，TP1ClientException，ErrNetDownException

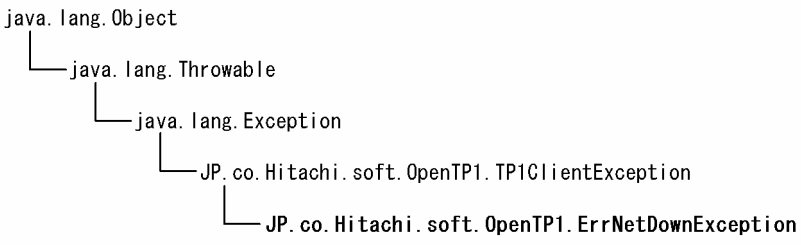
コンストラクタ

●ErrNetDownAtServerException

```
public ErrNetDownAtServerException()
```

詳細メッセージなしで ErrNetDownAtServerException のインスタンスを作成します。

クラス ErrNetDownException



直系の既知のサブクラス

```
ErrNetDownAtClientException
ErrNetDownAtServerException
```

```
public class ErrNetDownException
extends TP1ClientException
```

TP1/Client/J が返す例外のクラスです。次の事象を表します。

ネットワーク障害が発生しました。または，通信先の TP1/Server が稼働していません。

関連項目

TP1Client，TP1ClientException

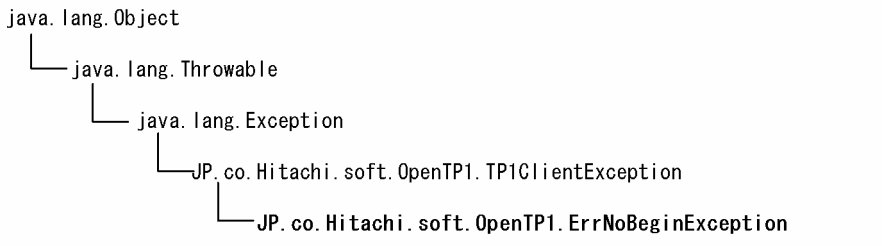
コンストラクタ

●ErrNetDownException

```
public ErrNetDownException()
```

詳細メッセージなしで ErrNetDownException のインスタンスを作成します。

クラス ErrNoBeginException




```
public class ErrNoBeginException
extends TP1ClientException
```

TP1/Client/J が返す例外のクラスです。次の事象を表します。

コミットまたはロールバック処理は正常に終了しましたが，新しいトランザクションは開始できませんでした。この例外が返されたあと，このプロセスはトランザクション下にありません。

関連項目

TP1Client，TP1ClientException

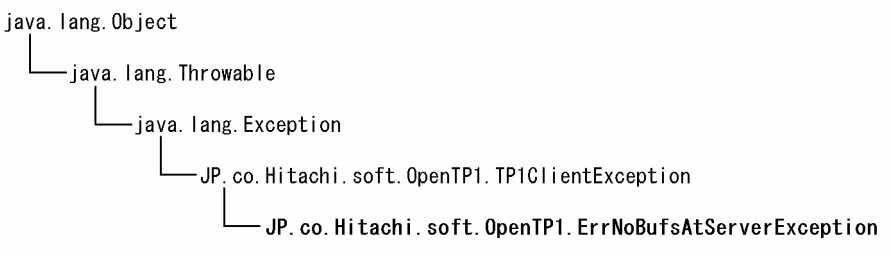
コンストラクタ

●ErrNoBeginException

```
public ErrNoBeginException()
```

詳細メッセージなしで ErrNoBeginException のインスタンスを作成します。

クラス ErrNoBufsAtServerException




```
public class ErrNoBufsAtServerException
extends TP1ClientException
```

TP1/Client/J が返す例外のクラスです。次の事象を表します。

指定したサービスでメモリ不足が発生しました。

関連項目

TP1Client，TP1ClientException

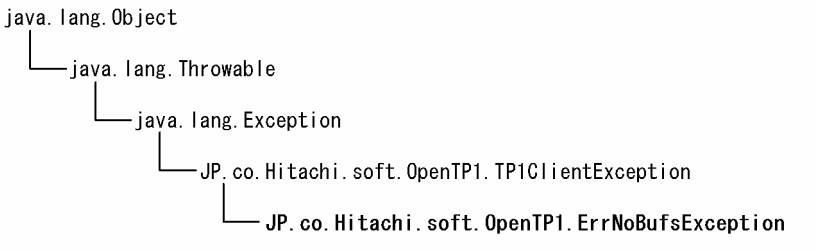
コンストラクタ

●ErrNoBufsAtServerException

```
public ErrNoBufsAtServerException()
```

詳細メッセージなしで ErrNoBufsAtServerException のインスタンスを作成します。

クラス ErrNoBufsException




```
public class ErrNoBufsException
extends TP1ClientException
```

TP1/Client/J が返す例外のクラスです。次の事象を表します。

メモリ不足が発生しました。

関連項目

TP1Client, TP1ClientException

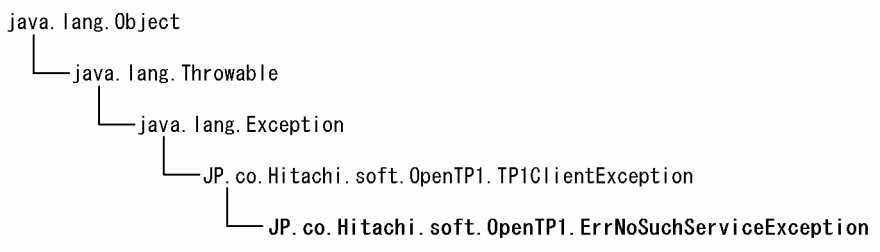
コンストラクタ

●ErrNoBufsException

```
public ErrNoBufsException()
```

詳細メッセージなしで ErrNoBufsException のインスタンスを作成します。

クラス ErrNoSuchServiceException




```
public class ErrNoSuchServiceException
extends TP1ClientException
```

TP1/Client/J が返す例外のクラスです。次の事象を表します。

service 引数に指定したサービス名は定義されていません。

関連項目

TP1Client，TP1ClientException

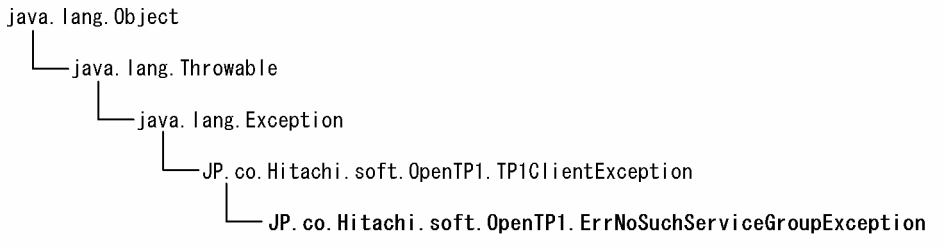
コンストラクタ

●ErrNoSuchServiceException

```
public ErrNoSuchServiceException()
```

詳細メッセージなしで ErrNoSuchServiceException のインスタンスを作成します。

クラス ErrNoSuchServiceGroupException




```
public class ErrNoSuchServiceGroupException
extends TP1ClientException
```

TP1/Client/J が返す例外のクラスです。次の事象を表します。

group 引数に指定したサービスグループ名は定義されていません。

関連項目

TP1Client, TP1ClientException

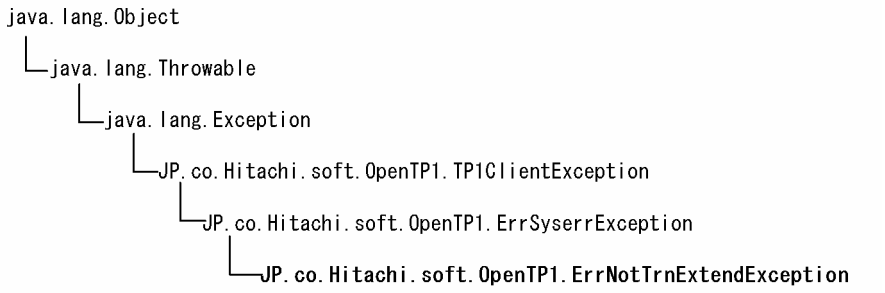
コンストラクタ

●ErrNoSuchServiceGroupException

```
public ErrNoSuchServiceGroupException()
```

詳細メッセージなしで ErrNoSuchServiceGroupException のインスタンスを作成します。

クラス ErrNotTrnExtendException




```
public class ErrNotTrnExtendException
extends ErrSyserrException
```

TP1/Client/J が返す例外のクラスです。次の事象を表します。

トランザクション処理の連鎖型 RPC を使用したあとで，flags 引数に DCRPC_TPNOTRAN を指定したサービスを要求しています。

関連項目

TP1Client，TP1ClientException，ErrSyserrException

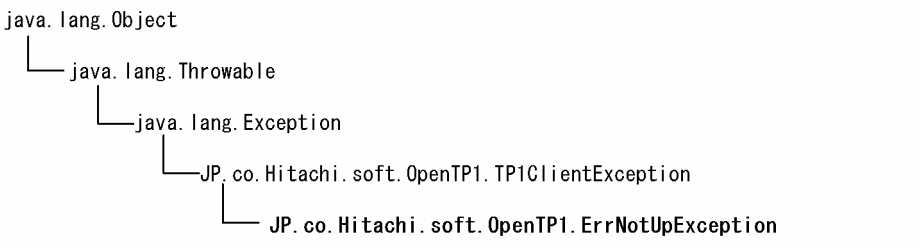
コンストラクタ

●ErrNotTrnExtendException

```
public ErrNotTrnExtendException()
```

詳細メッセージなしで ErrNotTrnExtendException のインスタンスを作成します。

クラス ErrNotUpException




```
public class ErrNotUpException
extends TP1ClientException
```

TP1/Client/J が返す例外のクラスです。次の事象を表します。

指定したサービスが存在するノードの TP1/Server が稼働していません。

関連項目

TP1Client, TP1ClientException

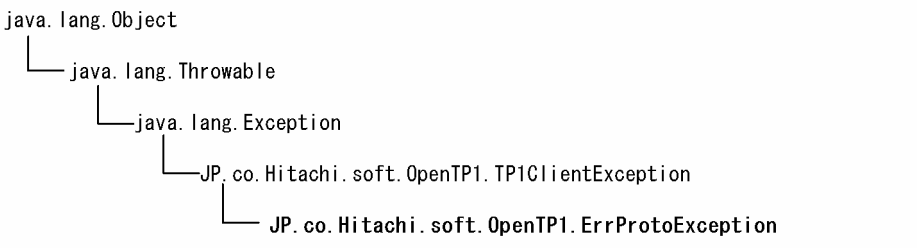
コンストラクタ

●ErrNotUpException

```
public ErrNotUpException()
```

詳細メッセージなしで ErrNotUpException のインスタンスを作成します。

クラス ErrProtoException




```
public class ErrProtoException
extends TP1ClientException
```

TP1/Client/J が返す例外のクラスです。次の事象を表します。

メソッドの発行順序に誤りがあります。

関連項目

TP1Client, TP1ClientException

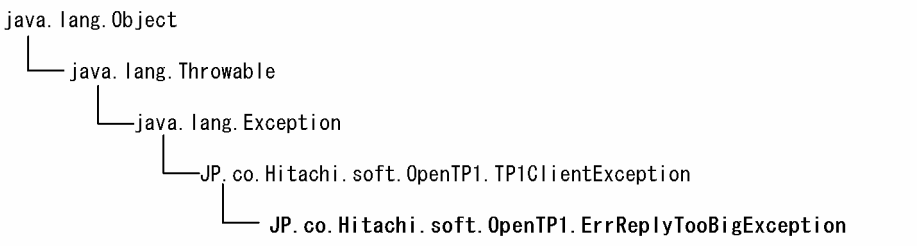
コンストラクタ

●ErrProtoException

```
public ErrProtoException()
```

詳細メッセージなしで ErrProtoException のインスタンスを作成します。

クラス ErrReplyTooBigException




```
public class ErrReplyTooBigException
extends TP1ClientException
```

TP1/Client/J が返す例外のクラスです。次の事象を表します。

サーバから返された応答の長さが，CUP で用意した領域（out_data 引数の指定値）の長さを超えています。

関連項目

TP1Client，TP1ClientException

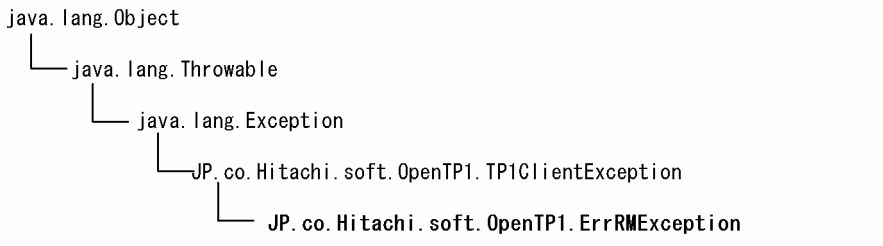
コンストラクタ

●ErrReplyTooBigException

```
public ErrReplyTooBigException()
```

詳細メッセージなしで ErrReplyTooBigException のインスタンスを作成します。

クラス ErrRMException




```
public class ErrRMException
extends TP1ClientException
```

