HITACHI
Inspire the Next

OpenTP1 Version 7

高速トランザクション処理基盤

TP1/EE/Extended Transaction Controller **使用の手引**

解説・手引・文法書

3000-3-F54-20

■対象製品

P-9W64-9211 uCosminexus TP1/EE/Extended Transaction Controller 01-03（適用 OS：Red Hat Enterprise Linux 5 (AMD/Intel 64)，Red Hat Enterprise Linux 5 Advanced Platform (AMD/Intel 64)，Red Hat Enterprise Linux Server 6(64-bit x86_64)）
P-8264-9211 uCosminexus TP1/EE/Extended Transaction Controller 01-03（適用 OS：Red Hat Enterprise Linux Server 6(64-bit x86_64)，Red Hat Enterprise Linux Server 7(64-bit x86_64)）
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。
これらの製品は，ISO9001 および TickIT の認証を受けた品質マネジメントシステムで開発されました。

■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

2015 年 10 月 3000-3-F54-20

■著作権

変更内容

変更内容（3000-3-F54-20） uCosminexus TP1/EE/Extended Transaction Controller　01-03

追加・変更内容	変更個所
RPC 応答の送信タイミング変更機能をサポートした。	2.5.4
読み出しできる後続メッセージの条件について，説明を追加した。	2.6.3(1)
次のオペランドの説明を変更した。 メモリ関連定義 mem_uibf_extend_num mem_obf_extend_num mem_uobf_extend_num	5.3　XTC サービス定義 • メモリ関連定義 mem_uibf_extend_num mem_obf_extend_num mem_uobf_extend_num
次のオペランドの説明を変更した。 RPC 関連定義 rpc_tcp_communication_use	5.3　XTC サービス定義 • RPC 関連定義 rpc_tcp_communication_use
次のオペランドのデフォルト値を変更した。 RPC 関連定義 rpc_recv_permanence	5.3　XTC サービス定義 • RPC 関連定義 rpc_recv_permanence
次のオペランドを追加した。 ユーザサービス関連定義 errtrn_serialize	5.3　XTC サービス定義 • ユーザサービス関連定義 errtrn_serialize
次のオペランドを追加した。 トランザクション関連定義 trn_cl_rpc_reply_timing	5.3　XTC サービス定義 • トランザクション関連定義 trn_cl_rpc_reply_timing
次のオペランドを削除した。 高速メッセージ送信関連定義 mch_send_err_rollback	5.3　XTC サービス定義 • 高速メッセージ送信関連定義 mch_send_err_rollback
次のオペランドを追加した。 クラスタ連携関連定義 mch_clreq_rcvthd	5.3　XTC サービス定義 • クラスタ連携関連定義 mch_clreq_rcvthd 7.7.2
次のオペランドの説明を変更した。 HA モニタ関連定義 ham_stop_msg_interval_time	5.3　XTC サービス定義 • HA モニタ関連定義 ham_stop_msg_interval_time
次のオペランドの説明を変更した。 UDP グループ情報関連定義（コマンド形式） clgrpdef（クラスタグループ情報定義）	5.3　XTC サービス定義 • UDP グループ情報関連定義（コマンド形式） clgrpdef（クラスタグループ情報定義）
eehamls コマンドの説明を変更した。	6.4　運用コマンド eehamls（系の状態表示）
系の状態表示の説明を変更した。	7.3
実行系孤立状態になった場合の DB 管理者の対処の説明を変更した。	7.4

単なる誤字・脱字などはお断りなく訂正しました。

はじめに

このマニュアルは，次に示すプログラムプロダクトのシステム構築方法および運用方法について説明したものです。

- P-9W64-9211 uCosminexus TP1/EE/Extended Transaction Controller
- P-8264-9211 uCosminexus TP1/EE/Extended Transaction Controller

以降，このマニュアルでは，「uCosminexus TP1/EE/Extended Transaction Controller」を「XTC」と表記します。

■対象読者

TP1 キャッシュ機能（インメモリデータ処理技術を中心とした高速トランザクション処理機能）を使用したシステムを構築・運用するシステム管理者，システム設計者，プログラマ，およびオペレータの方を対象としています。

なお，このマニュアルは，次に示す知識があることを前提にして説明されています。

- Linux のシステム管理の基礎的な知識
- オンラインシステムの基礎的な知識
- TP1/Server Base および TP1/Server Base Enterprise Option の基礎的な知識
- HA モニタによる系切り替え機能の基礎的な知識

■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 キャッシュ機能の概要とシステム構成，および特長について説明しています。

第 2 章　XTC の機能

XTC の機能について説明しています。

第 3 章　XTC が扱うファイルの種類と容量見積もり

XTC が扱うファイルの種類と容量見積もりについて説明しています。

第 4 章　環境設定

TP1 キャッシュ機能使用時の環境設定とその手順について説明しています。

第 5 章　システム定義

TP1 キャッシュ機能使用時のシステム定義について説明しています。

第 6 章　運用

XTC の開始・終了方法，および運用コマンドについて説明しています。

第 7 章　CL 連携による系切り替え機能

CL 連携による系切り替え機能の仕組みと運用方法について説明しています。

第 8 章　障害時の運用

障害時の運用について説明しています。

付録 A　インストールディレクトリとファイル

XTC のインストールディレクトリの構成とファイルについて説明しています。

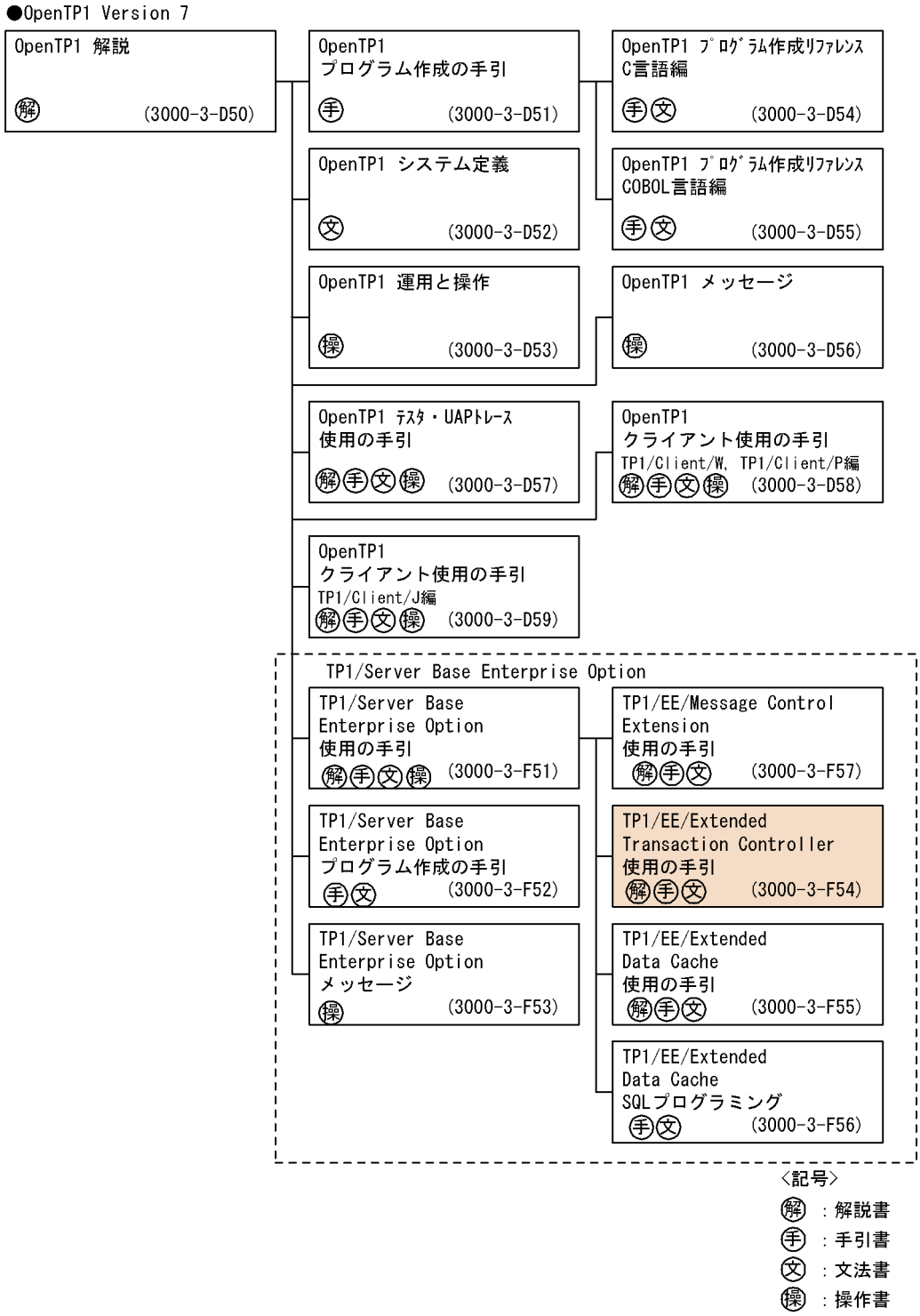
付録 B　各バージョンの変更内容

各バージョンの変更内容について説明しています。

付録 C　用語解説

このマニュアルで使用している用語について説明しています。

■関連マニュアル


●OpenTP1 Version 7
OpenTP1 解説
(3000-3-D50)
OpenTP1
プログラム作成の手引
(3000-3-D51)
OpenTP1 プログラム作成リファレンス
C言語編
(3000-3-D54)
OpenTP1 システム定義
(3000-3-D52)
OpenTP1 プログラム作成リファレンス
COBOL言語編
(3000-3-D55)
OpenTP1 運用と操作
(3000-3-D53)
OpenTP1 メッセージ
(3000-3-D56)
OpenTP1 テスタ・UAPトレース
使用の手引
(3000-3-D57)
OpenTP1
クライアント使用の手引
TP1/Client/W, TP1/Client/P編
(3000-3-D58)
OpenTP1
クライアント使用の手引
TP1/Client/J編
(3000-3-D59)
TP1/Server Base Enterprise Option
TP1/Server Base
Enterprise Option
使用の手引
(3000-3-F51)
TP1/EE/Message Control
Extension
使用の手引
(3000-3-F57)
TP1/Server Base
Enterprise Option
プログラム作成の手引
(3000-3-F52)
TP1/EE/Extended
Transaction Controller
使用の手引
(3000-3-F54)
TP1/Server Base
Enterprise Option
メッセージ
(3000-3-F53)
TP1/EE/Extended
Data Cache
使用の手引
(3000-3-F55)
TP1/EE/Extended
Data Cache
SQLプログラミング
(3000-3-F56)
〈記号〉
解 ：解説書
手 ：手引書
文 ：文法書
操 ：操作書