TP1/Client/J が返す例外のクラスです。次の事象を表します。

リソースマネジャでエラーが発生しました。トランザクションは開始できませんでした。

関連項目

TP1Client，TP1ClientException

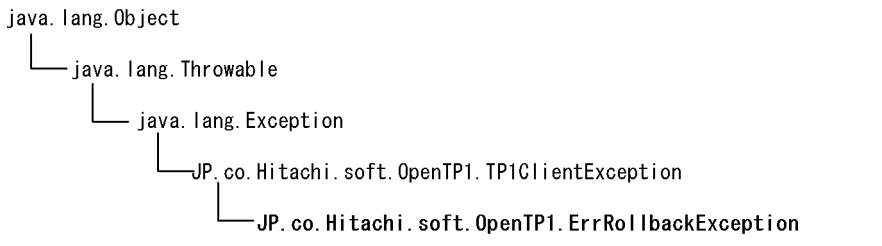
コンストラクタ

●ErrRMException

```
public ErrRMException()
```

詳細メッセージなしで ErrRMException のインスタンスを作成します。

クラス ErrRollbackException




```
public class ErrRollbackException
extends TP1ClientException
```

TP1/Client/J が返す例外のクラスです。次の事象を表します。

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。

trnChainedCommit メソッドでこの例外が返された場合，このプロセスはトランザクションの範囲内です。trnUnchainedCommit メソッドでこの例外が返された場合，このプロセスはトランザクションの範囲外です。

関連項目

TP1Client，TP1ClientException

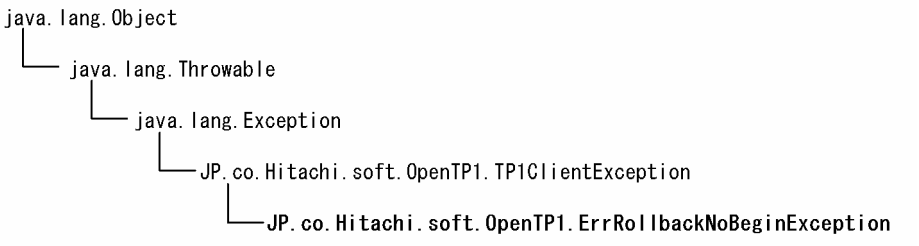
コンストラクタ

●ErrRollbackException

```
public ErrRollbackException()
```

詳細メッセージなしで ErrRollbackException のインスタンスを作成します。

クラス ErrRollbackNoBeginException




```
public class ErrRollbackNoBeginException
extends TP1ClientException
```

TP1/Client/J が返す例外のクラスです。次の事象を表します。

コミットしようとした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。新しいトランザクションは開始できませんでした。この例外が返されたあと，このプロセスはトランザクション下にありません。

関連項目

TP1Client，TP1ClientException

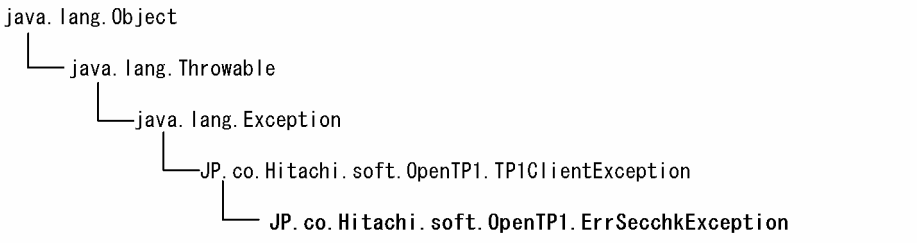
コンストラクタ

●ErrRollbackNoBeginException

```
public ErrRollbackNoBeginException()
```

詳細メッセージなしで ErrRollbackNoBeginException のインスタンスを作成します。

クラス ErrSecchkException




```
public class ErrSecchkException
extends TP1ClientException
```

TP1/Client/J が返す例外のクラスです。次の事象を表します。

サービス要求先の SPP は，OpenTP1 のセキュリティ機能で保護されています。サービス要求した CUP には，SPP へのアクセス権限がありません。

関連項目

TP1Client，TP1ClientException

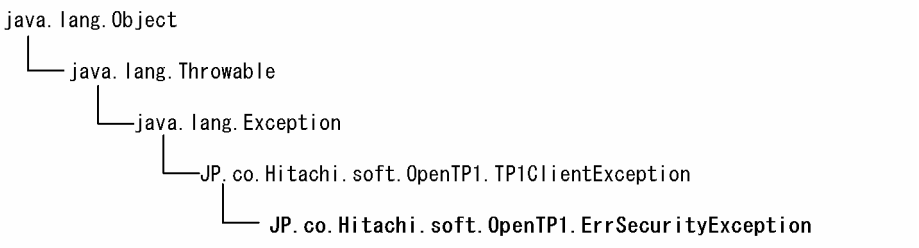
コンストラクタ

●ErrSecchkException

```
public ErrSecchkException()
```

詳細メッセージなしで ErrSecchkException のインスタンスを作成します。

クラス ErrSecurityException




```
public class ErrSecurityException
extends TP1ClientException
```

TP1/Client/J が返す例外のクラスです。次の事象を表します。

セキュリティ例外が発生しました

関連項目

TP1Client, TP1ClientException

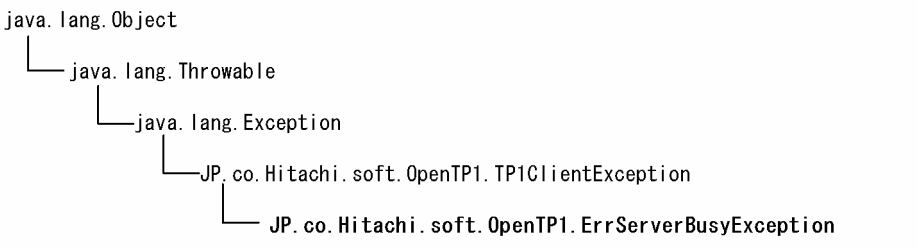
コンストラクタ

●ErrSecurityException

```
public ErrSecurityException()
```

詳細メッセージなしで ErrSecurityException のインスタンスを作成します。

クラス ErrServerBusyException




```
public class ErrServerBusyException
extends TP1ClientException
```

TP1/Client/J が返す例外のクラスです。次の事象を表します。

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。

関連項目

TP1Client，TP1ClientException

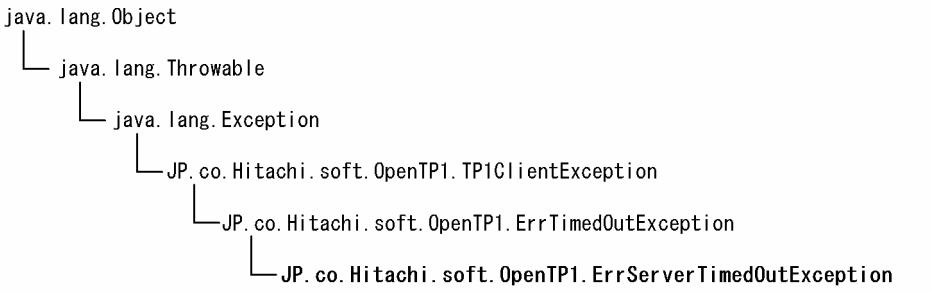
コンストラクタ

●ErrServerBusyException

```
public ErrServerBusyException()
```

詳細メッセージなしで ErrServerBusyException のインスタンスを作成します。

クラス ErrServerTimedOutException




```
public class ErrServerTimedOutException
extends ErrTimedOutException
```

TP1/Client/J が返す例外のクラスです。次の事象を表します。

TP1/Server 側でサービスの実行中にタイムアウトが発生しました。または，サービス要求先 SPP が処理を終了する前に異常終了しました。

関連項目

TP1Client，TP1ClientException，ErrTimedOutException

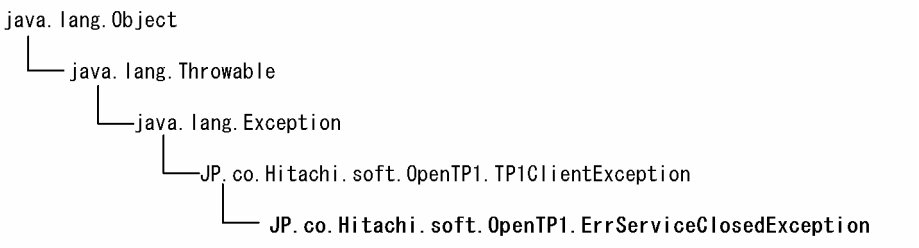
コンストラクタ

●ErrServerTimedOutException

```
public ErrServerTimedOutException()
```

詳細メッセージなしで ErrServerTimedOutException のインスタンスを作成します。

クラス ErrServiceClosedException




```
public class ErrServiceClosedException
extends TP1ClientException
```

TP1/Client/J が返す例外のクラスです。次の事象を表します。

指定され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。

関連項目

TP1Client, TP1ClientException

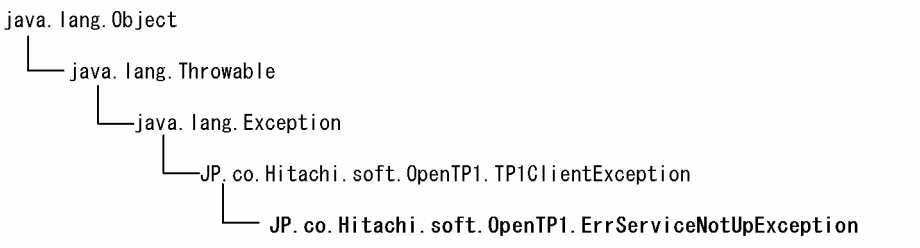
コンストラクタ

●ErrServiceClosedException

```
public ErrServiceClosedException()
```

詳細メッセージなしで ErrServiceClosedException のインスタンスを作成します。

クラス ErrServiceNotUpException




```
public class ErrServiceNotUpException
extends TP1ClientException
```

TP1/Client/J が返す例外のクラスです。次の事象を表します。

サービス要求した SPP は稼働していません。または，サービス要求した SPP が処理を完了する前に異常終了しました。

関連項目

TP1Client，TP1ClientException

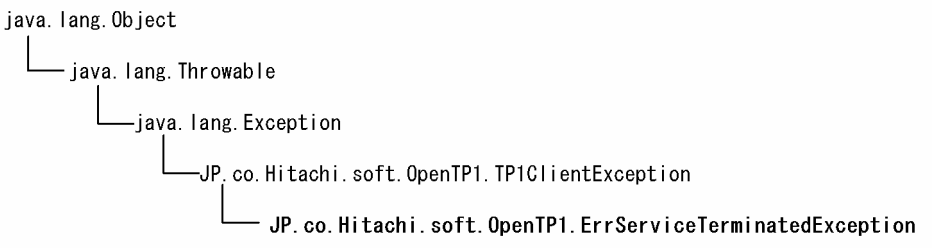
コンストラクタ

●ErrServiceNotUpException

```
public ErrServiceNotUpException()
```

詳細メッセージなしで ErrServiceNotUpException のインスタンスを作成します。

クラス ErrServiceTerminatedException




```
public class ErrServiceTerminatedException
extends TP1ClientException
```

TP1/Client/J が返す例外のクラスです。次の事象を表します。

サービス要求をした SPP が処理を完了する前に異常終了しました。

関連項目

TP1Client，TP1ClientException

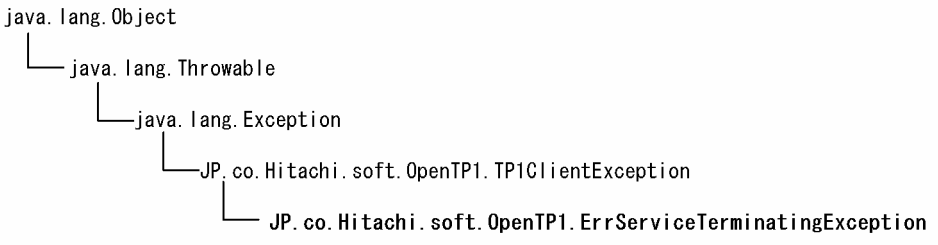
コンストラクタ

●ErrServiceTerminatedException

```
public ErrServiceTerminatedException()
```

詳細メッセージなしで ErrServiceTerminatedException のインスタンスを作成します。

クラス ErrServiceTerminatingException




```
public class ErrServiceTerminatingException
extends TP1ClientException
```

TP1/Client/J が返す例外のクラスです。次の事象を表します。

指定されたサービスは終了処理中です。

関連項目

TP1Client, TP1ClientException

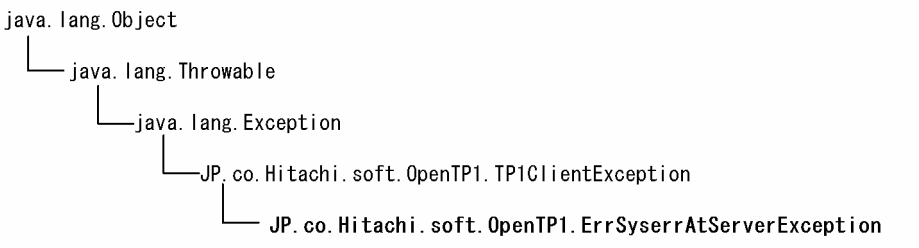
コンストラクタ

●ErrServiceTerminatingException

```
public ErrServiceTerminatingException()
```

詳細メッセージなしで ErrServiceTerminatingException のインスタンスを作成します。

クラス ErrSyserrAtServerException




```
public class ErrSyserrAtServerException
extends TP1ClientException
```

TP1/Client/J が返す例外のクラスです。次の事象を表します。

指定されたサービスでシステムエラーが発生しました。

関連項目

TP1Client, TP1ClientException

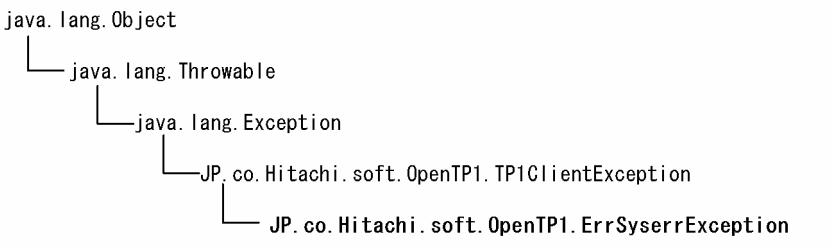
コンストラクタ

●ErrSyserrAtServerException

```
public ErrSyserrAtServerException()
```

詳細メッセージなしで ErrSyserrAtServerException のインスタンスを作成します。

クラス ErrSyserrException



直系の既知のサブクラス

```
ErrNotTrnExtendException
ErrTrnchkExtendException
```

```
public class ErrSyserrException
extends TP1ClientException
```

TP1/Client/J が返す例外のクラスです。次の事象を表します。

システムエラーが発生しました。詳細については，エラートレースまたはメモリトレースを参照してください。

エラートレースおよびメモリトレースの内容については，「2.11.4　エラートレース，メモリトレース」を参照してください。

関連項目

TP1Client，TP1ClientException

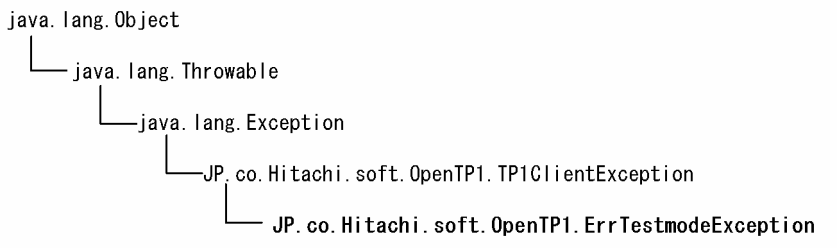
コンストラクタ

●ErrSyserrException

```
public ErrSyserrException()
```

詳細メッセージなしで ErrSyserrException のインスタンスを作成します。

クラス ErrTestmodeException




```
public class ErrTestmodeException
extends TP1ClientException
```

TP1/Client/J が返す例外のクラスです。次の事象を表します。

テストモードの SPP に対してサービス要求を行いました。

関連項目

TP1Client, TP1ClientException

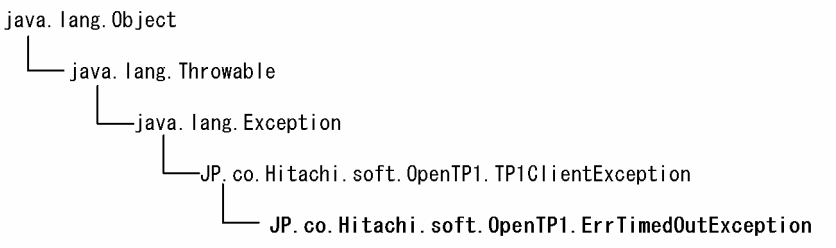
コンストラクタ

●ErrTestmodeException

```
public ErrTestmodeException()
```

詳細メッセージなしで ErrTestmodeException のインスタンスを作成します。

クラス ErrTimedOutException



直系の既知のサブクラス

```
ErrClientTimedOutException
ErrServerTimedOutException
```

```
public class ErrTimedOutException
extends TP1ClientException
```

TP1/Client/J が返す例外のクラスです。次の事象を表します。

openConnection メソッドがこの例外を返した場合

TP1/Web の擬似セション開始中または rap リスナーとのコネクションの確立中に，タイムアウトが発生しました。

closeConnection メソッドがこの例外を返した場合

TP1/Web との擬似セション切断中または rap リスナー，rap サーバとのコネクション切断中に，タイムアウトが発生しました。

rpcCall メソッド，rpcCallTo メソッドがこの例外を返した場合

次の要因が考えられます。

- このメソッドの処理でタイムアウトが発生しました。
- サービス要求先 SPP が処理を完了する前に異常終了しました。
- dcresponsehost オペランドに指定したホスト名または IP アドレスが誤っているため，応答電文の受信ができず時間切れ（タイムアウト）が発生しました。

cltAssemSend メソッドがこの例外を返した場合

応答メッセージの受信時に，タイムアウトが発生しました。

cltAssemReceive メソッドがこの例外を返した場合

メッセージの受信時に，タイムアウトが発生しました。

trnChainedCommit メソッド，trnChainedRollback メソッド，trnUnchainedCommit メソッド，trnUnchainedRollback メソッド，または cltReceive メソッドがこの例外を返した場合

タイムアウトが発生しました。

関連項目

TP1Client，TP1ClientException

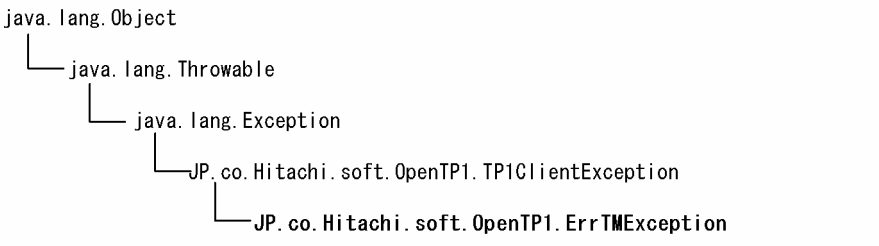
コンストラクタ

●ErrTimedOutException

```
public ErrTimedOutException()
```

詳細メッセージなしで ErrTimedOutException のインスタンスを作成します。

クラス ErrTMException




```
public class ErrTMException
extends TP1ClientException
```

TP1/Client/J が返す例外のクラスです。次の事象を表します。

トランザクションサービスでエラーが発生したため，トランザクションを開始できませんでした。この例外が返されたあと，適当な時間を置いてから再度実行すると，トランザクションを開始できることがあります。

関連項目

TP1Client，TP1ClientException

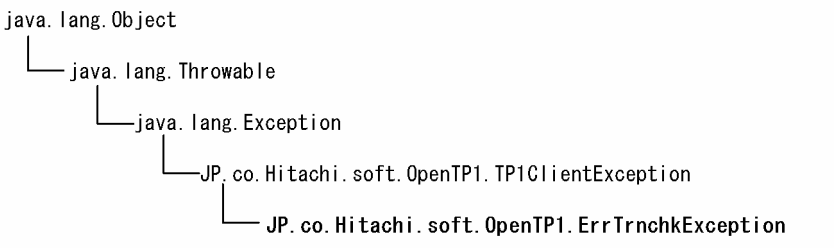
コンストラクタ

●ErrTMException

```
public ErrTMException()
```

詳細メッセージなしで ErrTMException のインスタンスを作成します。

クラス ErrTrnchkException




```
public class ErrTrnchkException
extends TP1ClientException
```

TP1/Client/J が返す例外のクラスです。次の事象を表します。

ノード間負荷バランス機能を使用している環境で，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の TP1/Server のバージョンが，TP1/Client/J のバージョンよりも古いため，ノード間負荷バランス機能を実行できません。

この例外は，ノード間負荷バランス機能を使っている SPP にサービスを要求した場合にだけ返されます。

関連項目

TP1Client，TP1ClientException

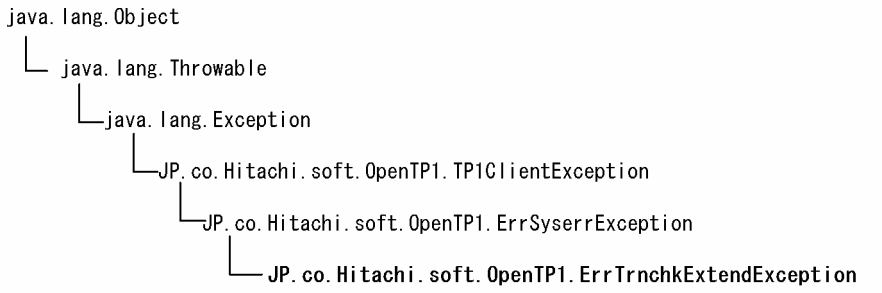
コンストラクタ

●ErrTrnchkException

```
public ErrTrnchkException()
```

詳細メッセージなしで ErrTrnchkException のインスタンスを作成します。

クラス ErrTrnchkExtendException




```
public class ErrTrnchkExtendException
extends ErrSyserrException
```

TP1/Client/J が返す例外のクラスです。次の事象を表します。

次のどれかの要因が考えられます。

- 同時に起動できるトランザクションブランチの数を超えたため，トランザクションブランチを開始できない
- 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超えたため，トランザクションブランチを開始できない
- 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 引数に DCRPC_TPNOTRAN を設定していない

関連項目

TP1Client，TP1ClientException，ErrSyserrException

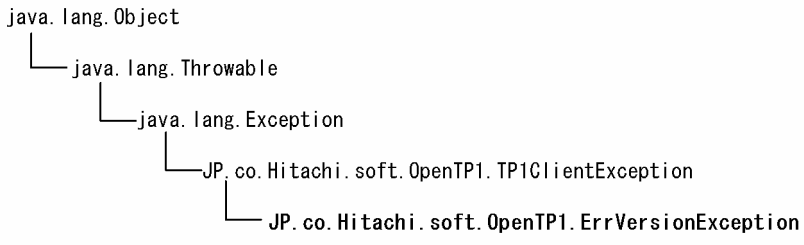
コンストラクタ

●ErrTrnchkExtendException

```
public ErrTrnchkExtendException()
```

詳細メッセージなしで ErrTrnchkExtendException のインスタンスを作成します。

クラス ErrVersionException




```
public class ErrVersionException
extends TP1ClientException
```

TP1/Client/J が返す例外のクラスです。次の事象を表します。

通知元サーバのバージョンが不正です。

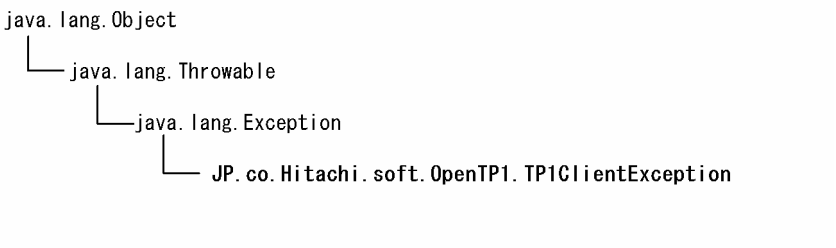
コンストラクタ

●ErrVersionException

```
public ErrVersionException()
```

詳細メッセージなしで ErrVersionException クラスのインスタンスを作成します。

クラス TP1ClientException



直系の既知のサブクラス

```
ErrBufferOverflowException
ErrCollisionMessageException
ErrConnfreeException
ErrFatalException
ErrHazardException
ErrHazardNoBeginException
ErrHeuristicException
ErrHeuristicNoBeginException
ErrHostUndefException
ErrInitializingException
ErrInvalidArgsException
ErrInvalidMessageException
ErrInvalidPortException
ErrInvalidReplyException
ErrIOErrException
ErrMessageTooBigException
ErrNetDownException
ErrNoBeginException
ErrNoBufsAtServerException
ErrNoBufsException
ErrNoSuchServiceException
ErrNoSuchServiceGroupException
ErrNotUpException
ErrProtoException
ErrReplyTooBigException
ErrRMException
ErrRollbackException
ErrRollbackNoBeginException
ErrSecchkException
ErrSecurityException
ErrServerBusyException
ErrServiceClosedException
ErrServiceNotUpException
ErrServiceTerminatedException
ErrServiceTerminatingException
ErrSyserrAtServerException
ErrSyserrException
ErrTestmodeException
ErrTimedOutException
ErrTMException
ErrTrnchkException
```

```
public class TP1ClientException
extends java.lang.Exception
```

TP1ClientException クラスは TP1/Client/J が返すすべての例外のスーパークラスです。

関連項目

TP1Client

コンストラクタ

●TP1ClientException

```
public TP1ClientException()
```

詳細メッセージなしで TP1ClientException のインスタンスを作成します。

●TP1ClientException

```
public TP1ClientException(String msg)
```

詳細メッセージ付きで TP1ClientException のインスタンスを作成します。

詳細メッセージは getMessage メソッドで取り出せます。

5 定義

TP1/Client/J 環境定義について説明します。

5.1　定義の概要

この節では，TP1/Client/J 環境定義の一覧，定義の規則，およびパス名の形式について説明します。

5.1.1　TP1/Client/J 環境定義の一覧

TP1/Client/J 環境定義の一覧を，次の表に示します。

表 5-1　TP1/Client/J 環境定義の一覧

項番	オペランド	定義内容	指定値
1	dcnamuse	ネームサービスを使用した RPC を使用するかどうかを指定	Y ｜《N》
2	dcnamport	ネームサービスのポート番号	〈符号なし整数〉((5001～65535))《10000》
3	dchost	窓口となる TP1/Server	〈文字列〉
4	dcwatchtim	最大応答待ち時間	〈符号なし整数〉((0～65535))《180》（単位：秒）
5	dccltinquiretime	問い合わせ間隔最大時間	〈符号なし整数〉((0～1048575))（単位：秒）
6	dcwatchtiminherit	リモート API 機能を使用した RPC を行う場合に，CUP の最大応答待ち時間を rap サーバ側に引き継ぐかどうかを指定	Y ｜《N》
7	dcwatchtimrpcinherit	CUP の最大応答待ち時間を TP1/Server 側に引き継ぐかどうかを指定	Y ｜《N》
8	dccltdelay	最大通信遅延時間	〈符号なし整数〉((0～65535))《0》（単位：秒）
9	dcselint	応答電文監視時間間隔	〈符号なし整数〉((1～65535))《100》（単位：ミリ秒）
10	dccltextend	TP1/Client/J の機能拡張レベルの設定を指定	《00000000》｜00000001
11	dccache	ネームキャッシュの最大エントリ数	〈符号なし整数〉((2～256))《8》
12	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》
13	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，スケジューラダイレクト機能を使用した RPC を使用するかどうかを指定	Y ｜《N》
14	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))