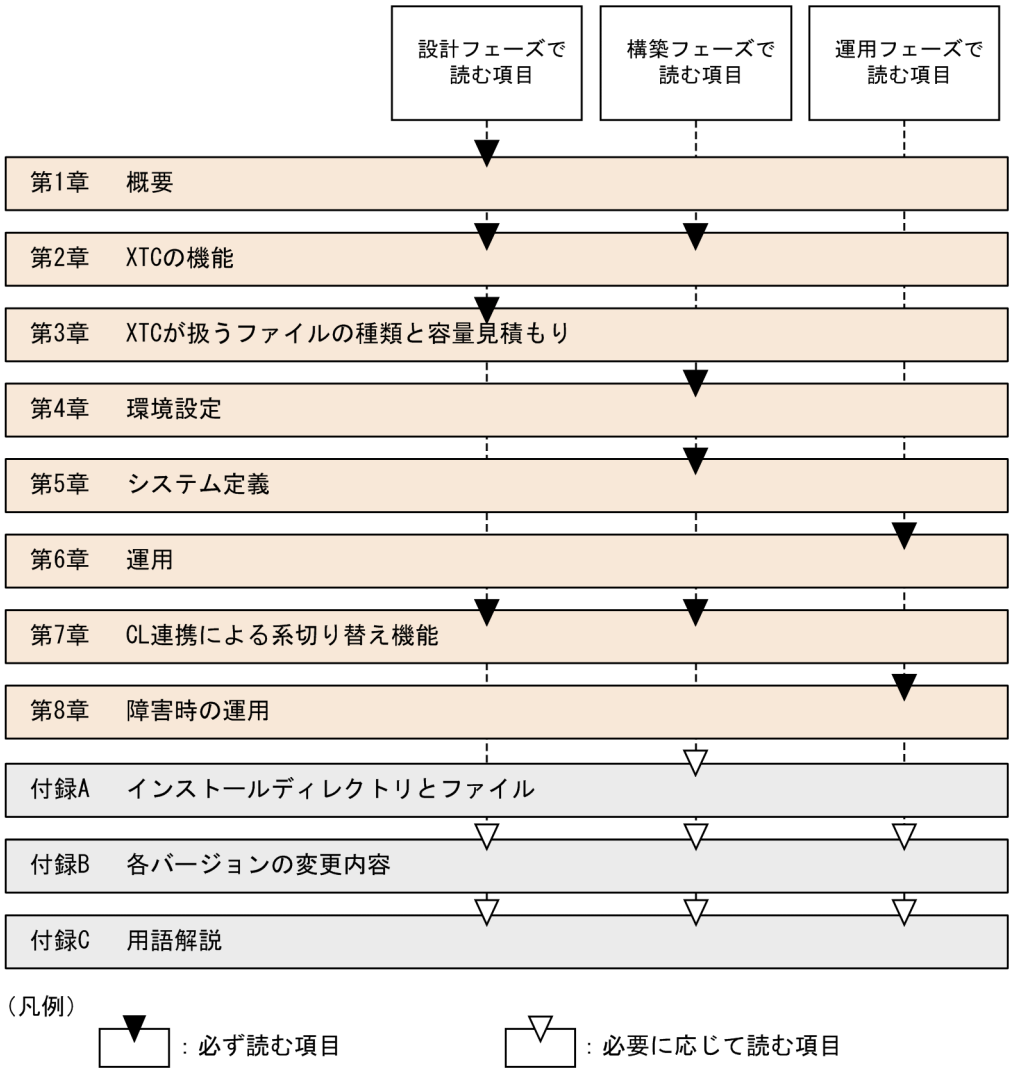
●関連製品

COBOL2002 使用の手引 手引編 ㊫㊪ (3000-3-D42)	COBOL2002 使用の手引 操作編 ㊭ (3000-3-D43)	
HiRDB Version 8 システム導入・設計ガイド （UNIX(R)用） ㊫㊪㊭ (3000-6-352)	HiRDB Version 8 システム定義 （UNIX(R)用） ㊬ (3000-6-353)	HiRDB Version 8 コマンドリファレンス （UNIX(R)用） ㊬ (3000-6-355)
HiRDB Version 8 UAP開発ガイド ㊫㊪ (3020-6-356)	HiRDB Version 8 SQLリファレンス ㊬ (3020-6-357)	HiRDB Version 8 メッセージ ㊭ (3020-6-358)
HiRDB Version 8 XDM/RD E2接続機能 ㊫㊪㊬ (3020-6-365)		
高信頼化システム監視機能 HAモニタ Linux(R)編 ㊫㊪㊭ (3000-9-132)	高信頼化システム監視機能 HAモニタ メッセージ ㊭ (3000-9-134)	

〈記号〉

㊫ ：解説書
㊪ ：手引書
㊬ ：文法書
㊭ ：操作書

■読書手順



■図中で使用する記号

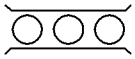
このマニュアルの図中で使用する記号を，次のように定義します。

●端末

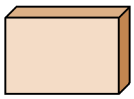
●入出力の動作

●ファイル，メモリ

●キュー

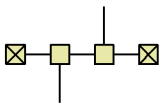


●プログラム



●コネクション

●ネットワーク（LAN）



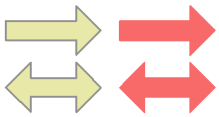
●情報

●エラー，障害

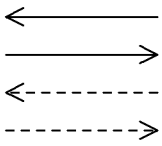
●プログラムの流れ

●工程，作業項目の流れ

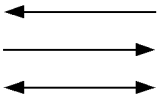
●データの流れ



●制御の流れ



●その他の流れ



■文法の記号

(1) 文法記述記号

システム定義，コマンドのオプションおよび引数の指定方法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意味しています。 (例)〔-g サービスグループ名〕 　この場合，-g オプションの指定を省略できます。
…	この記号で示す直前の項目を繰り返し指定できることを意味しています。 (例) eetrbqueed キューダンプファイル名 〔キューダンプファイル名〕… 　この場合，eetrbqueed コマンドにキューダンプファイル名を複数指定できます。
|	この記号で区切られた項目は選択できることを意味しています。 (例)〔Y | N〕 　この場合，Y または N のどちらかを指定できます。

文法記述記号	意味
{{　　}}	この記号で囲まれた複数の項目が一つの繰り返し項目の単位であることを示します。 （例）{{eemchsrvdef -g サービスグループ名}} 　これは，次のように繰り返し指定できることを示します。 　eemchsrvdef -g サービスグループ名 　eemchsrvdef -g サービスグループ名
{　　}	この記号で囲まれている複数の項目のうち，一つだけ選択できることを意味しています。 （例）{act ｜ dct ｜ all} 　この場合，act，dct，all のうち，どれか一つを指定できます。
_ （下線）	この記号で示す項目は，該当オペランド，またはオプションを省略した場合に仮定される値を意味しています。 （例）trn_transactional_rpcless_use ＝ Y ｜ N 　この場合，trn_transactional_rpcless_use オペランドを省略すると，N が仮定されます。

(2) 属性表示記号

システム定義，およびコマンドに指定できる値の範囲を説明する記号です。

属性表示記号	意味
～	この記号のあとに指定値の属性を示します。
《　》	指定を省略したときに仮定される値を示します。
〈　〉	指定値の構文要素を示します。
((　))	指定値の指定範囲を示します。

(3) 構文要素記号

システム定義，およびコマンドに指定できる文字または値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）と _（アンダスコア）
〈英数字〉	英字と数字（0 ～ 9）
〈符号なし整数〉	数字列（0 ～ 9）
〈16 進数〉	数字（0 ～ 9），A ～ F，および a ～ f
〈識別子〉	先頭がアルファベットの英数字列
〈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 に依存します。
〈ホスト名〉	英数字，および !，#，$，%，&，'，(，)，*，+，-，.（ピリオド），/，;，<，=，>，?，@，[，¥，]，^，_（アンダスコア），{，｜，}，~，,（コンマ） ただし，OS によって使用できる文字が異なります。詳細は，使用している OS のマニュアルを参照してください。

使用上の注意

すべて半角文字を使用してください。

■計算式の記号

記号	意味
↑↑	計算結果の値を小数点以下で切り上げます。 （例） ↑34 ÷ 3 ↑ この場合，計算結果は，12 となります。
↓↓	計算結果の値を小数点以下で切り捨てます。 （例） ↓34 ÷ 3 ↓ この場合，計算結果は，11 となります。
MAX	括弧内の項目のうち，最も大きい値を選びます。 （例）MAX（3，8+2，2 × 2） この場合，計算結果は 10 となります。
MIN	括弧内の項目のうち，最も小さい値を選びます。 （例）MIN（3，8+2，2 × 2） この場合，計算結果は 3 となります。

■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	このマニュアルでの表記
Linux(R)	Linux
Red Hat Enterprise Linux 5 (AMD/Intel 64)	
Red Hat Enterprise Linux 5 (Intel Itanium)	
Red Hat Enterprise Linux 5 (x86)	
Red Hat Enterprise Linux 5 Advanced Platform (AMD/Intel 64)	
Red Hat Enterprise Linux 5 Advanced Platform (Intel Itanium)	
Red Hat Enterprise Linux 5 Advanced Platform (x86)	
Red Hat Enterprise Linux Server 6(64-bit x86_64)	
Red Hat Enterprise Linux Server 7(64-bit x86_64)	
uCosminexus TP1/EE/Extended Data Cache	XDB
uCosminexus TP1/EE/Extended Transaction Controller	XTC
uCosminexus TP1/EE/Message Control Extension	MCP
uCosminexus TP1/Server Base	TP1/Server Base
uCosminexus TP1/Server Base Enterprise Option	TP1/EE
uCosminexus TP1/Server Base Enterprise Option(64)	

■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
ASCII	American Standard Code for Information Interchange
CPU	Central Processing Unit
DB	Database
DNS	Domain Name System
HA	High Availability
I/O	Input/Output
ID	Identifier
IP	Internet Protocol
IT	Information Technology
LAN	Local Area Network
MP	Management Processor
MTU	Maximum Transmission Unit
OS	Operating System
RPC	Remote Procedure Call
SPP	Service Providing Program
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
TTL	Time To Live
UAP	User Application Program
UDP	User Datagram Protocol
UOC	User Own Coding
XA	Extended Architecture

■ 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

1 概要

この章では，TP1 キャッシュ機能の概要とシステム構成，および特長について説明します。

1.1 TP1 キャッシュ機能とは

TP1 キャッシュ機能とは，インメモリデータ処理技術を中心とした高速トランザクション処理機能の総称です。インメモリデータ処理技術とは，オンライントランザクション処理システムで処理するデータをすべてメモリ上に配置し，高速に処理するための技術です。

TP1 キャッシュ機能は，TP1/Server Base Enterprise Option（TP1/EE）を前提とする機能です。高速メッセージ処理基盤である uCosminexus TP1/EE/Extended Transaction Controller（XTC）と，高速データ処理基盤である uCosminexus TP1/EE/Extended Data Cache（XDB）の 2 製品から構成されます。

TP1 キャッシュ機能を使用することで，トランザクション処理の高速化と，高信頼性，高可用性を同時に実現するオンライントランザクション処理システムを構築できます。

ここでは，高速トランザクション処理を取り巻く環境について説明したあと，トランザクション処理の高速化と高信頼性，高可用性の共存について説明します。

1.1.1 高速トランザクション処理を取り巻く環境

近年，IT 技術の進歩やインターネットの発展に伴い，オンライントランザクション処理システムで処理されるトランザクション量は増加の一途をたどっています。金融・証券系システムでは，オンライントレードの普及とそれに伴う株取引の小口化やアルゴリズム取引※の浸透によって，今後さらに，トランザクション量の大幅な増加が予想されています。

一方，企業ビジネスでは，ミリ秒レベルといわれる超高速な応答性能を持つシステム基盤が求められています。

このような背景から，高信頼性，高可用性に加えて，大量のデータを高速に処理でき，かつ，将来のトランザクション量の増大にも耐えうる，高速トランザクション処理基盤が必要不可欠なものとなってきています。

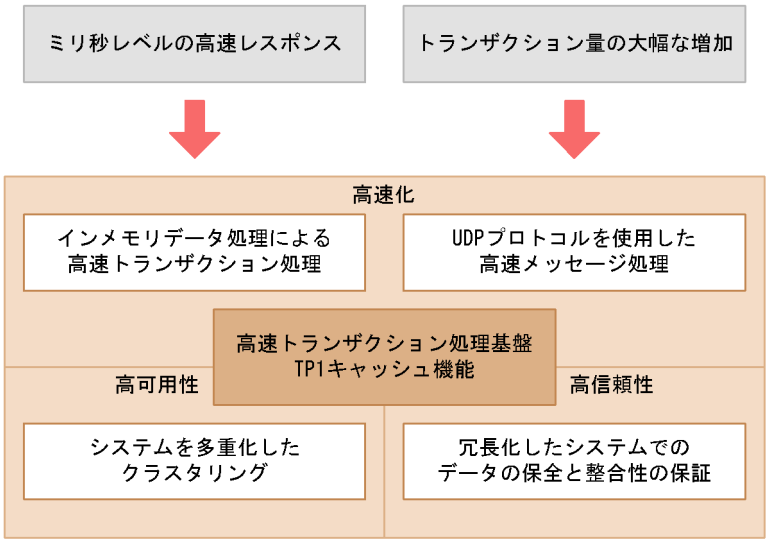
TP1 キャッシュ機能では，オンラインシステムで処理するデータをすべてメモリ上に配置し，データ操作に伴う磁気ディスク装置に対する入出力処理をなくすことによって，これらの要件を満たす高速トランザクション処理基盤を提供します。

注※

コンピュータシステムが株価や出来高などに応じて，自動的に株式売買注文のタイミングや数量を決めて注文を行う取引のことです。

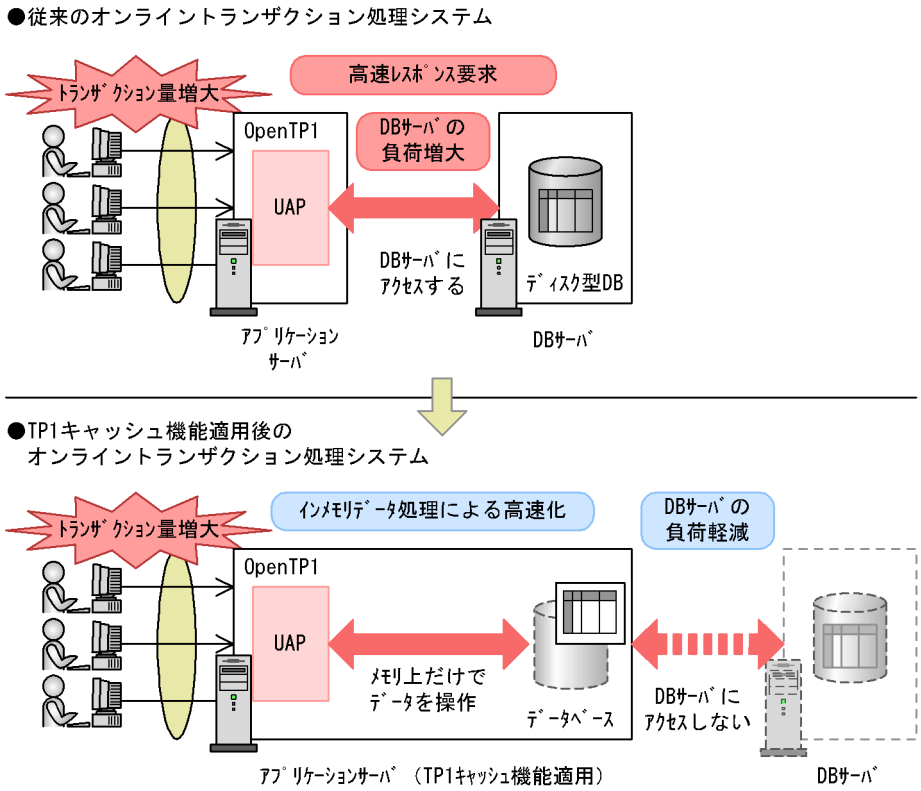
TP1 キャッシュ機能の概要を次の図に示します。

図 1-1　TP1 キャッシュ機能の概要



TP1 キャッシュ機能を適用すると，例えばトランザクション量が増加しても，インメモリ処理によって高速な応答性能を実現できます。また，アプリケーションサーバと DB サーバ間のアクセスがなくなるため，DB サーバの負荷も軽減されます。TP1 キャッシュ機能の適用イメージを次の図に示します。

図 1-2 TP1 キャッシュ機能の適用イメージ



1.1.2 トランザクション処理の高速化と高信頼性，高可用性の共存

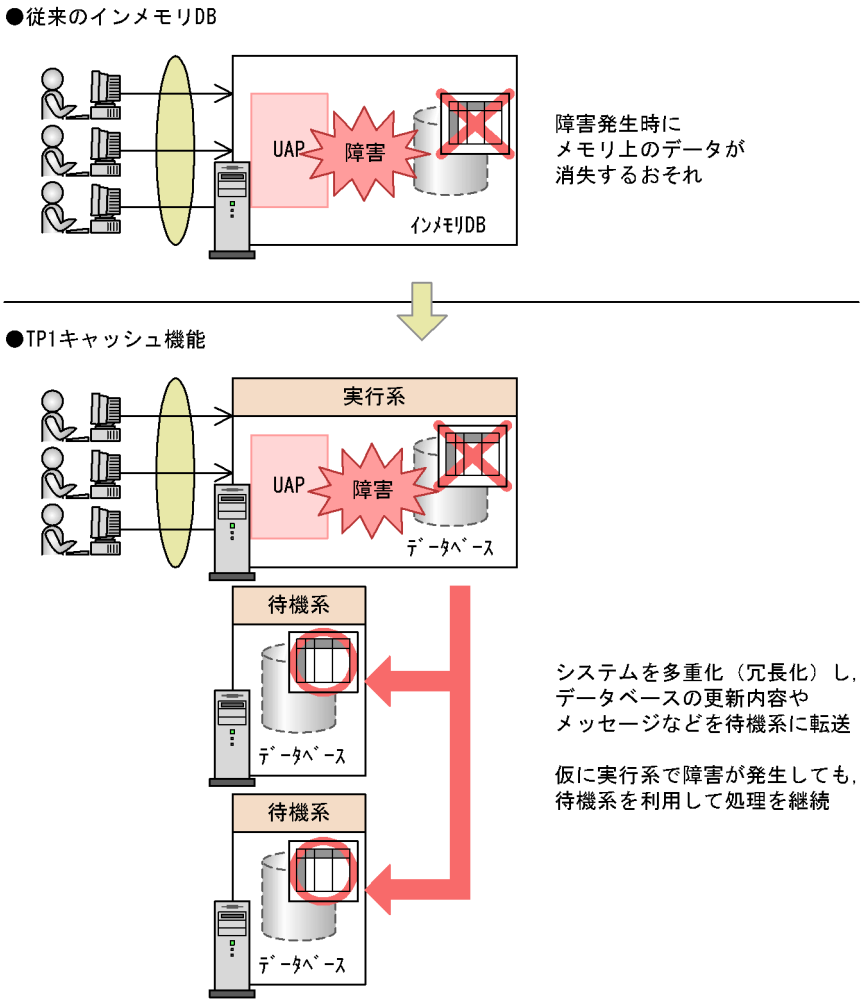
メモリの大容量化や，64 ビット OS の普及といった背景もあり，操作対象のデータをすべてメモリ上に保持してデータ処理を行うインメモリデータ処理技術は，トランザクション処理を高速化する手段として注目されつつあります。磁気ディスク装置にデータを格納するディスク型 DB とは異なり，メモリ上にデータを配置，処理するデータベース（インメモリ DB）では，磁気ディスク装置に対する入出力処理が発生しないため，トランザクション処理を高速化できるからです。