項番	オペランド	定義内容	指定値
15	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》
16	dcrapdirect	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 rap リスナーに問い合わせる機能，すなわちリモート API 機能を使用するかどうかを指定	《Y》｜ N
17	dcrapport	rap リスナーのポート番号	〈符号なし整数〉((5001～65535))
18	dcrapautoconnect	CUP と rap サーバ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》
19	dcerrtrace	CUP でエラートレースを取得するかどうかを指定	Y ｜《N》
20	dcerrtracepath	エラートレースファイル作成ディレクトリ	〈文字列〉
21	dcerrtracesize	エラートレースファイルサイズ	〈符号なし整数〉((4096～1048576))《4096》(単位：バイト)
22	dcmethodtrace	CUP でメソッドトレースを取得するかどうかを指定	Y ｜《N》
23	dcmethodtracepath	メソッドトレースファイル作成ディレクトリ	〈文字列〉
24	dcmethodtracesize	メソッドトレースファイルサイズ	〈符号なし整数〉((4096～1048576))《4096》(単位：バイト)
25	dcuaptrace	CUP で UAP トレースを取得するかどうかを指定	Y ｜《N》
26	dcuaptracepath	UAP トレースファイル作成ディレクトリ	〈文字列〉
27	dcuaptracesize	UAP トレースファイルサイズ	〈符号なし整数〉((4096～1048576))《4096》(単位：バイト)
28	dcdatatrace	CUP でデータトレースを取得するかどうかを指定	Y ｜《N》
29	dcdatatracepath	データトレースファイル作成ディレクトリ	〈文字列〉
30	dcdatatracesize	データトレースファイルサイズ	〈符号なし整数〉((4096～1048576))《4096》(単位：バイト)
31	dcdatatracemaxsize	データトレースの最大データ長	〈符号なし整数〉((16～1048576))《128》(単位：バイト)
32	dccltrstatisitem	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］…

項番	オペランド	定義内容	指定値
33	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〕…
34	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）
35	dcclttrrbinfo	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
36	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）
37	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
38	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
39	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）
40	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）
41	dcclttrexpsp	トランザクションブランチの処理を監視するとき，次の処理時間も監視時間に含むかどうかを指定	Y ｜ N ｜ F
42	dcsndrcvtype	TCP/IP 通信機能の使用時に，初期化する環境を指定	DCCLT_ONEWAY_SND ｜ DCCLT_ONEWAY_RCV ｜ DCCLT_SNDRCV
43	dcrcvport	受信用の CUP のポート番号	〈符号なし整数〉((1～65535))《11000》
44	dcsndhost	接続する MHP が存在するノードのホスト名	〈文字列〉
45	dcsndport	接続する MHP のポート番号	〈符号なし整数〉((1～65535))《12000》
46	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》
47	dcweburl	サービス要求先の TP1/Web の URL	〈パス名〉
48	dccltdbgtrcfilecount	デバッグトレースファイルの最大ファイル数	〈符号なし整数〉((0～256))《0》
49	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）

項番	オペランド	定義内容	指定値
50	dcscdhostchange	TP1/Client/J 環境定義の dchost オペランドに指定したサービス要求先スケジューラを RPC ごとに分散させるかどうかを指定	Y | 《N》
51	dccltloadbalance	ネームサーバから得た複数のサービス要求先スケジューラの情報から，負荷レベルが最も低いサービス要求先スケジューラの情報をキャッシュに格納するかどうかを指定	Y | 《N》
52	dccltcachetim	キャッシュの有効期限	〈符号なし整数〉((0～65535))《30》（単位：秒）
53	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）
54	dccltprftrace	TP1/Client/J を Cosminexus Application Server 上で動作させた場合に，Cosminexus Application Server の性能解析トレースを取得するかどうかを指定	《Y》 | N
55	dccltprfinfosend	TP1/Server に対して，ネームサービスを使用した RPC，およびスケジューラダイレクト機能を使用した RPC を行う場合に，OpenTP1 の性能検証用トレースに出力する情報として TP1/Client/J 内部で識別情報を付加するかどうかを指定	《Y》 | N
56	dccltnammlthost	マルチホームドホスト形態の TP1/Server に対してネームサービスを使用した RPC を行うかどうかを指定	Y | 《N》
57	dccltdatacomp	データ圧縮機能を使用するかどうかを指定	Y | 《N》
58	dccltconnectinf	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	端末識別情報
59	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》
60	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》
61	dccltcupsndhost	CUP の送信元ホスト	〈文字列〉
62	dccltcuprcvport	サーバからのメッセージを受信する CUP のポート番号	〈符号なし整数〉((5001～65535))
63	dcclttrcmplmttm	トランザクション完了限界時間	〈符号なし整数〉((0～65535))（単位：秒）
64	dccltsendbuffsize	TCP/IP の送信バッファサイズ	〈符号なし整数〉((8192～2147483647))（単位：バイト）
65	dccltrecvbuffsize	TCP/IP の受信バッファサイズ	〈符号なし整数〉((8192～2147483647))（単位：バイト）

項番	オペランド	定義内容	指定値
66	dcinquiretimecheck	常設コネクションを使用し，かつオートコネクトモードの場合に，CUP 実行プロセスで問い合わせ間隔最大時間を監視するかどうかを指定	《Y》｜N
67	dcnotifyreshost	サービス要求元が応答電文を受信するために使用する IP アドレスを通知するかどうかを指定	Y｜《N》
68	dcresponsehost	サービス要求元が応答電文を受信するために使用する IP アドレスを通知する場合のホスト名	〈文字列〉

5.1.2　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。 （例）dcnamuse=Y ｜ N 　これは，dcnamuse=Y と指定するか，または dcnamuse=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。

属性表示記号	意味
(( ))	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）

5.1.3 パス名の記述形式

次の四つのオペランドにパス名を指定する場合，ファイル区切り文字に"¥"を使用するときは，"¥"を"¥¥"と記述してください。

- dcerrtracepath
- dcmethodtracepath
- dcuaptracepath
- dcdatatracepath

例　"C:¥Clt4J¥trace"にエラートレースを取得する場合

```
dcerrtracepath = C:¥¥Clt4J¥¥trace
```

または，

```
dcerrtracepath = C:/Clt4J/trace
```

5.2　TP1/Client/J 環境定義の詳細

Java アプリケーション，または Java サーブレットでは，TP1/Client/J 環境定義を任意のファイルに格納して rpcOpen メソッドの引数でこのファイル名を指定します。また，rpcOpen メソッドを引数無しで呼び出した場合は，システムプロパティとして定義することもできます。

Java アプレットでは，param タグで TP1/Client/J 環境定義を定義します。

TP1/Client/J 環境定義を次に示します。

5.2.1　形式

```
〔dcnamuse=Y|N〕
〔dcnamport=ネームサービスのポート番号〕
〔dchost=窓口となるTP1/Serverのホスト名〕
〔dcwatchtim=最大応答待ち時間〕
〔dccltinquiretime=問い合わせ間隔最大時間〕
〔dcwatchtiminherit=Y|N〕
〔dcwatchtimrpcinherit=Y|N〕
〔dccltdelay=最大通信遅延時間〕
〔dcselint=応答電文監視時間間隔〕
〔dccltextend=機能拡張レベル〕
〔dccache=ネームキャッシュの最大エントリ数〕
〔dchostselect=Y|N〕
〔dcscddirect=Y|N〕
〔dcscdport=スケジュールサービスのポート番号〕
〔dcscdloadpriority=Y|N〕
〔dcrapdirect=Y|N〕
〔dcrapport=rapリスナーのポート番号〕
〔dcrapautoconnect=Y|N〕
〔dcerrtrace=Y|N〕
〔dcerrtracepath=エラートレースファイル作成ディレクトリ〕
〔dcerrtracesize=エラートレースファイルサイズ〕
〔dcmethodtrace=Y|N〕
〔dcmethodtracepath=メソッドトレースファイル作成ディレクトリ〕
〔dcmethodtracesize=メソッドトレースファイルサイズ〕
〔dcuaptrace=Y|N〕
〔dcuaptracepath=UAPトレースファイル作成ディレクトリ〕
〔dcuaptracesize=UAPトレースファイルサイズ〕
〔dcdatatrace=Y|N〕
〔dcdatatracepath=データトレースファイル作成ディレクトリ〕
〔dcdatatracesize=データトレースファイルサイズ〕
〔dcdatatracemaxsize=データトレースの最大データ長〕
〔dcclttrstatisitem=統計情報項目〕
〔dcclttroptiitem=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〕
〔dcclttrexptm=トランザクションブランチ限界経過時間〕
〔dcclttrcputm=トランザクションブランチCPU監視時間〕
```

```
〔dccltrexpsp=Y|N|F〕
〔dcsndrcvtype=DCCLT_ONEWAY_SND｜DCCLT_ONEWAY_RCV｜DCCLT_SNDRCV〕
〔dcrcvport=受信用のCUPのポート番号〕
〔dcsndhost=接続するMHPが存在するノードのホスト名〕
〔dcsndport=接続するMHPのポート番号〕
〔dcsockopenatrcv=Y|N〕
〔dcweburl=サービス要求先のTP1/WebのURL〕
〔dccltdbgtrcfilecount=デバッグトレースファイルの最大ファイル数〕
〔dccltrpcmaxmsgsize=RPC送受信電文の最大長〕
〔dcscdhostchange=Y|N〕
〔dccltloadbalance=Y|N〕
〔dccltcachetim=キャッシュの有効期限〕
〔dccltconnecttimeout=コネクション確立最大監視時間〕
〔dccltprftrace=Y|N〕
〔dccltprfinfosend=Y|N〕
〔dccltnammlthost=Y|N〕
〔dccltdatacomp=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〕
〔dccltcupsndhost=送信元ホスト〕
〔dccltcuprcvport=CUPの受信で使用するポート番号〕
〔dccltrcmplmttm=トランザクション完了限界時間〕
〔dccltsendbuffsize=TCP/IPの送信バッファサイズ〕
〔dccltrecvbuffsize=TCP/IPの受信バッファサイズ〕
〔dcinquiretimecheck=Y|N〕
〔dcnotifyreshost=Y|N〕
〔dcresponsehost=ホスト名〕
```

5.2.2　オペランド

●dcnamuse=Y｜N　～《N》

ネームサービスを使用した 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た RPC を使用します。

N：ネームサービスを使用した RPC を使用しません。

ネームサービスを使用した RPC を行う場合，TP1/Client/J 環境定義の dchost オペランドに窓口となる TP1/Server のホスト名とネームサービスのポート番号を指定してください。ポート番号を省略する場合は，TP1/Client/J 環境定義の dcnamport オペランドにネームサービスのポート番号を指定してください。詳しくは，TP1/Client/J 環境定義の dchost オペランドを参照してください。TP1/Client/J 環境定義の dcnamuse，dcrapdirect，および dcscddirect オペランドは排他関係にあるため，同時に Y を指定できません。これらのオペランドの二つ以上に Y を指定した場合，定義エラーとなります。TP1/Client/J 環境定義の dcnamuse，dcrapdirect，および dcscddirect オペランドのユースケースごとの設定方法は，TP1/Client/J 環境定義の dcrapdirect オペランドを参照してください。

TP1/Web 接続機能は，dcrapdirect オペランドに Y を指定した場合にだけ使用できます。dcweburl オペランドを定義して，かつ dcscddirect=Y および dcrapdirect=Y の場合は，定義エラーとなります。

●dcnamport=ネームサービスのポート番号

～〈符号なし整数〉((5001～65535))《10000》

クライアントが属している OpenTP1 のシステム共通定義の name_port オペランドで指定したネームサービスのポート番号を指定します。dchost オペランドでポート番号を指定していないホストに対して，この定義の値が有効になります。

●dchost=窓口となる TP1/Server　～〈文字列〉

窓口となる TP1/Server のホスト名と，rap リスナーのポート番号，スケジュールサービスのポート番号，またはネームサービスのポート番号を指定します。区切り文字','を使用して，複数の TP1/Server を指定できます。RPC を rap リスナー経由で行う場合は，rap リスナーのホスト名，またはファイアウォールのホスト名を指定します。

dchost オペランドと dcweburl オペランドの両方を指定した場合は，dcweburl オペランドの指定が有効になります。

なお，XA リソースサービス機能を使用する場合，dchost オペランドに指定できる窓口となる TP1/Server は一つだけです。複数指定した場合の動作は保証できません。

形式

dcrapdirect=Y の場合

ホスト名〔:rap リスナーのポート番号〕〔,ホスト名〔:rap リスナーのポート番号〕,…〕

dcscddirect=Y の場合

ホスト名〔:スケジュールサービスのポート番号〕〔,ホスト名〔:スケジュールサービスのポート番号〕,…〕

dcnamuse=Y の場合

ホスト名〔:ネームサービスのポート番号〕〔,ホスト名〔:ネームサービスのポート番号〕,…〕

・ホスト名　～〈文字列〉

・ポート番号　～〈符号なし整数〉((5001～65535))

指定できる最大文字数は 256 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。ホスト名には，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。

TP1/Server のホスト名を二つ以上指定した場合，窓口となる TP1/Server の障害を検出したとき，dchost オペランドに指定されている次の TP1/Server を参照して切り替えを試みます。ただし，RPC を rap リスナー経由で行う場合は，TP1/Client/J 環境定義で dcrapautoconnect=Y，または setDccltextend メソッドで DCRPC_RAP_AUTOCONNECT を指定した場合だけ，dchost オペランドの次の TP1/Server を参照して切り替えを試みます。