一方，インメモリ DB には，その特性上，予期しないシステムダウンなどの障害が発生した場合に，データの消失をどのようにして防ぐかという大きな課題があります。特に，ミッションクリティカルなシステムでは，インメモリ DB による高速化に加え，高信頼性と高可用性が必須要件となります。

TP1 キャッシュ機能では，システムを多重化してクラスタリングを行うことで，信頼性と可用性を確保します。多重化されたシステムが協調して動作することによって，業務処理を行っているシステム（実行系）に障害が発生してもデータやメッセージを保持し

たまま，バックアップシステム（待機系）で業務処理を継続できます。TP1 キャッシュ機能でのデータ保持の概要を次の図に示します。

図 1-3　TP1 キャッシュ機能でのデータ保持の概要



TP1 キャッシュ機能は，高信頼性，高可用性を兼ね備えた高速データ処理基盤として，ミッションクリティカルなシステムでも使用できます。

1.2 システム構成

ここでは，TP1 キャッシュ機能を使用する場合のシステム構成について説明します。

1.2.1 TP1 キャッシュ機能使用時のサーバ構成

TP1 キャッシュ機能を使用するシステムでは，HA サーバと CL サーバを配置します。
HA サーバと CL サーバ間はネットワーク経路を多重化することもできます。

■ HA サーバ

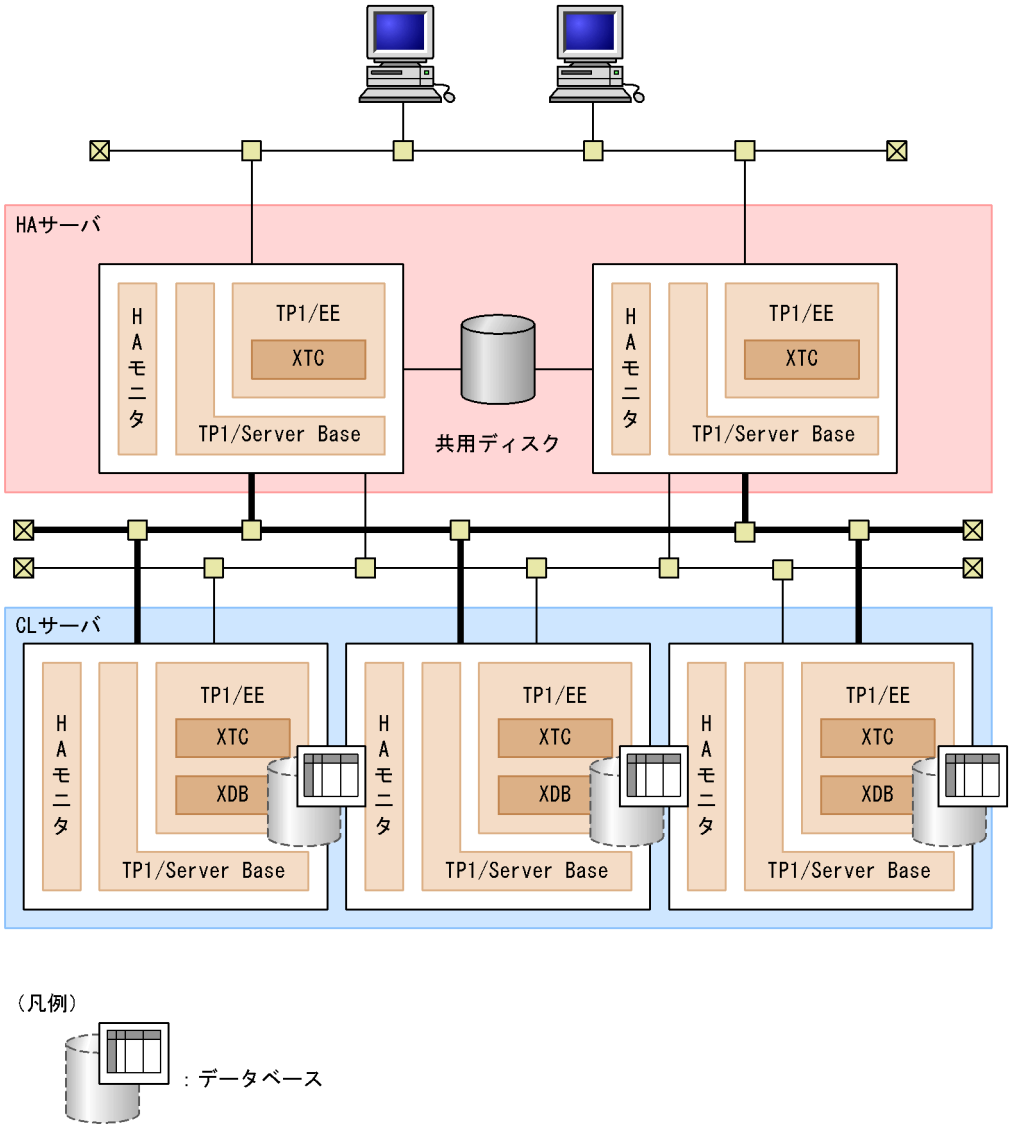
HA サーバとは，TP1 キャッシュ機能使用時に，メッセージや RPC の送信側となるサーバです。それぞれのサーバに XTC を配置します。
HA サーバは，一般的なクラスタ構成とし，実行系 1 台，待機系 1 台の 2 台で構成します。
共用ディスクなどを使用してデータの永続化を図る場合は HA サーバとします。

■ CL サーバ

CL サーバとは，TP1 キャッシュ機能使用時に，メッセージや RPC の受信側となって，高速データ処理を行うサーバです。それぞれのサーバに XTC および XDB を配置します（XDB を配置したサーバは必ず CL サーバにする必要があります）。
CL サーバは，HA モニタのマルチスタンバイ機能を使用したクラスタ構成として，実行系 1 台，待機系 2 台の 3 台で構成します。
TP1 キャッシュ機能を使用すると，CL サーバ間で連携して，実行系のデータベースの更新内容を，待機系のデータベースに逐次反映して整合性を保持します。
インメモリデータ処理を行う場合は CL サーバとします。

TP1 キャッシュ機能を使用する場合のシステム構成例を次の図に示します。

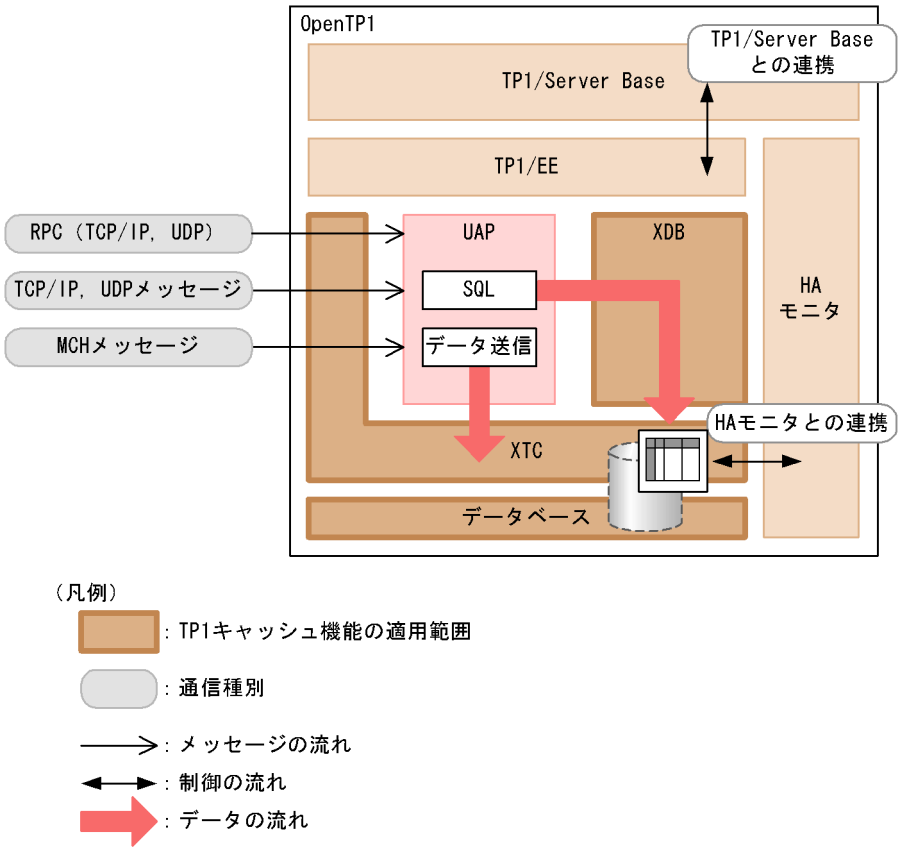
図 1-4　TP1 キャッシュ機能を使用する場合のシステム構成例



1.2.2　ソフトウェア構成

XTC と XDB は TP1/EE 下で動作します。TP1/EE と XTC，XDB の関係を次の図に示します。

図 1-5 TP1/EE と XTC，XDB の関係



TP1 キャッシュ機能での XTC と XDB の役割について，次に示します。

(1) TP1 キャッシュ機能での XTC の役割

XTC は，高速メッセージ処理基盤として，次のような処理を行います。

- UDP プロトコルを使用した高速メッセージ制御
- マルチキャストによる CL サーバの実行系，待機系への同時送信
- 実行系と待機系との間の整合性の保証
- XDB とのトランザクション連携

(a) UDP プロトコルを使用した高速メッセージ制御

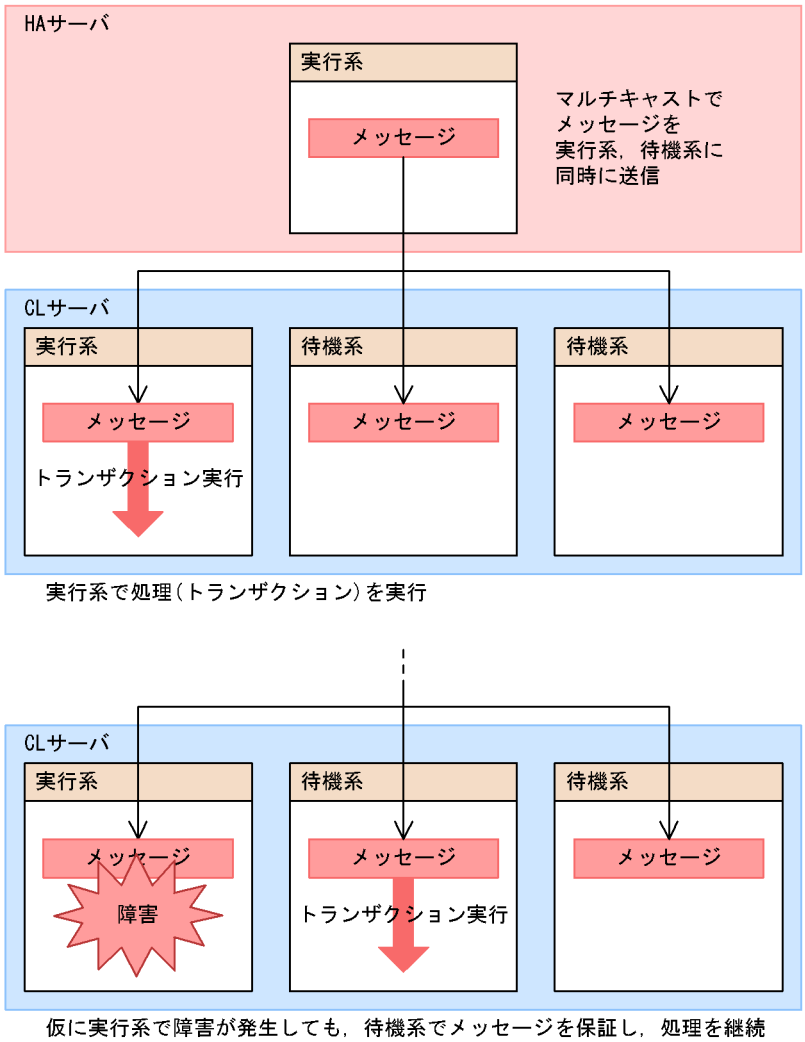
TP1/EE 間でのシステム間通信（RPC）やメッセージの送受信を，UDP プロトコルを使用した独自のインタフェースによって実行できます。これによって，高速にメッセージを送受信することができます。

(b) マルチキャストによる CL サーバの実行系，待機系への同時送信

HA サーバからの処理要求（メッセージ）を，マルチキャストで CL サーバの実行系，待

機系それぞれのサーバに同時に送信できます。処理要求をそれぞれのサーバで保証することによって，CL サーバの実行系で障害が発生しても，待機系で処理を継続できます。マルチキャストによる CL サーバの実行系，待機系への同時送信の概要を次の図に示します。

図 1-6　マルチキャストによる CL サーバの実行系，待機系への同時送信



(c) 実行系と待機系との間の整合性の保証

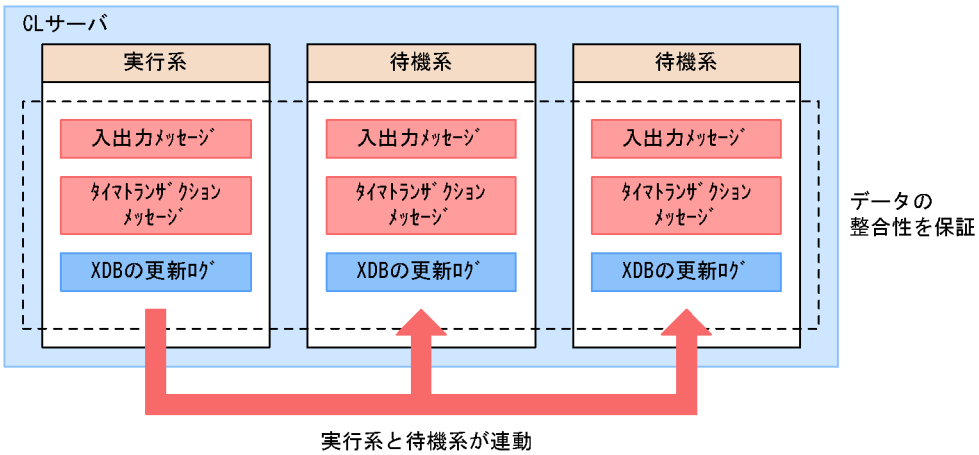
データの保全を行うため，CL サーバの実行系のデータを待機系に転送します。その際，データの送信順序を保証することによって，CL サーバの実行系と待機系との間でデータの整合性を保証します。

保全の対象となるデータには，入出力メッセージ，XDB の更新ログ，タイマトランザク

ションメッセージなどがあります。

CL サーバの実行系と待機系の整合性の保証とデータの保全を次の図に示します。

図 1-7 CL サーバの実行系と待機系の整合性の保証とデータの保全



(d) XDB とのトランザクション連携

インメモリデータ処理を実現する XDB と連携したトランザクション制御を行うことによって，高速なトランザクション処理を実現します。

(2) TP1 キャッシュ機能での XDB の役割

XDB は，高速データ処理基盤として次のような処理を行います。

- インメモリデータ処理の制御
- 更新ログによる待機系のデータベースの更新

XDB の詳細については，マニュアル「TP1/EE/Extended Data Cache 使用の手引」を参照してください。

(a) インメモリデータ処理の制御

インメモリデータ処理では，すべてのデータをメモリ上に展開します。これによって，トランザクション実行時間の大半を占めていたディスクアクセスに掛かる時間が大幅に削減され，トランザクションの処理性能が大きく向上します。

メモリ上へのデータベースの展開は，CL サーバの実行系の正常開始時に自動的に行われます（XDB が行います）。CL サーバの待機系の起動時には，実行系のデータベースの内容がコピーされて展開されます。

XDB の正常終了時には，メモリ上に展開されていたデータベースの内容が指定されたファイルに自動的に取得されます。次回のオンライン起動時には，ここで取得した内容をメモリ上に展開することができます。

(b) 更新ログによる待機系のデータベースの更新

CL サーバの実行系と待機系の整合性を保証するため，XDB では実行系のデータベースの更新内容を，待機系のデータベースに逐次反映します。

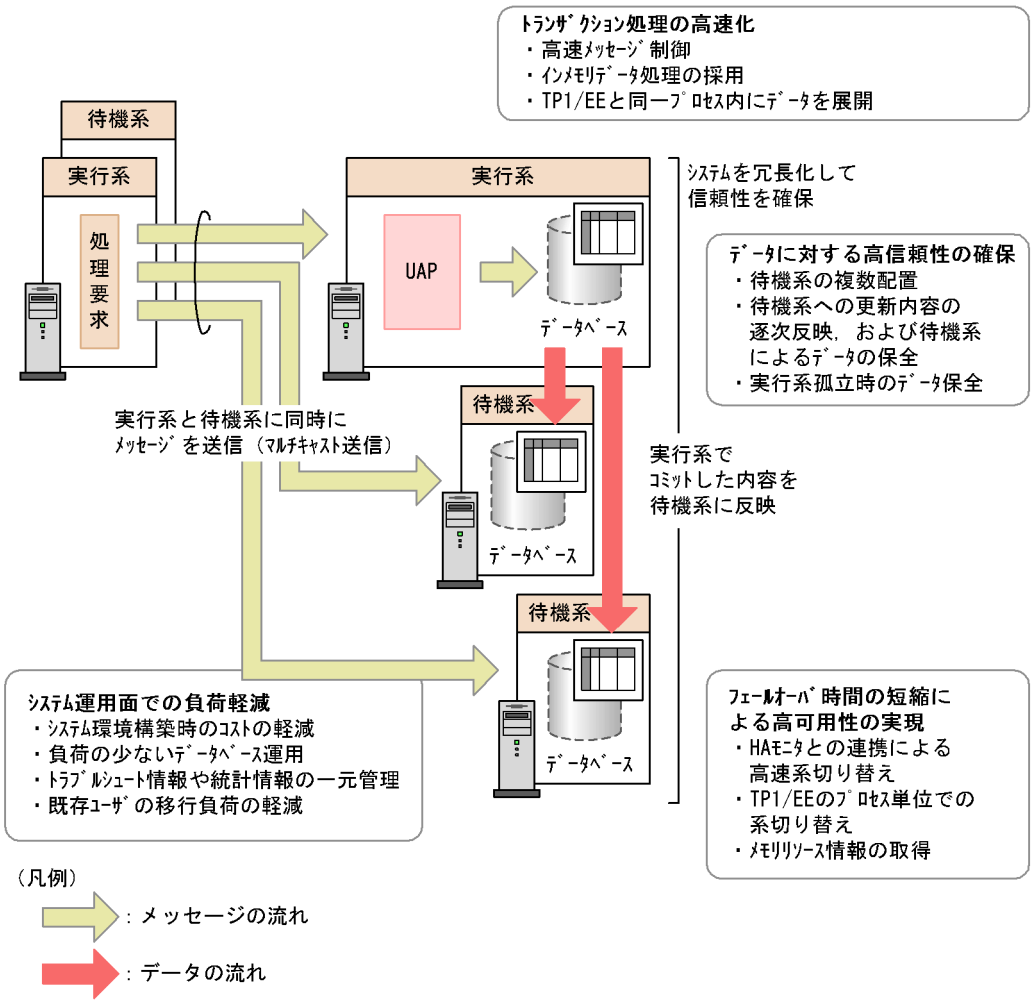
1.3 TP1 キャッシュ機能の特長

ここでは，TP1 キャッシュ機能の特長について次の観点から説明します。

- トランザクション処理の高速化
- データに対する高信頼性の確保
- フェールオーバ時間の短縮による高可用性の実現
- システム運用面での負荷軽減

TP1 キャッシュ機能の特長を次の図に示します。

図 1-8 TP1 キャッシュ機能の特長



1.3.1 トランザクション処理の高速化

TP1 キャッシュ機能では，トランザクション処理でのスループットの向上とレイテンシの低減を図るため，次のような特長があります。

- 高速メッセージ制御
- インメモリデータ処理の採用
- TP1/EE と同一プロセス内にデータを展開

(1) 高速メッセージ制御

UDP プロトコルを使用することによって，メッセージや RPC の送受信を高速化しています。その際，UDP プロトコルの欠点である，送信順序や送達確認を XTC が保証します。