ポート番号は，dcrapdirect=Y の場合，rap リスナーのポート番号を指定します。dcscddirect=Y の場合，スケジュールサービスのポート番号を指定します。dcnamuse=Y の場合，ネームサービスのポート番号を指定します。

ポート番号を省略すると，dcrapdirect=Y の場合，dcrapport オペランドの値を仮定します。dcscddirect=Y の場合，dcscdport オペランドの値を仮定します。dcnamuse=Y の場合，dcnamport オペランドの値を仮定します。

●dcwatchtim=最大応答待ち時間

～〈符号なし整数〉((0～65535))《180》（単位：秒）

CUP から TP1/Server に対しての要求を開始し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，応答を受信するまで無限に待ち続けます。

dcwatchtim オペランドの指定が適用される API（メソッド）を次に示します。

- closeConnection メソッド
- openConnection メソッド
- rpcCall メソッド（スケジューラダイレクト機能を使用した非応答型 RPC の場合を除く）
- rpcCallTo メソッド（同期応答型 RPC の場合）
- trnBegin メソッド
- trnChainedCommit メソッド
- trnChainedRollback メソッド
- trnUnchainedCommit メソッド
- trnUnchainedRollback メソッド

●dccltinquiretime=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行ってから，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。

問い合わせ間隔最大時間は rap サーバで監視するタイマです。rap サーバでは，問い合わせ間隔を監視し，問い合わせ間隔最大時間を超えても rap サーバに問い合わせがない場合，常設コネクションを解放します。また，トランザクション内で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，問い合わせ間隔を監視しません。この指定を省略した場合は，rap サーバで指定された問い合わせ間隔最大時間が有効になります。

このオペランドに 0 以外を指定し，オートコネクトモードで rpcCall メソッドを使用して RPC を実行した場合，CUP 実行プロセスでも問い合わせ間隔最大時間を監視します。

CUP 実行プロセスで問い合わせ間隔最大時間を監視するかどうかは，dcinquiretimecheck オペランドの指定値に従います。このオペランドとあわせて，dcinquiretimecheck オペランドも参照してください。

なお，rap サーバが常設コネクションを解放するタイミングと，CUP 実行プロセスが RPC を実行するタイミングが重なった場合，RPC の実行に失敗して ErrNetDownAtClientException 例外を返すことがあります。

この現象を回避するため，次のオペランドを指定し，CUP 実行プロセスで「問い合わせ間隔最大時間」をチェックするように設定してください。

- dcinquiretimecheck オペランドに Y を指定
- dccltinquiretime オペランドに 0 以外を指定
- dcrapautoconnect オペランドに Y を指定

オペランドの詳細は，TP1/Client/J 環境定義の dcinquiretimecheck，および dcrapautoconnect オペランドを参照してください。

●dcwatchtiminherit=Y | N　～《N》

リモート API 機能を使用した RPC を行う場合に，CUP の最大応答待ち時間を rap サーバ側に引き継ぐかどうかを指定します。

Y：rap サーバ側に CUP の最大応答待ち時間を引き継ぎます。

N：rap サーバ側に CUP の最大応答待ち時間を引き継ぎません。

●dcwatchtimrpcinherit=Y | N　～《N》

CUP の最大応答待ち時間を TP1/Server 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す。

Y：TP1/Server 側に CUP の最大応答待ち時間を引き継ぎます。

N：TP1/Server 側に CUP の最大応答待ち時間を引き継ぎません。

●dccltdelay=最大通信遅延時間

～〈符号なし整数〉((0～65535))《0》（単位：秒）

CUP と rap サーバとの間の通信オーバヘッドを考慮し，rap サーバ側の応答監視を TP1/Client/J 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，rap サーバ側の監視を指定した時間分だけ早く終了させ，TP1/Client/J 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，TP1/Client/J 環境定義に dcrapdirect=Y および dcwatchtiminherit=Y を指定した場合にだけ有効です。

TP1/Client/J 環境定義に dcwatchtim=0 を指定した場合または setDcwatchtim メソッドで 0 を指定した場合，および TP1/Client/J 環境定義に dcwatchtiminherit=N を指定した場合は，dccltdelay オペランドの指定は無視されます。dcwatchtim オペランドに指定した値から dccltdelay オペランド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も，dccltdelay オペランドの指定は無視され，1 が仮定されます。

●dcselint=応答電文監視時間間隔　～〈符号なし整数〉((1～65535))《100》（単位：ミリ秒）

このオペランドは，旧バージョンとの互換性のためだけに存在します。現在はこのオペランドを指定しても無視されます。

●dccltextend=00000000 | 00000001　～《00000000》

TP1/Client/J の機能拡張レベルの設定を指定します。

00000000：TP1/Client/J の機能を拡張しません。

00000001：rpcCall メソッド呼び出し時，自 CUP の IP アドレスをサービスに連絡します。呼び出したサービスで dc_rpc_get_callers_address()関数を実行し，CUP のアドレスを求める必要がある場合に指定します。また，リモート API 機能を使用した RPC で，rap サーバとクライアント間で通信障害が発生した場合，rap サーバ側が出力するメッセージにクライアント側の IP アドレスが出力されるようになります。

●dccache=ネームキャッシュの最大エントリ数　～〈符号なし整数〉((2～256))《8》

TP1/Client/J で使用するサービス情報をキャッシュするためのキャッシュの最大エントリ数を指定します。LRU 方式で管理します。キャッシュのエントリ数が指定値を超える場合は，最も過去に呼び出されたサービス情報を削除します。

●dchostselect=Y | N　～《N》

rpcOpen メソッド実行時，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，dchost オペランドに窓口となる TP1/Server を複数指定した場合にだけ有効です。

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。

窓口となる TP1/Server 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，dchost オペランドに指定された次の TP1/Server に切り替えを試みます。詳しくは，「2.15　ホスト切り替え機能」を参照してください。

●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，スケジューラダイレクト機能を使用した RPC を使用するかどうかを指定します。

Y：スケジューラダイレクト機能を使用した RPC を使用します。

N：スケジューラダイレクト機能を使用した RPC を使用しません。

スケジューラダイレクト機能を使用する場合，TP1/Client/J 環境定義の dchost オペランドに窓口となる TP1/Server のホスト名とスケジュールサービスのポート番号を指定してください。ポート番号を省略する場合は，TP1/Client/J 環境定義の dcscdport オペランドにスケジュールサービスのポート番号を指定してください。詳しくは，TP1/Client/J 環境定義の dchost オペランドを参照してください。TP1/Client/J 環境定義の dcrapdirect，dcscddirect，および dcnamuse オペランドは排他関係にあるため，同時に Y を指定できません。これらのオペランドの二つ以上に Y を指定した場合，定義エラーとなります。TP1/Client/J 環境定義の dcscddirect，dcrapdirect，および dcnamuse オペランドのユースケースごとの設定方法は，TP1/Client/J 環境定義の dcrapdirect オペランドを参照してください。

TP1/Web 接続機能は，dcrapdirect オペランドに Y を指定した場合にだけ使用できます。dcweburl オペランドを定義して，かつ dcscddirect=Y および dcrapdirect=Y の場合は，定義エラーとなります。

●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))

クライアントが属している OpenTP1 のスケジュールサービス定義の scd_port オペランドで指定したスケジュールサービスのポート番号を指定します。

TP1/Client/J 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）のポート番号を指定することもできます。

dchost オペランドでポート番号を指定していないホストに対して，この定義の値が有効になります。このオペランドは TP1/Client/J 環境定義に dcscddirect=Y を指定したときだけ有効です。詳しくは，TP1/Client/J 環境定義の dcscddirect オペランドを参照してください。

●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：サービス要求を受け付けた窓口となる TP1/Server を優先しません。TP1/Server は，ノード間の負荷を分散します。

この定義は，スケジューラダイレクト機能を使用した RPC を行う場合（TP1/Client/J 環境定義に dcscddirect=Y を指定）にだけ，有効です。

●dcrapdirect=Y ｜ N　～《Y》

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 rap リスナーに問い合わせる機能，すなわちリモート API 機能を使用するかどうかを指定します。リモート API 機能を使用する場合，TP1/Client/J 環境定義の dchost オペランドに窓口となる TP1/Server のホスト名と rap リスナーのポート番号を指定してください。ポート番号を省略する場合は，TP1/Client/J 環境定義の dcrapport オペランドに rap リスナーのポート番号を指定してください。詳しくは，TP1/Client/J 環境定義の dchost オペランドを参照してください。

オートコネクトモードの場合，TP1/Client/J 環境定義の dccltinquiretime オペランドに 0 以外を指定する必要があります。詳しくは，TP1/Client/J 環境定義の dcinquiretimecheck オペランドを参照してください。

dcrapdirect，dcscddirect，および dcnamuse オペランドは排他関係にあるため，同時に Y を指定することはできません。これらのオペランドの二つ以上に Y を指定した場合，定義エラーとなります。ユースケースごとの TP1/Client/J 環境定義の指定値を次の表に示します。

表 5-2　ユースケースごとの TP1/Client/J 環境定義の指定値

ユースケース※	TP1/Client/J 環境定義のオペランドの指定		
	dcrapdirect	dcscddirect	dcnamuse
リモート API 機能を使用した RPC を行う場合	Y または 省略	N または 省略	N または 省略
スケジューラダイレクト機能を使用した RPC を行う場合	N または 省略	Y	N または 省略
ネームサービスを使用した RPC を行う場合	N または 省略	N または 省略	Y

注※

各機能を使用する場合の詳細なオペランドの設定は，TP1/Client/J 環境定義の dcrapdirect，dcscddirect，および dcnamuse オペランドを参照してください。

●dcrapport=rap リスナーのポート番号　～〈符号なし整数〉((5001～65535))

クライアントが属している TP1/Server の rap リスナーのポート番号，またはファイアウォールのポート番号を指定します。dchost オペランドでポート番号を指定していないホストに対して，このオペランドの値が有効になります。

このオペランドは，TP1/Client/J 環境定義に dcrapdirect=Y を指定したときだけ有効です。詳しくは，TP1/Client/J 環境定義の dcrapdirect オペランドを参照してください。

●dcrapautoconnect=Y | N　～《N》

CUP と rap サーバ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：次回メソッド実行時に自動的に常設コネクションを確立します。常設コネクション確立要求先は，TP1/Client/J 環境定義の dchost オペランドに指定した rap リスナーです。

N：常設コネクションを自動的に確立させません。

コネクション確立モードの違いによる各メソッドの動作

非オートコネクトモードの場合およびオートコネクトモードの場合のコネクション確立処理は，次の表のようになります。

表 5-3　コネクション確立モードの違いによる各メソッドの動作

dcrapautoconnect オペランドの指定値	openConnection()メソッドを発行	openConnection(host, port)メソッドを発行	openConnection(url, flags)メソッドを発行	rpcCall()メソッドを発行
N	• dchost オペランドが指定されている場合，dchost オペランドに指定された rap サーバにコネクションを確立する。 • dcweburl オペランドが指定されている場合，dcweburl オペランドに指定された URL にコネクションを確立する。 • dchost オペランドおよび dcweburl オペランドが指定されていない場合，ErrHostUndefException が返される。	host 引数および port 引数に指定された rap サーバにコネクションを確立する。	url 引数に指定された URL にコネクションを確立する。	ErrProtoException が返される。
Y	ErrProtoException が返される。	ErrProtoException が返される。	ErrProtoException が返される。	• dchost オペランドが指定されている場合，dchost オペランドに指定された rap サーバにコネクションを確立する。 • dcweburl オペランドが指定されている場合，dcweburl オペランドに指定された URL にコネクションを確立する。 • dchost オペランドおよび dcweburl オペランドが指定されていない場合，

dcrapautoconnect オペランドの指定値	openConnection()メソッドを発行	openConnection(host, port)メソッドを発行	openConnection(url, flags)メソッドを発行	rpcCall()メソッドを発行
Y	ErrProtoException が返される。	ErrProtoException が返される。	ErrProtoException が返される。	ErrHostUndefException が返される。

注

- setRpcextend メソッドによって dcrapautoconnect オペランドの値を動的に変更できます。
- rpcOpen メソッドで定義を読み込みます。
- rpcClose メソッドで読み込んだ定義情報を破棄します。
- 引数に URL を指定した openConnection メソッドを実行した場合，指定された URL は，その URL を指定した openConnection メソッドだけで採用します。このメソッドの実行後，closeConnection メソッドでコネクションを切断し，次に引数指定無しの openConnection メソッドを実行しても，前回の引数に指定した URL は採用されません。

●dcerrtrace=Y ｜ N　～《N》

CUP でエラートレースを取得するかどうかを指定します。

Y：エラートレースを取得します。

N：エラートレースを取得しません。

CUP を Java アプレットとして動作させる場合，Y を指定しないでください。Y を指定した場合の動作は保証されません。

●dcerrtracepath=エラートレースファイル作成ディレクトリ　～〈文字列〉

エラートレースファイルを作成するディレクトリのパス名を指定します。

このオペランドは，TP1/Client/J 環境定義に dcerrtrace=Y を指定した場合にだけ有効です。

●dcerrtracesize=エラートレースファイルサイズ　～〈符号なし整数〉((4096～1048576))《4096》(単位：バイト)

エラートレースファイルのサイズを指定します。

●dcmethodtrace=Y ｜ N　～《N》

CUP でメソッドトレースを取得するかどうかを指定します。

Y：メソッドトレースを取得します。

N：メソッドトレースを取得しません。

CUP を Java アプレットとして動作させる場合，Y を指定しないでください。Y を指定した場合の動作は保証されません。

●dcmethodtracepath=メソッドトレースファイル作成ディレクトリ　～〈文字列〉

メソッドトレースファイルを作成するディレクトリのパス名を指定します。

このオペランドは，TP1/Client/J 環境定義に dcmethodtrace=Y を指定した場合にだけ有効です。

●dcmethodtracesize=メソッドトレースファイルサイズ　～〈符号なし整数〉((4096～1048576))《4096》(単位：バイト)

メソッドトレースファイルのサイズを指定します。

●dcuaptrace=Y｜N　～《N》

CUP で UAP トレースを取得するかどうかを指定します。

Y：UAP トレースを取得します。

N：UAP トレースを取得しません。

CUP を Java アプレットとして動作させる場合，Y を指定しないでください。Y を指定した場合の動作は保証されません。

●dcuaptracepath=UAP トレースファイル作成ディレクトリ　～〈文字列〉

UAP トレースファイルを作成するディレクトリのパス名を指定します。

このオペランドは，TP1/Client/J 環境定義に dcuaptrace=Y を指定した場合にだけ有効です。

●dcuaptracesize=UAP トレースファイルサイズ　～〈符号なし整数〉((4096～1048576))《4096》(単位：バイト)

UAP トレースファイルのサイズを指定します。

●dcdatatrace=Y｜N　～《N》

CUP でデータトレースを取得するかどうかを指定します。

Y：データトレースを取得します。

N：データトレースを取得しません。

CUP を Java アプレットとして動作させる場合，Y を指定しないでください。Y を指定した場合の動作は保証されません。

●dcdatatracepath=データトレースファイル作成ディレクトリ　～〈文字列〉

データトレースファイルを作成するディレクトリのパス名を指定します。

このオペランドは，TP1/Client/J 環境定義に dcdatatrace=Y を指定した場合にだけ有効です。

●dcdatatracesize=データトレースファイルサイズ　～〈符号なし整数〉((4096～1048576))《4096》(単位：バイト)

データトレースファイルのサイズを指定します。

●dcdatatracemaxsize=データトレースの最大データ長　～〈符号なし整数〉((16～1048576))《128》(単位：バイト)

1 件のデータトレースのデータ長の最大値を指定します。

●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にだけ有効です。

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に trn_tran_statistics=Y を指定
- 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，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にだけ有効です。

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，コミット制御で必要な 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一 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ケージされていること。

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 が返される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，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，または応答など）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にだけ有効です。

指定時間を過ぎ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，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にだけ有効です。

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，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，rpcCall メソッド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。

- rpcCall メソッドのタイムアウト時間

  「K≧このオペランドの指定時間」の場合：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」および「(このオペランドの指定時間－K）≧W」の場合：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」および「(このオペランドの指定時間－K）＜ W」の場合：(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」「W」の内容は次のとおりです。

  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≧このオペランドの指定時間」の場合：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」および「(このオペランドの指定時間－K）≧W」の場合：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」および「(このオペランドの指定時間－K）＜ W」の場合：(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」「W」の内容は次のとおりです。

  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，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。

●dcclttrrbrcv=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にだけ有効です。

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ないで）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，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。

●dccltrrecoverytype=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にだけ有効です。

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および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できないため，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。

このオペランドでは，次に示す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，指定値に示す三つの方式から選択して指定します。

（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ません。そのため，RPC 発行元トランザクションブランチおよび 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。

（障害 2）rap サーバが 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ません。そのため，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。

（障害 3）RPC 発行先 UAP からの応答受信後に rap サーバと 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，相手プロセスのダウンを知らないため，すでに存在しない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。

それぞれの指定値が示す処理方式について説明します。

type1

（障害 1）が発生した場合，RPC 発行元トランザクションブランチおよび 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受信処理がタイムアウトすることによって，トランザクションを決着します。

（障害 2）が発生した場合，RPC 発行元トランザクションブランチは，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しないでトランザクションを決着します。

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受信処理がタイムアウトすることによって，トランザクションを決着します。

（障害 3）が発生した場合，RPC 発行元トランザクションブランチおよび 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受信処理がタイムアウトすることによって，トランザクションを決着します。

type2

（障害 1）が発生してトランザクションをコミットする場合は type1 と同じです。

（障害 1）が発生してトランザクションをロールバックする場合，または（障害 2）が発生した場合は，RPC 発行元トランザクションブランチは，RPC 発行先トランザクションブランチが存在するノードのトランザクションサービスプロセスにトランザクション同期点メッセージを送信後，トランザクションを決着します。トランザクション同期点メッセージを受信したトランザクションサービスプロセスは，該当するトランザクションブランチを処理中のプロセスに，トランザクション同期点指示を送信します。

（障害 3）が発生した場合，RPC 発行元トランザクションブランチおよび 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受信処理がタイムアウトすることによって，トランザクションを決着します。

type3

（障害 1）が発生してトランザクションをコミットする場合は，type1 と同じです。

（障害 1）が発生してトランザクションをロールバックする場合，（障害 2）が発生した場合，または（障害 3）が発生した場合，相手トランザクションブランチが存在するノードのトランザクションサービスプロセスに，トランザクション同期点メッセージを送信します。トランザクション同期点メッセージを受信したトランザクションサービスプロセスは，該当するトランザクションブランチを処理中のプロセスに，トランザクション同期点指示を送信します。

次に示す場合，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，UAP 閉塞など），ほかのノードの同一 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，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。

●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にだけ有効です。

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，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に含むかどうかは，TP1/Client/J 環境定義の dcclttrexpsp オペランドで指定してください。

●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）

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にだけ有効です。

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，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。

●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，次の処理時間も監視時間に含むかどうかを指定します。

1. 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
2. 連鎖型 RPC で呼び出されたサーバ UAP が次のサービス要求を待つ時間
3. 監視対象のトランザクションブランチが，非応答型 RPC を使用してほかのトランザクションブランチを呼び出したあと，処理結果受信処理を行っている時間

それぞれの指定値について説明します。

Y：1，2，3 すべてを監視時間に含みます。

N：3 だけを監視時間に含みます。

F：1，2，3 のどれも監視時間に含みません。

この指定を省略した場合は，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。

●dcsndrcvtype=DCCLT_ONEWAY_SND | DCCLT_ONEWAY_RCV | DCCLT_SNDRCV

TCP/IP 通信機能の使用時に初期化する環境を指定します。

このオペランドを指定してない場合は，TCP/IP 通信機能を使用できません。

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

●dcrcvport=受信用の 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。同一マシン

内で，複数のプロセスまたは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（TP1/Client/J 環境定義の dcsndrcvtype オペランドに 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=接続する MHP が存在するノードのホスト名　～〈文字列〉

CUP から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 MHP が存在するノードのホスト名を指定します。

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。

●dcsndport=接続する MHP のポート番号

～〈符号なし整数〉((1～65535))《12000》

CUP から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 MHP のポート番号を指定します。dcnamport オペランドに指定するポート番号と重複しないように指定してください。

●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。

なお，このオペランドの指定を省略した場合は，rpcOpen メソッド実行時に受信用ソケットを開設します。

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。そのため，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「2.4.5　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：rpcOpen メソッド実行時に受信ソケットを開設しません。ただし，次のメソッドを実行したときに，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。

- cltReceive
- cltAssemReceive

N：rpcOpen メソッド実行時に，受信用ソケットを開設します。

●dcweburl=サービス要求先の TP1/Web の URL　～〈パス名〉

サービス要求先の TP1/Web の URL を指定します。

プロトコル，WWW サーバ，プロンプターの CGI 名称，TP1/Web のサービス名などの情報を URL 形式で指定します。

dchost オペランドと dcweburl オペランドの両方を指定した場合は，dcweburl オペランドの指定が有効になります。

また，TP1/Web 接続機能を使用する場合は，dcscddirect オペランドおよび dcnamuse オペランドを使用できません。dcweburl オペランドを定義して，かつ dcscddirect=Y および dcnamuse=Y の場合は，定義エラーとなります。

●dccltdbgtrcfilecount=デバッグトレースファイルの最大ファイル数

～〈符号なし整数〉((0～256))《0》

デバッグトレースファイルの最大ファイル数を指定します。0 を指定した場合，ファイル数の制限はなくなります。デバッグトレースファイルの総数が，このオペランドの指定値を超えた場合，更新日時の最も古いファイルが削除されます。

dccltdbgtrcfilecount オペランドで異なる値を指定した複数の TP1Client インスタンスを同時に使用した場合，実際の最大ファイル数は保証できません。

rpcOpen メソッド未発行の状態で TP1Client インスタンスの作成した場合，dccltdbgtrcfilecount オペランドは，システムプロパティとしても定義できます。この場合，TP1Client インスタンスの作成後に rpcOpen メソッドを発行すると，rpcOpen メソッドの内容を解析した結果を上書きして動作します。rpcOpen メソッドを発行するとシステムプロパティに定義された dccltdbgtrcfilecount オペランドの指定は無効になります。

dccltdbgtrcfilecount オペランドの指定が不正な場合，エラーは通知されません。この場合，dccltdbgtrcfilecount オペランドの指定はデフォルト値が有効になります。

●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 で受け渡しできる電文の最大長を指定します。

rpcCall メソッドで送受信できる RPC 電文の最大長は，このオペランドを省略した場合，またはデフォルト値（1）を指定した場合は，1 メガバイト（1048576 バイト）です。このオペランドに指定できる最大値（8）を指定した場合は，8 メガバイト（8388608 バイト）です。このオペランドの指定よりも大きいサイズの送信メッセージ長および受信メッセージ長を，rpcCall メソッドの引数に指定することはできません。送信メッセージ長または受信メッセージ長がこのオペランドの指定値よりも大きい場合，rpcCall メソッドは ErrInvalidArgsException を返します。

dccltrpcmaxmsgsize オペランドを指定した場合，次の機能は使用しないでください。通信先の TP1/Server ノードでエラーが発生します。

- スケジューラダイレクト機能を使用した RPC
- 通信先を指定した RPC
- TP1/Web 接続機能

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（バージョン 06-02 以降）以外の TP1/Server にサービス要求した場合の動作については次の表を参照してください。

表 5-4　dccltrpcmaxmsgsize オペランドに 2 以上を指定した場合の rpcCall メソッドの動作

メソッド	RPC の種類を指定する定義	窓口となる TP1/Server のバージョン	ノードのバージョン※1	
			06-02 より前の場合	06-02 以降の場合※2
rpcCall	dcrapdirect=Y	06-02 より前	×※3	×※3
		06-02 以降	×※4	○
	dcnamuse=Y	06-02 より前	×※4	○
		06-02 以降	×※4	○

（凡例）

○：要求できます。

×：要求できません。

注※1

窓口となる TP1/Server に指定されている all_node オペランドで定義された，SPP を起動しているノードのバージョンです。

注※2

システム共通定義に rpc_max_message_size オペランドを指定している場合。

注※3

入力データ長が 1 メガバイト以上の場合は，ErrMessageTooBigException が発生し，入力データ長が 1 メガバイト以下で出力領域長が 1 メガバイト以上の場合は ErrInvalidArgsException が発生します。

注※4

ErrNoSuchServiceGroupException が発生します。

●dcscdhostchange=Y | N　～《N》

TP1/Client/J 環境定義の dchost オペランドに指定したサービス要求先スケジューラを RPC ごとに分散させるかどうかを指定します。

Y：サービス要求先スケジューラを RPC ごとに分散させます。

N：サービス要求先スケジューラを RPC ごとに分散させません。

Y を指定した場合の処理を説明します。

RPC でのサービス要求が 1 度目の場合，TP1/Client/J 環境定義に dchostselect=Y を指定したときは，TP1/Client/J 環境定義の dchost オペランドに指定したサービス要求先スケジューラがランダムに選択されます。TP1/Client/J 環境定義に dchostselect=N を指定したときは，TP1/Client/J 環境定義の dchost オペランドの先頭に指定したサービス要求先スケジューラが選択されます。

RPC でのサービス要求が 2 度目以降の場合，前回のサービス要求先スケジューラから dchost オペランドで指定した順にラウンドロビン方式でサービス要求先スケジューラが選択されます。

●dccltloadbalance=Y | N　～《N》

ネームサーバから得た複数のサービス要求先スケジューラの情報から，負荷レベルが最も低いサービス要求先スケジューラの情報をキャッシュに格納するかどうかを指定します。

Y：負荷レベルが最も低いサービス要求先スケジューラの情報をキャッシュに格納します。

N：負荷レベルが最も低いサービス要求先スケジューラの情報をキャッシュに格納しません。

Y を指定した場合の処理を説明します。

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，キャッシュに格納されたサービス要求先スケジューラが，ラウンドロビン方式で選択されます。

負荷レベルの詳細については，マニュアル「OpenTP1 解説」を参照してください。

●dccltcachetim=キャッシュの有効期限

～〈符号なし整数〉((0～65535))《30》(単位：秒)

サービス要求先スケジューラの情報をキャッシュに格納した時点からの，キャッシュの有効期限を指定します。キャッシュの有効期限を満了したサービス要求先スケジューラの情報は，キャッシュから削除されます。その後，サービス要求先スケジューラの情報を再度取得して，その情報をキャッシュに格納することで，キャッシュを更新します。

一度格納したサービス情報は rpcClose メソッドを発行するまで，またはキャッシュのエントリ数が dccache の指定値を超えた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，0 を指定した場合は，有効期限はありません。

このオペランドは，TP1/Client/J 環境定義に dccltloadbalance=Y を指定したときだけ有効です。

●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)