(2) インメモリデータ処理の採用

インメモリデータ処理によって，磁気ディスク装置の入出力に掛かるオーバヘッドをなくし，トランザクションの処理性能を大きく向上させます。

(3) TP1/EE と同一プロセス内にデータを展開

TP1/EE と同一のプロセス内にデータを展開することによって，データアクセス時に他プロセスとの通信を発生させません。これによって，OpenTP1 と DBMS（リソースマネジャ）の間で発生する通信オーバヘッドを削減しています。

1.3.2 データに対する高信頼性の確保

TP1 キャッシュ機能では，データに対する高信頼性を確保するため，次のような特長があります。

- 待機系の複数配置
- 待機系への更新内容の逐次反映，および待機系によるデータの保全
- 実行系孤立時のデータ保全

(1) 待機系の複数配置

インメモリデータ処理の課題である障害発生時のデータの消失を防ぐため，CL サーバとして複数の待機系を配置します。CL サーバの実行系と待機系が連携して動作することによって，データベースの冗長化を図り，高い信頼性を確保します。

(2) 待機系への更新内容の逐次反映，および待機系によるデータの保全

CL サーバの実行系のデータベースの更新内容（更新ログ）は，XTC によって逐次，待機系に転送されます。待機系では更新ログを基にデータベースの追い付き処理が行われます。

更新ログを転送する際，XTC は転送順序（待機系のデータベースの更新順序）を保証し

ます。これによって，実行系と待機系の間の整合性が保証されます。

CL サーバの実行系と待機系の間の整合性が保証されることによって，待機系がバックアップの位置づけになります（待機系でデータが保全されます）。これによって，実行系で障害が発生しても，メモリ上のデータの消失を防ぐことができます。

(3) 実行系孤立時のデータ保全

障害などによって系切り替えが発生した結果，待機系がなくなり，実行系だけとなった場合（実行系が孤立した場合），XTC はデータの保全を最優先して，メモリ上のダンプ情報をファイルに出力します。

1.3.3 フェールオーバ時間の短縮による高可用性の実現

TP1 キャッシュ機能では，CL サーバの実行系で障害が発生した際のフェールオーバ（系切り替え）に掛かる時間を大幅に短縮しています。高可用性を実現する上で次のような特長があります。

- HA モニタとの連携による高速系切り替え
- TP1/EE のプロセス単位での系切り替え
- メモリリソース情報の取得

(1) HA モニタとの連携による高速系切り替え

TP1 キャッシュ機能では，XTC が HA モニタと独自のインタフェースで連携し，障害発生時の系の切り替えを高速で行います。CL サーバの実行系と待機系の整合性が保証されているため，待機系で再開始処理（実行系への追い付き処理）を実行することなく，高速で系切り替えを実行できます。

(2) TP1/EE のプロセス単位での系切り替え

TP1 キャッシュ機能では，マシン単位ではなく，TP1/EE のプロセス単位に系切り替えを行います。これによって，高速な切り替えを実現し，オンライン停止時間を極小化します。

(3) メモリリソース情報の取得

CL サーバが 2 台構成時に，実行系で障害が発生して残り 1 台になると，自動的にオンライン処理を停止します。その際，データを保持するためにメモリリソース情報（データベースの情報，キューの情報など）を取得します。

1.3.4 システム運用面での負荷軽減

TP1 キャッシュ機能を使用する場合，次に示すようなシステム運用面での負荷や，システム環境構築時のコストを軽減できます。

- システム環境構築時のコストの軽減

- 負荷の少ないデータベース運用
- トラブルシュート情報や統計情報の一元管理
- 既存ユーザの移行負荷の軽減

(1) システム環境構築時のコストの軽減

インメモリデータ処理によって，磁気ディスク装置の使用を抑えることができます。磁気ディスク装置は冗長化されていることも多いため，使用を抑えることでシステム環境構築時のコストを軽減できます。

(2) 負荷の少ないデータベース運用

インメモリデータ処理では，データベースの更新履歴（ログ）の管理や，バックアップの取得などの運用が必要ありません。また，バッファプールやチェックポイントなどのチューニングも必要ありません。そのため，データベースの運用に掛かる負荷を少なくすることができます。

(3) トラブルシュート情報や統計情報の一元管理

XDB に関するトラブルシュート情報や統計情報を，TP1/EE のトラブルシュート，統計情報として一元管理できます。これによって，OpenTP1 のインタフェースでリソースマネジャに関する情報を管理できます。

(4) 既存ユーザの移行負荷の軽減

TP1/EE の機能を使用できることから，すでに TP1/EE を使用しているユーザが TP1 キャッシュ機能を適用する際，アプリケーションや運用手順などの変更を抑えることができます。これによって，システム移行時の作業負荷やコストを軽減できます。

2 XTC の機能

この章では，XTC の機能について説明します。

2.1 XTC が備える TP1 キャッシュ機能の全体像

XTC には，次の機能があります。

- 高速通信・高速処理を実現する機能
- 高信頼性を実現する機能
- 運用性を向上させる機能

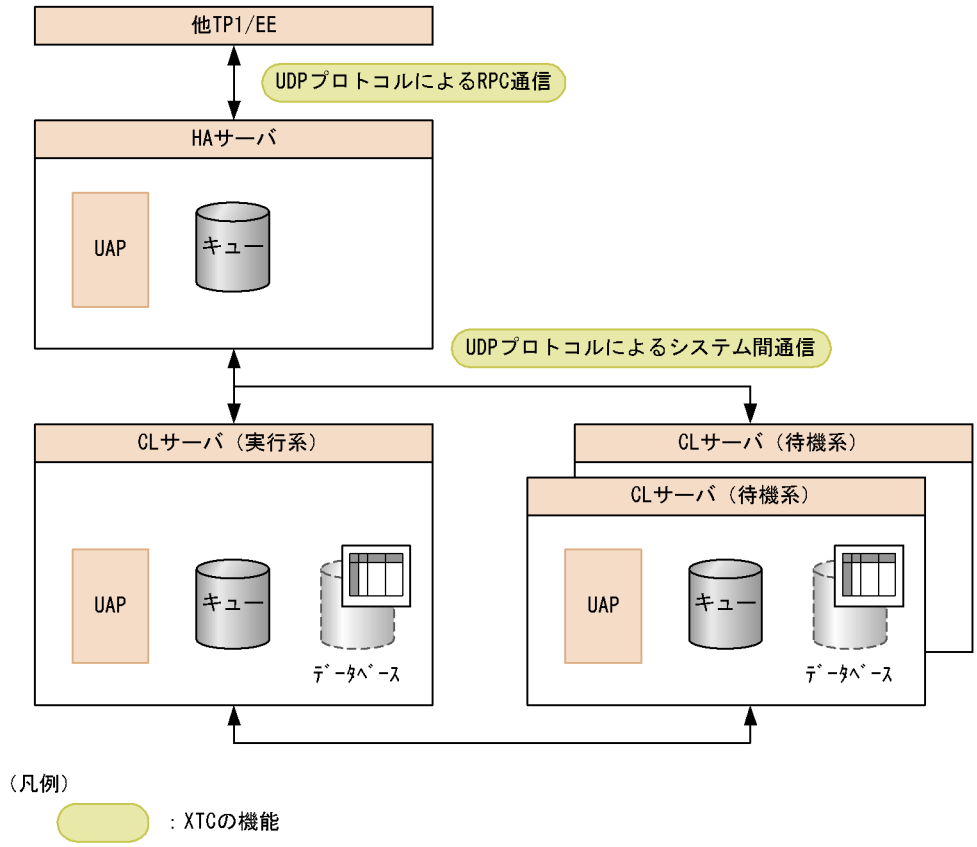
ここでは，それぞれの機能の概要と，TP1 キャッシュ機能使用時の位置づけを説明します。