データ送信時のコネクション確立処理に対する最大監視時間を指定します。

このオペランドで指定する値は，java.net.Socket.connect メソッド（以降「connect メソッド」と呼びます）の処理時間ではなく，connect メソッドで実行されるコネクションの確立処理に掛かる最大監視時間です。

相手システムが起動していないなどの要因でコネクションを確立できない場合，このオペランドで指定した監視時間が経過する前に CUP から発行したメソッドがエラーリターンすることがあります。これは，このオペランド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および再送処理の間隔は，プラットフォームによって異なります。また，使用するメソッドや機能によっては，connect メソッドの処理時間がこのオペランドで指定した値よりも大きくなる場合があります。

0 を指定した場合，またはこのオペランドの指定を省略した場合は，OS がコネクションの確立処理を監視します。

●dccltprftrace=Y | N ～《Y》

TP1/Client/J を Cosminexus Application Server 上で動作させた場合に，Cosminexus Application Server の性能解析トレースを取得するかどうかを指定します。

Y：性能解析トレースを取得します。ただし，PRF デーモンが起動していない場合は性能解析トレースを取得しません。

N：性能解析トレースを取得しません。

●dccltprfinfosend=Y | N ～《Y》

TP1/Server に対して，ネームサービスを使用した RPC，およびスケジューラダイレクト機能を使用した RPC を行う場合に，OpenTP1 の性能検証用トレースに出力する情報として TP1/Client/J 内部で識別情報（IP アドレスなど）を付加するかどうかを指定します。

Y：TP1/Client/J 内部で識別情報を付加します。

N：TP1/Client/J 内部で識別情報を付加しません。

●dccltnammlthost=Y ｜ N　～《N》

マルチホームドホスト形態の TP1/Server に対してネームサービスを使用した RPC を行うかどうかを指定します。

Y：マルチホームドホスト形態の TP1/Server に対してネームサービスを使用した RPC を行います。RPC の要求先を複数定義している場合，またはマルチホームドホスト形態の TP1/Server が一つ以上ある場合に指定してください。

N：マルチホームドホスト形態の TP1/Server に対してネームサービスを使用した RPC を行いません。マルチホームドホスト形態の TP1/Server に対してネームサービスを使用した RPC を行うと，エラーが発生する場合があります。

詳細については，「2.2.9(3)　マルチホームドホスト形態の TP1/Server に対して RPC を行う場合の定義」を参照してください。

●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_buflen 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。

●dccltconnectinf=端末識別情報

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。端末識別情報は，先頭に 0x を付け，0x の後ろに 128 文字までの 16 進数表記で指定します。先頭の 0x は 128 文字に含まれません。2 文字で 1 バイトの値として，64 バイト（128 文字）まで指定できます。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，9 バイト目以降に指定した値は無視されます。

このオペランドに指定した値は，setConnectInformation メソッドを呼び出すことで，動的に変更できます。このオペランドに指定した値は，setConnectInformation メソッドを呼び出したあと，再び rpcOpen メソッドを呼び出すまで無視されます。

このオペランドに指定した値は，TP1/Client/J 環境定義の dchost オペランドに DCCM3 論理端末のホスト名を，dchost オペランドまたは dcrapport オペランドに DCCM3 論理端末のポート番号を指定し，次のどちらかの方法で DCCM3 論理端末との常設コネクションを確立した場合に有効となります。

- openConnection メソッドを呼び出します。引数有りの openConnection メソッドの場合，引数 host に DCCM3 論理端末のホスト名，引数 port に DCCM3 論理端末のポート番号を指定します。
- TP1/Client/J 環境定義の dcrapautoconnect オペランドに Y を指定し，rpcCall メソッドを呼び出します。

このオペランドの指定を省略した場合，DCCM3 論理端末に端末識別情報は通知されません。ただし，setConnectInformation メソッドを呼び出した場合は，setConnectInformation メソッドに指定した端末識別情報が，DCCM3 論理端末との常設コネクション確立時に DCCM3 論理端末に通知されます。

●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，TP1/Client/J 環境定義の dchost，dcscdport，dcscdmulticount も併せて参照してください。また，この定義は，rpcCallTo メソッド実行時は無効です。

●dcscdmulticount=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，TP1/Client/J 環境定義 dcscdport に指定するポート番号の内容によって，次のように異なります。

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じ値か，それ以下の値を指定します。	
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，またはそれ以下の値を指定します。	
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，またはそれ以下の値を指定します。
	上記以外	1 を指定するか，または指定を省略します(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，TP1/Client/J 環境定義に dcscdmulti=Y および dcscddirect=Y を指定した場合に有効です。なお，この場合，ポート番号は，次に示す範囲の値からランダムに選択されます。

- 下限値：dchost または dcscdport に指定したポート番号の値
- 上限値：下限値 + dcscdmulticount に指定したプロセス数 - 1

このポート番号の最大値が 65535 を超える場合は，定義解析で ErrFatalException，および setDchost で ErrInvalidPortException が発生します。

scdmulti の-m オプションに指定したプロセス数よりも大きい値を指定した場合，マルチスケジューラが起動していないポートに接続を試み，ErrNetDownAtClientException が発生する場合があります。

●dccltcupsndhost=送信元ホスト　～〈文字列〉

次に示すコネクション確立要求時の，送信元ホストを指定します。

- スケジューラダイレクト機能を使用した RPC
- ネームサービスを使用した RPC
- リモート API 機能を使用した RPC
- 通信先を指定した RPC

- TCP/IP 通信機能

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。

なお，次の場合は，発行したメソッドで例外が発生します。

- ホスト名として localhost を指定した場合
- 127 で始まる IP アドレスを指定した場合
- 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

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を受信する 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。

- スケジューラダイレクト機能を使用した RPC の受信時（受信ポート）
- ネームサービスを使用した RPC の受信時（受信ポート）

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。

●dccltrcmplmttm=トランザクション完了限界時間　～〈符号なし整数〉((0～65535))（単位：秒）

rap サーバで代理実行するトランザクションブランチの，開始から終了までの最大実行時間を指定します。この指定時間を超えた場合，rap サーバのプロセスが異常終了したあとに，トランザクションブランチが回復プロセスでコミット，またはロールバックのどちらかに決着して終了します。

0 を指定した場合は，トランザクションブランチの最大実行時間を監視しません。

この指定を省略した場合は，rap リスナーサービス定義の trn_completion_limit_time オペランドの指定に従います。

この機能をバージョン 07-01 以前の TP1/Server に対して使用した場合，機能が無効の状態で動作します。

●dccltsendbuffsize=TCP/IP の送信バッファサイズ　～〈符号なし整数〉((8192～2147483647))（単位：バイト）

コネクションごとに確保される TCP/IP 送信バッファのサイズを指定します。

高速な通信媒体や MTU の大きな通信媒体を使用している場合，この値を大きくすることによって性能向上を図れます。

このオペランドを省略した場合は，JavaVM の送信バッファサイズを適用します。

このオペランドの値は，JavaVM で使用できる TCP/IP 送信バッファの上限値以下を指定してください。TCP/IP 送信バッファのデフォルト値，変更方法，変更できるサイズの範囲は，OS によって異なります。詳しくは，各 OS のマニュアルを参照してください。また，CUP と通信するすべてのノード

で同じ値を指定してください。同じ値を指定しない場合，通信するノードとバッファサイズに差異が生じ，通信性能が劣化するおそれがあります。

●dccltrecvbuffsize=TCP/IP の受信バッファサイズ　～〈符号なし整数〉((8192～2147483647))（単位：バイト）

コネクションごとに確保される TCP/IP 受信バッファのサイズを指定します。

高速な通信媒体や MTU の大きな通信媒体を使用している場合，この値を大きくすることによって性能向上を図れます。

このオペランドを省略した場合は，JavaVM の受信バッファサイズを適用します。

このオペランドの値は，JavaVM で使用できる TCP/IP 受信バッファの上限値以下を指定してください。TCP/IP 受信バッファのデフォルト値，変更方法，変更できるサイズの範囲は，OS によって異なります。詳しくは，各 OS のマニュアルを参照してください。また，CUP と通信するすべてのノードで同じ値を指定してください。同じ値を指定しない場合，通信するノードとバッファサイズに差異が生じ，通信性能が劣化するおそれがあります。

●dcinquiretimecheck=Y | N　～《Y》

常設コネクションを使用し，かつオートコネクトモード※1 の場合に，CUP 実行プロセスで「問い合わせ間隔最大時間」をチェックするかどうかを指定します。

rap サーバが常設コネクションを解放するタイミングと，CUP 実行プロセスが RPC を実行するタイミングが重なった場合，RPC の実行に失敗して ErrNetDownAtClientException 例外を返すことがあります。

この現象を回避するためには，このオペランドを指定します。

このオペランドを省略，またはこのオペランドに Y を指定した場合，「問い合わせ間隔最大時間」には 0 以外を指定※2 してください。

このオペランドは省略できます。

Y：CUP 実行プロセスで，「問い合わせ間隔最大時間」をチェックします。

N：CUP 実行プロセスで，「問い合わせ間隔最大時間」をチェックしません。

省略した場合は，CUP 実行プロセスでも「問い合わせ間隔最大時間」をチェックします。

CUP 実行プロセスは，トランザクション処理の処理区間外※3，かつ連鎖型 RPC の処理区間外※4 で rpcCall メソッドを使用して RPC を実行するタイミングで問い合わせ間隔をチェックします。

問い合わせ間隔のチェックは次の計算式で求めた時間を使用します。この時間を超過した場合は，常設コネクションを解放して再接続し RPC を実行します。

```
チェック時間 = 問い合わせ間隔最大時間 － T
```

T：問い合わせ間隔最大時間の 2%の値

ただし，250 ミリ秒 ≦ T ≦ 3 秒となります。

注※1

次のどちらかの場合が，オートコネクトモードに該当します。

- TP1/Client/J 環境定義の dcrapautoconnect オペランドに Y を指定している。

- CUP 内で，SetRpcextend メソッドの extendoption パラメタに DCRPC_RAP_AUTOCONNECT を指定して実行している。

注※2

TP1/Client/J 環境定義の dccltinquiretime オペランドに 0 以外を指定してください。または CUP 内で，setDccltinquiretime メソッドの sec パラメタに 0 以外を指定して実行してください。

注※3

次の両方の条件が重なった場合がトランザクション処理の処理区間外となります。

- XA リソースサービス機能を使用していない（CUP が uCosminexus TP1 Connector 上で動作していない，または CUP が uCosminexus TP1 Connector 上で動作しているが Connector 属性ファイルの transaction-support タグに XATransaction を指定していない）。
- trnBegin メソッド実行から trnUnchainedCommit メソッドまたは trnUnchainedRollback メソッドを実行するまでの処理区間外である。

注※4

連鎖型 RPC でも，初回の rpcCall メソッドでは問い合わせ間隔をチェックします。

●dcnotifyreshost=Y | N　～《N》

OpenTP1 システム内に Kubernetes ノードで起動する TP1/Server が存在する場合，TP1/Client/J でこのオペランドに Y を指定してください。

このオペランドに Y を指定することで，サービス要求先の TP1/Server に，dcresponsehost で指定した応答電文受信用 IP アドレスを通知します。

これによって，Kubernetes ノードを含む OpenTP1 システム内で通信が可能となります。

Y：サービス要求先の TP1/Server に応答電文受信用 IP アドレスを通知します。次の RPC 通信に対して有効になります。

- スケジューラダイレクト機能を使用した RPC（rpcCall メソッド）
- 通信先を指定した RPC（rpcCallTo メソッド）
- ネームサービスを使用した RPC（rpcCall メソッド）

N：サービス要求先の TP1/Server に，応答電文受信用 IP アドレスを通知しません。

注意事項

- この定義で Y を指定した場合，dcresponsehost の指定が必要です。

●dcresponsehost=ホスト名　～〈文字列〉

OpenTP1 システム内に Kubernetes ノードで起動する TP1/Server が存在する場合，このオペランドの指定が必要です。TP1/Client/J が応答電文を受信するために使用する，ホスト名または IP アドレスを指定します。

このオペランドを指定することで，サービス要求先の TP1/Server に応答電文受信用 IP アドレスが通知され，Kubernetes ノードを含む OpenTP1 システム内で通信が可能となります。

ホスト名に使用できる文字は先頭が英数字または-（ハイフン）で，先頭以外が英数字，-（ハイフン），または.（ピリオド）です。

ホスト名は，/etc/hosts ファイルまたは DNS などで，IP アドレスとのマッピングができなければなりません。なお，localhost または名前解決した結果が 127 で始まる IP アドレス（例：127.0.0.1）になるホスト名は指定しないでください。

ホスト名は，1 個だけ指定できます。

ホスト名または IP アドレスは，TP1/Client/J の環境によって以下を指定してください。

- TP1/Client/J が Kubernetes クラスタ外の場合
  TP1/Client/J が起動するマシンのホスト名または IP アドレス
- TP1/Client/J が Kubernetes クラスタ内の場合
  Kubernetes クラスタ内のどれかの Kubernetes ノードのホスト名または IP アドレス

注意事項

- ホスト名に 256 文字以上指定した場合，指定した文字列の先頭から 255 文字までをノード名として扱います。
- この指定は，dcnotifyreshost オペランドに Y を指定した場合に有効です。
- dcnotifyreshost オペランドに Y を指定した場合，このオペランドは省略できません。
- 指定したホスト名の名前解決は rpcOpen メソッド発行時に実施します。そのためホスト名にマッピングしている IP アドレスの変更は，CUP 実行プロセスを正常停止したあとに実施してください。

ホスト名または IP アドレスの変更手順

- ホスト名または IP アドレスを変更する場合，CUP 実行プロセスを正常停止したあとに実施してください。CUP が Java サーブレットで動作している場合は Web サーバを停止したあとに実施してください。

5.2.3 TP1/Client/J 環境定義を指定するときの注意事項

- TP1/Client/J 環境定義のオペランドを記述した行に，設定値以外の文字を記述しないでください。設定値以外の文字を記述した場合，rpcOpen メソッドが ErrFatalException を返す場合があります。
- TP1/Client/J は，TP1/Client/J 環境定義のオペランドで始まる行（行頭の空白およびタブは無視されます）を定義解析対象とします。TP1/Client/J 環境定義のオペランド以外の文字列が指定された行は，すべてコメント行とみなされます。
- 同一マシンで CUP を複数実行する場合，CUP ごとに TP1/Client/J 環境定義を設定してください。次に示す TP1/Client/J 環境定義は，プロセス単位，またはスレッド単位で CUP ごとに異なる値を指定する必要があります。
  - dcrcvport
  - dccltcuprcvport
  - dcerrtracepath
  - dcmethodtracepath

- dcuaptracepath
- dcdatatracepath

6 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。

6.1　トレース情報の採取

TP1/Client/J のトラブルシュート機能で，次のトレースを取得できます。

- UAP トレース
- データトレース
- エラートレース
- メモリトレース
- メソッドトレース
- デバッグトレース
- 性能解析トレース
- 性能検証用トレース

トラブルシュートを容易にするため，トレース情報の取得をお勧めします。障害発生時は，トレース情報を基に原因を調査してください。なお，障害発生時にトレースを取得していない場合は，トレースを取得するように設定してから，障害が発生した現象を再現して，トレース情報を取得してください。

トレース情報の詳細については，「2.11　トラブルシュート機能」を参照してください。

6.2　ネットワーク障害時の対処

- ネットワーク障害が発生しても，JDK 5.0 環境では十分な障害情報を取得できません。Java 環境以外のネットワークツールを使用して障害の原因を調査することをお勧めします。
- ネームサービスを使用した RPC では，CUP と通信先 TP1/Server 間で，NAT(Network Address Translator：アドレス変換)をしている場合に通信障害が発生することがあります。これを回避するためには，TP1/Client/J 環境定義に dccltnammlthost=Y を指定してください。

6.3　タイマ設定値の妥当性

TCP/IP 通信を使用したシステムでは，TCP/IP の制限から，障害の発生をリアルタイムには検出できません。このため，障害検出の契機は時間監視のタイムアウト時です。OpenTP1 システムでは，処理ごとに時間監視を設定できるので，各設定値には最適な値を設定してください。

rap サーバへのコネクション確立中，rap サーバ側でシステムダウンなどの障害があった場合，Java ではコネクションが切断されたことを検知できません。この場合，setDcwatchtim メソッドで指定した応答待ち時間を満了するまで待ったあと，ErrTimedOutException 例外でエラーリターンします。

6.4　その他の障害の対処

障害の原因が，TP1/Server 側の rap サーバおよびサービス要求先の TP1/Server の障害であったり，定義値の誤りであったりすることも考えられます。この場合，TP1/Server のログファイルなどを参照して調査してください。

付録

付録 A　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。

- API，定義およびコマンドの追加と削除
- 動作の変更
- API，定義およびコマンドのデフォルト値の変更

付録 A.1　07-50 での変更点

TP1/Client/J 07-50 での API，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 A–1　TP1/Client/J 07-50 での API，定義およびコマンドの追加と削除

種別	分類	内容
追加	API	なし
	定義	TP1/Client/J 環境定義 • dccltsendbuffsize オペランド • dccltrecvbuffsize オペランド • dcinquiretimecheck オペランド
	コマンド	なし
削除	なし	

TP1/Client/J 07-50 での動作の変更点を次の表に示します。

表 A–2　TP1/Client/J 07-50 での動作の変更点

分類	内容
API	なし
定義	TP1/Client 環境定義 dccltinquiretime オペランドに 0 以外を指定し，オートコネクトモードで rpcCall メソッドを使用して RPC を実行した場合，CUP 実行プロセスでも問い合わせ間隔最大時間を監視するように変更
コマンド	なし
メッセージ	なし
その他	なし

TP1/Client/J 07-50 での API，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 A.2　07-03 での変更点

TP1/Client/J 07-03 での API，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 A–3　TP1/Client/J 07-03 での API，定義およびコマンドの追加と削除

種別	分類	内容
追加	API	TP1Client クラス • setRpcServicePrio メソッド • getRpcServicePrio メソッド
	定義	なし
	コマンド	なし
削除	なし	

TP1/Client/J 07-03 での API，定義，およびコマンドの動作，ならびにデフォルト値の変更はありません。

付録 A.3　07-02 での変更点

TP1/Client/J 07-02 での API，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 A–4　TP1/Client/J 07-02 での API，定義およびコマンドの追加

種別	分類	内容
追加	API	TP1Client クラス • acceptNotification メソッド • cancelNotification メソッド • openNotification メソッド • acceptNotificationChained メソッド • closeNotification メソッド
		ErrVersionException クラス
		ErrAcceptCanceledException クラス
	定義	TP1/Client/J 環境定義 • dcscdmulti オペランド • dcscdmulticount オペランド • dccltcupsndhost オペランド • dccltcuprcvport オペランド • dcclttrcmplmttm オペランド
	コマンド	なし
削除	なし	

TP1/Client/J 07-02 での動作の変更点を次の表に示します。

表 A–5　TP1/Client/J 07-02 での動作の変更点

分類	内容
API	setUapTraceMode メソッド，setErrorTraceMode メソッド，setMethodTraceMode メソッド，setDataTraceMode メソッドで，UAP トレースを取得するように変更
定義	なし
コマンド	なし
その他	リモート API 機能を使用した RPC で，TP1/Server の性能検証用トレースに TP1/Client/J のインスタンスごとに一意である識別情報を出力するように変更
	UAP トレースに TP1/Client/J のインスタンスごとに一意である識別情報を出力するように変更

TP1/Client/J 07-02 での API，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 A.4　07-01 での変更点

TP1/Client/J 07-01 での API，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 A–6　TP1/Client/J 07-01 での API，定義およびコマンドの追加

種別	分類	内容
追加	API	TP1Client クラス • setConnectInformation メソッド
	定義	TP1/Client/J 環境定義 • dccltdatacomp オペランド • dccltconnectinf オペランド
	コマンド	なし
削除	なし	

TP1/Client/J 07-01 での動作の変更点を次の表に示します。

表 A–7　TP1/Client/J 07-01 での動作の変更点

分類	内容
API	なし
定義	なし
コマンド	なし
その他	TP1/Client/J から TP1/Server への接続時，自システムの自動割当てポートの不足によるエラー（java.net.BindException 例外）が発生した場合に，リトライ処理を行うように変更

分類	内容
その他	SPP から応答電文受信処理中に一時クローズ要求電文を受信した場合，その一時クローズ要求電文を破棄して本来の応答電文を受信し，なおも受信待ちを不当に行い，ErrClientTimedOutException 例外が発生する問題を修正
	XA リソースサービス機能の使用時に TP1/Client/J から不正な ErrRMErrException 例外がスローされないように修正

TP1/Client/J 07-01 での API，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 A.5　07-00 での変更点

TP1/Client/J 07-00 での API，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 A–8　TP1/Client/J 07-00 での API，定義およびコマンドの追加

種別	分類	内容
追加	API	なし
	定義	TP1/Client/J 環境定義 • dccltnammlthost オペランド
	コマンド	なし
削除	なし	

TP1/Client/J 07-00 での動作の変更点を次の表に示します。

表 A–9　TP1/Client/J 07-00 での動作の変更点

分類	内容
API	なし
定義	なし
コマンド	なし
その他	TP1/Client/J がサポートする JDK のバージョンを，Java(TM)2 Software Development Kit, Standard Edition, Version 5.0（JDK 5.0）以降に変更

TP1/Client/J 07-00 での API，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 B　TP1/Client/J が出力するファイル一覧

TP1/Client/J が出力するファイルの一覧を次の表に示します。各ファイルの詳細な説明については，「表 B-2　TP1/Client/J が出力するファイルの説明」を参照してください。

表 B-1　TP1/Client/J が出力するファイル一覧

項番	ファイル種別	ファイル名またはディレクトリ名	バージョン	タイプ	ファイル形式	取得タイミング	削除可否
1	デバッグトレースファイル	Java VM を実行するユーザのホームディレクトリ /TP1clientJ/ dcCltXXXXXXXXXXXX.dmp （XXXXXXXXXXXX：タイムスタンプ）	初期	M	テキスト	TP1/Client/J が提供する API が例外を返したとき	△
2	データトレースファイル	setDataTraceMode メソッドの TrcPath 引数で指定したディレクトリ，または TP1/Client/J 環境定義の dcdatatracepath オペランドで指定したディレクトリ /dcdat1.trc，dcdat2.trc	初期	C，E，H	テキスト	CUP がメッセージの送受信をしたとき	△
3	エラートレースファイル	setErrorTraceMode メソッドの TrcPath 引数で指定したディレクトリ，または TP1/Client/J 環境定義の dcerrtracepath オペランドで指定したディレクトリ /dcerr1.trc，dcerr2.trc	初期	C，E，H	テキスト	CUP から発行されたメソッドの実行中に例外を検出したとき	△
4	メソッドトレースファイル	setMethodTraceMode メソッドの TrcPath 引数で指定したディレクトリ，または TP1/Client/J 環境定義の dcmethodtrace オペランドで指定したディレクトリ /dcmtd1.trc，dcmtd2.trc	初期	C，E，H	テキスト	TP1Client クラスの内部メソッドの開始時とリターン時	△
5	UAP トレースファイル	setUapTraceMode メソッドの TrcPath 引数で指定したディレクトリ，または TP1/Client/J 環境定義の dcuaptracepath オペランドで指定したディレクトリ /dcuap1.trc，dcuap2.trc	初期	C，E，H	テキスト	TP1/Client/J が提供するメソッドの開始時とリターン時	△

（凡例）

C：ラウンドロビン（バックアップ取得機能がないタイプ）

E：ラウンドロビン（一定量に達した直後の出力で，新しいファイルに切り替わるタイプ）

H：ラウンドロビン（切り替わった先のファイルのデータを，削除してから先頭から書き込むタイプ）

M：ファイルの出力時に，該当するファイルの総数が定義で指定したファイル数を超える場合，更新日時の最も古いファイルを削除してから新しくファイルを作成するタイプ

△：削除してはいけません。ただし，障害調査が不要であれば，ユーザ判断で削除できます。

TP1/Client/J が出力するファイルの説明を次の表に示します。

表 B-2　TP1/Client/J が出力するファイルの説明

項番	ファイル種別	関連する定義	サイズ	最大ファイル数	説明
1	デバッグトレースファイル	＜ TP1/Client/J 環境定義＞ • dccltdbgtrcfilecount デバッグトレースファイルの最大ファイル数	最大 1 メガバイト	dccltdbgtrcfilecount で指定したファイル数※	障害調査用の保守資料です。内部メソッドの動作情報（トレース取得日時，実行スレッド名，インスタンス ID，出入り口種別，デバッグトレース情報）を出力します。 デバッグトレースファイルを出力する時点で，デバッグトレースファイルの総数が dccltdbgtrcfilecount オペランドで指定したファイル数を超える場合は，更新日時の最も古いファイルを削除します。
2	データトレースファイル	＜ TP1/Client/J 環境定義＞ • dcdatatrace 取得有無の指定 • dcdatatracesize ファイルサイズ • dcdatatracepath ファイル出力先ディレクトリ	dcdatatracesize オペランドの指定値	2 世代	障害調査用の保守資料です。CUP と TP1/Server 間の送受信メッセージの内容を出力します。 dcdatatrace パラメタに Y を指定した場合に出力します（デフォルトは出力しません）。
3	エラートレースファイル	＜ TP1/Client/J 環境定義＞ • dcerrtrace 取得有無の指定 • dcerrtracesize ファイルサイズ • dcerrtracepath ファイル出力先ディレクトリ	dcerrtracesize オペランドの指定値	2 世代	障害発生時のエラー情報や通信先情報を出力します。dcerrtrace パラメタに Y を指定した場合に出力します（デフォルトは出力しません）。
4	メソッドトレースファイル	＜ TP1/Client/J 環境定義＞ • dcmethodtrace 取得有無の指定 • dcmethodtracesize ファイルサイズ • dcmethodtracepath ファイル出力先ディレクトリ	dcmethodtracesize オペランドの指定値	2 世代	障害調査用の保守資料です。TP1Client クラスの内部メソッドの実行順序や実行時刻を出力します。dcmethodtrace パラメタに Y を指定した場合に出力します（デフォルトは出力しません）。
5	UAP トレースファイル	＜ TP1/Client/J 環境定義＞ • dcuaptrace 取得有無の指定 • dcuaptracesize ファイルサイズ • dcuaptracepath ファイル出力先ディレクトリ	dcuaptracesize オペランドの指定値	2 世代	CUP が発行した TP1/Client/J のメソッドの情報（指定情報，リターン情報）を出力します。 dcuaptrace パラメタに Y を指定した場合に出力します（デフォルトは出力しません）。

注※

TP1/Client/J 環境定義の dccltdbgtrcfilecount オペランドの指定値が 0 の場合は，ファイル数の制限はありません。

索引

A

C

D

E

G

O

P

R

S

こ

さ

し

す

せ

そ

た

つ

て

と

ね

の

ひ

ふ

ま

め

ゆ

り

れ

ろ