また，永続メッセージと非永続メッセージの違いについても説明します。

2.1.1 高速通信・高速処理を実現する機能

高速通信・高速処理を実現する機能として，次の図に示す機能があります。

図 2-1 高速通信・高速処理を実現する機能



■ UDP プロトコルによる RPC 通信

RPC 通信を UDP プロトコルによって行うことができます。
詳細については，「2.3　XTC の RPC 通信」を参照してください。

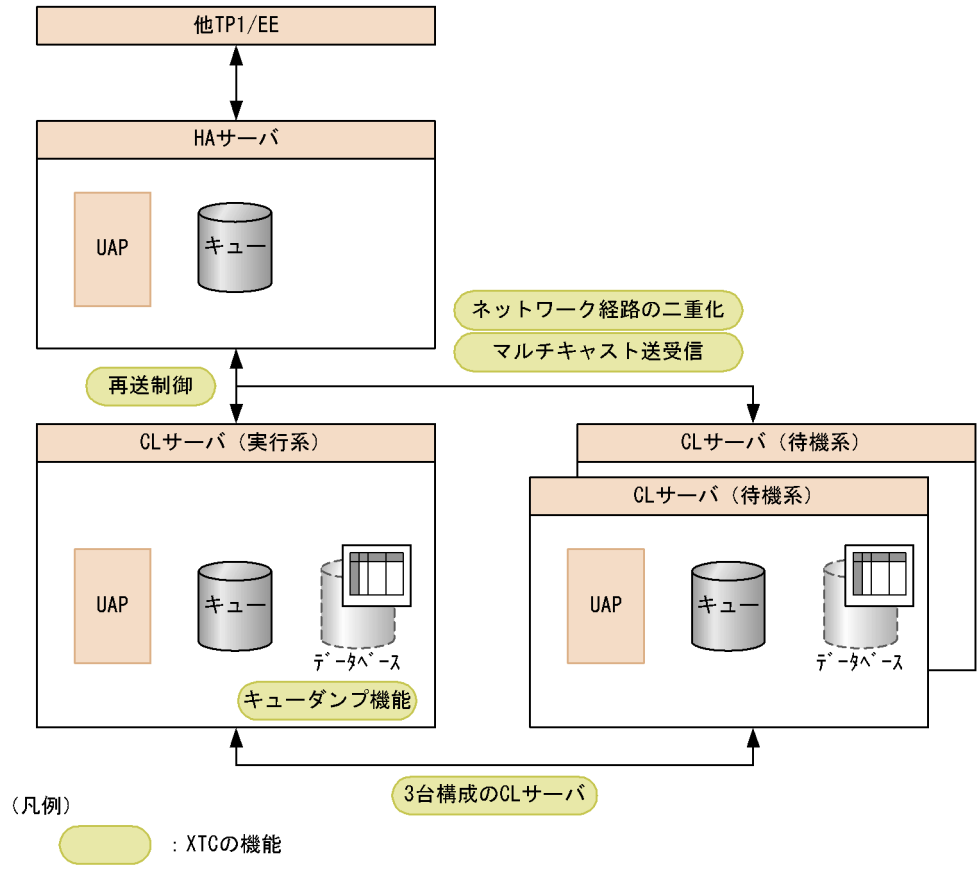
■ UDP プロトコルによるシステム間通信（高速メッセージ送信制御）

システム間通信を UDP プロトコルによって行うことで，高速にメッセージを送受信できます。
詳細については，「2.2　高速メッセージ送信制御」を参照してください。

2.1.2　高信頼性を実現する機能

高信頼性を実現する機能として，次の図に示す機能があります。

図 2-2　高信頼性を実現する機能



■再送制御

UDP プロトコルによって送受信するデータの整合性を保証するために，メッセージがロストした場合でも，再送制御によってメッセージを保証します。
詳細については，「2.4　UDP 通信機能」を参照してください。

■キューダンプ機能

オンライン終了時にキューに滞留したメッセージをファイル出力することで，次回オンライン開始時の整合性を保ちます。
詳細については，「2.9　キューダンプ機能」を参照してください。

■ネットワーク経路の二重化（W-send 機能）

ネットワーク経路を二重化した構成にすることで，ネットワークの信頼性を高めることができます。
詳細については，「2.4　UDP 通信機能」を参照してください。

■マルチキャスト送受信（高速メッセージ送信制御）

メッセージをマルチキャストで送受信することによって，系切り替え時のフェールオーバ時間を短縮できます。

■ 3 台構成の CL サーバ

CL サーバを 3 台構成にすることで，データの高信頼性を確保します。
詳細については，「7.　CL 連携による系切り替え機能」を参照してください。

2.1.3　運用性を向上させる機能

運用性を向上させる機能として，次の機能があります。

■トラブルシュート情報の取得とステータスファイルレス機能

トラブルシュート情報は TP1/EE で取得することで，一元管理します。また，ステータスファイルレス機能によって，TP1 キャッシュ機能時に取得する情報を限定します。
詳細については，「2.7　ステータスファイルレス機能」を参照してください。

■ CL サーバのシミュレート機能

HA モニタ環境，および待機系の環境を構築する前に，TP1 キャッシュ機能使用時の動作をシミュレートすることができます。
詳細については，「2.10.1　CL サーバのシミュレート機能」を参照してください。

2.1.4　永続メッセージと非永続メッセージ

CL サーバに対するメッセージの送信では，送信するメッセージを待機系で保証するかどうかを選択できます。このとき，保証するメッセージを**永続メッセージ**，保証しないメッセージを**非永続メッセージ**といいます。

永続メッセージが有効となる通信を次に示します。

- HA サーバから CL サーバへの，マルチキャストでのメッセージ送信要求
- CL サーバからのメッセージ送信要求

永続メッセージは送信先の待機系で破棄されませんが，非永続メッセージは破棄されます。

永続メッセージの送信が完了（トランザクション同期点完了）していると，送信先の実行系の障害発生によって待機系に切り替わってもメッセージは保証され，継続してトランザクション処理が実行できます。非永続メッセージでは，待機系に切り替わると継続したトランザクション処理が実行できません。

2.2 高速メッセージ送信制御

ここでは，高速メッセージ送信制御について説明します。

2.2.1 高速メッセージ送信制御の概要

TP1 キャッシュ機能使用時，高速にメッセージを送受信するために，TP1/EE 間でのシステム間通信を UDP プロトコルによって行います。これを，**高速メッセージ送信制御**といいます。

一般に，UDP プロトコルはコネクションを確立することなくメッセージの送受信ができる反面，メッセージが保証されないという問題があります。そのため，高速メッセージ送信制御では次のことを行ってメッセージを保証しています。

- メッセージの送達確認
- メッセージの欠落，重複，および到着順序のチェックやフロー制御

(1) 送受信できるメッセージ

高速メッセージ送信制御では，次に示すメッセージを送受信できます。

- トランザクション同期の一方送信メッセージ
- トランザクション非同期の一方送信メッセージ

なお，これらのメッセージを総称して MCH メッセージといいます。

(a) トランザクション同期の一方送信メッセージ

トランザクション同期の一方送信メッセージとは，送信要求元トランザクションのコミット確定時に有効となり，トランザクションの延長で，同期点処理中（UAP リターン後）に送信するメッセージです。

クライアント UAP で，サービス要求に必要な項目を ee_mch_cmtsend_sync 関数に指定することで，サーバ UAP にサービスを要求します。サーバ UAP は複数のサービスを実行できます。

トランザクション同期の一方送信メッセージの特徴を次に示します。

- CL サーバ（実行系）からの送信要求はできません。
- メッセージは，同一トランザクション処理からの送信要求順に送信します。
- メッセージ送信失敗時はエラートランザクション（ERRTRNS または ERRTRNR）を起動し，メッセージを破棄します。

(b) トランザクション非同期の一方送信メッセージ

トランザクション非同期の一方送信メッセージとは，送信要求元トランザクションのコミット確定後，トランザクション処理とは別のスレッド（送信スレッド）で送信するメッセージです。

クライアント UAP で，サービス要求に必要な項目を ee_mch_cmtsend 関数に指定することで，サーバ UAP にサービスを要求します。サーバ UAP は複数のサービスを実行できます。

トランザクション非同期の一方送信メッセージの特徴を次に示します。

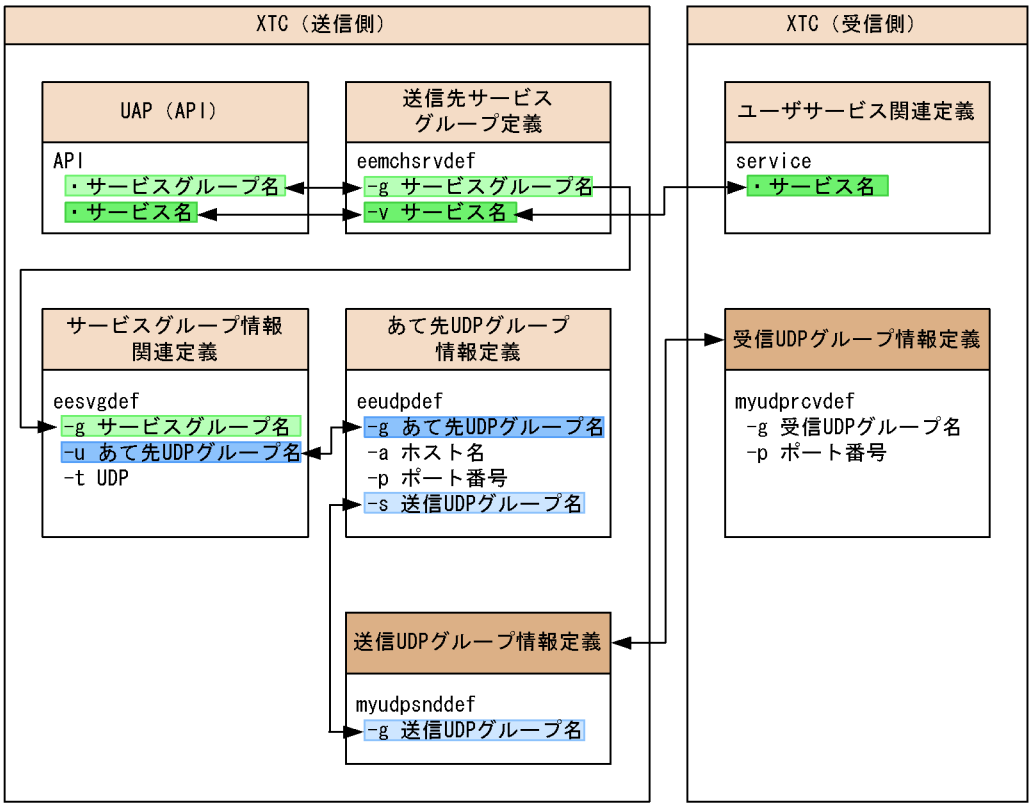
- メッセージはコミット確定時に出力キュー（OTQ）に登録され，その際に通番を採番します。これによって，未送信のメッセージから通番順に送信できるようになり，送信順序が保証されます。
- メッセージ送信失敗時はエラートランザクション（ERRTRNS）を起動し，メッセージを保持（別タイミングで再送）します。

（2） API の引数と定義の関連

高速メッセージ送信制御で使用する API の引数と定義には関連があります。API と定義の指定には整合性を持たせてください。

API の引数と定義の関連の例を次の図に示します。

図 2-3　高速メッセージ送信で使用する API の引数と定義の関連の例



（凡例）

←→：同じ値を指定します。

各定義の詳細については，「5.3　XTC サービス定義」を参照してください。

API の詳細については，マニュアル「TP1/Server Base Enterprise Option プログラム作成の手引」を参照してください。

(3) メッセージの通信形態

高速メッセージ送信制御では，UDP プロトコルを使用したクライアント／サーバ形態で通信します。

(a) クライアント側（送信形態）

マルチキャストまたはユニキャストによってメッセージを送信できます。

(b) サーバ側（受信形態）

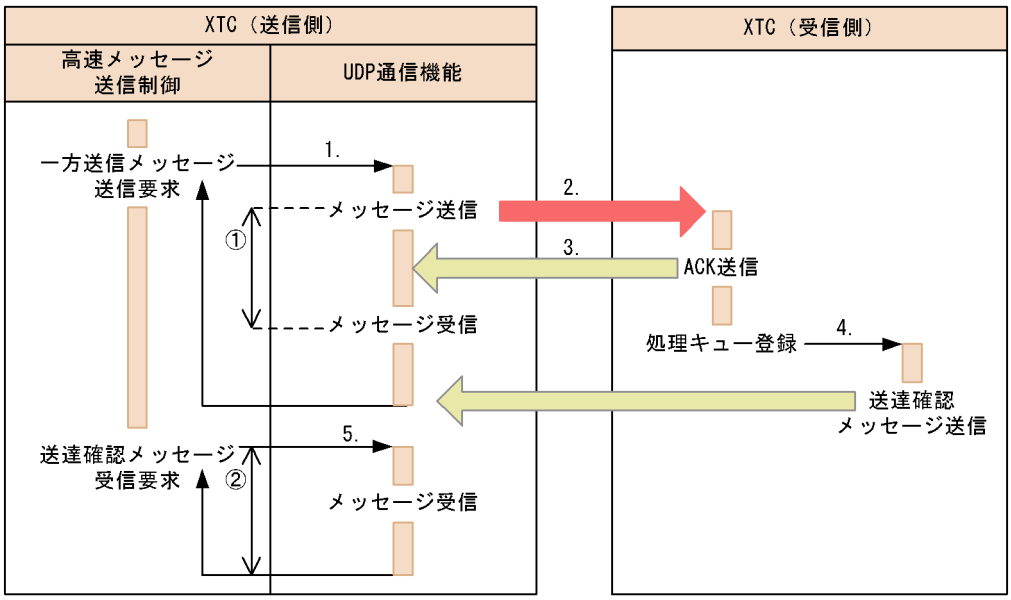
マルチキャストまたはユニキャストによって送信されたメッセージを受信できます。
メッセージは TP1/EE 内の受信スレッドが受信して，処理キューに登録します。

2.2.2　UDP 通信機能によるシステム間通信の流れ

高速メッセージ送信制御は，UDP 通信機能によって実現しています。

UDP 通信機能では，メッセージの整合性を保証するために送達確認を行います。UDP 通信機能によるシステム間通信の流れを次の図に示します。

図 2-4　UDP 通信機能によるシステム間通信の流れ



（凡例）

：ユーザメッセージの流れ

：制御メッセージの流れ

：制御の流れ

：タイマ監視範囲

ACK　：肯定応答

①　：UDPグループ情報関連定義のeeudpdef定義コマンドまたはmyudpsnddef定義コマンドの-wオプションまたは-Wオプションの指定値

②　：高速メッセージ送信関連定義のmch_send_ack_timerオペランドの指定値

説明

1. 送信側の UAP から，一方送信メッセージの送信要求を行います。
2. UDP 通信機能によって，一方送信メッセージを送信します。
3. 受信側では，一方送信メッセージに対する肯定応答（ACK）を送信します。
4. 送達確認用の処理キューを登録し，送信スレッドから肯定応答不要で送達確認

メッセージを送信します。

5. 送信側は，受信要求を行って，一方送信メッセージに対する送達確認メッセージを受信します。

UDP 通信機能の詳細については，「2.4　UDP 通信機能」を参照してください。

2.2.3　API からの要求受け付け時の動作

高速メッセージ送信制御では，受信側サーバの構成や送信形態によって，API からの要求受け付け時の動作が異なります。

API からの要求受け付け時の動作を次の表に示します。

表 2-1　API からの要求受け付け時の動作

項番	受信側サーバ	送信形態	動作
1	HA サーバ	ユニキャスト	―
2		マルチキャスト	受信側では，送達確認エラーを送信します。 送信側では，エラートランザクションを起動します。
3		自プロセスあて	―
4	CL サーバ	ユニキャスト	受信側では，送達確認エラーを送信します。※ 送信側では，エラートランザクションを起動します。※
5		マルチキャスト	非永続メッセージの場合，待機系では受信メッセージを破棄します。
6		自プロセスあて	―

（凡例）

―：正常に動作します。

注※

CL 単独起動機能を使用している場合，ここに示す動作はしません。正常に動作します。

2.2.4　一方送信メッセージの送信

ここでは，一方送信メッセージを送信するときの流れ，および一方送信メッセージの一括送信について説明します。

（1）一方送信メッセージを送信するときの流れ

トランザクション同期の場合と，トランザクション非同期の場合に分けて説明します。

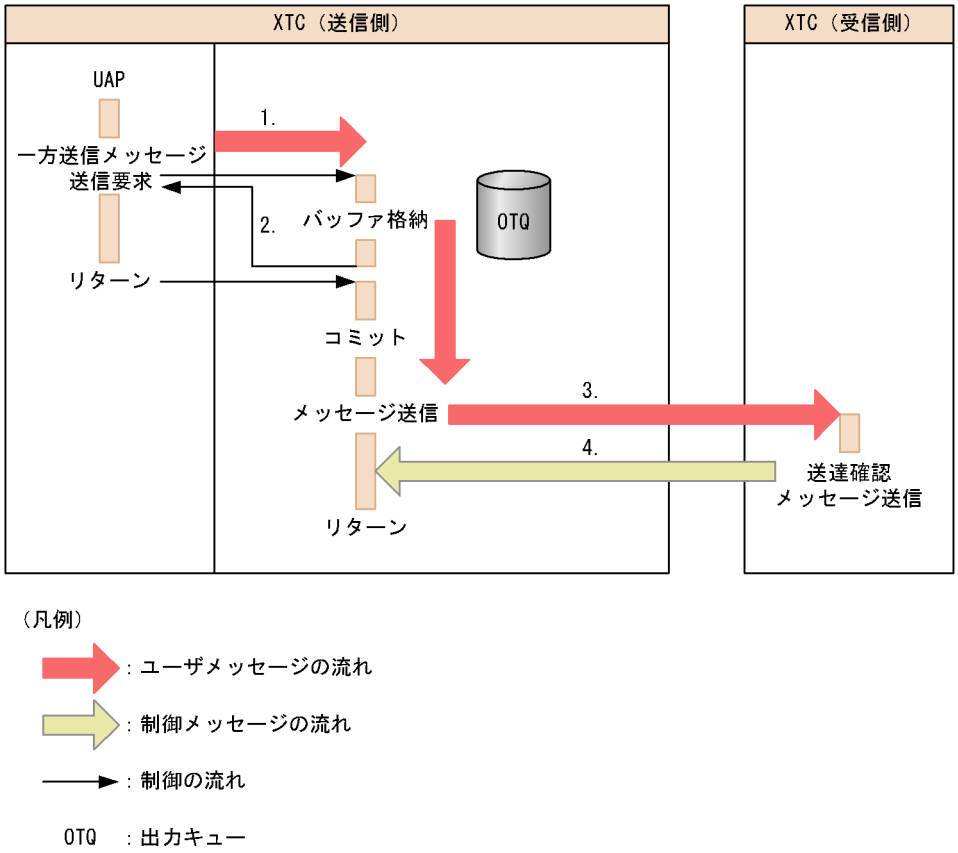
（a）トランザクション同期の場合

ユニキャストまたはマルチキャストによる送信

トランザクション同期の一方送信メッセージの送信で，ユニキャストまたはマルチキャ

ストによる送信の流れを次の図に示します。

図 2-5　ユニキャストまたはマルチキャストによる送信の流れ（トランザクション同期）



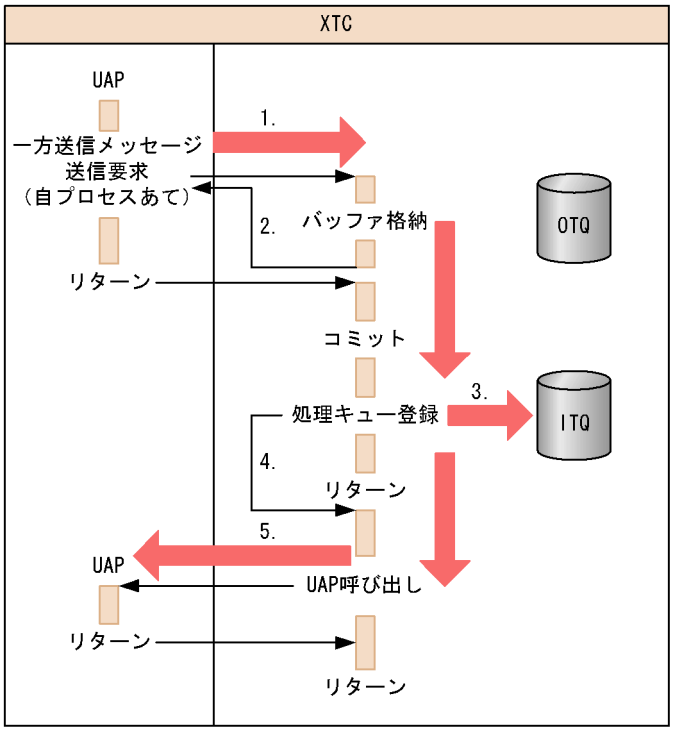
説明

1. クライアント UAP から，ユニキャストまたはマルチキャストによる一方送信メッセージの送信要求を行います。
2. TP1/EE は，出力キュー（OTQ）が閉塞状態でなければメッセージを UDP 用送信バッファ（UOBF）に格納し，UAP にリターンします。
3. UAP リターン後，TP1/EE はコミットが確定すると，指定された送信先にメッセージを送信します。
4. 送達確認メッセージを受信することで，送信処理を完了します。

自プロセスあての送信

自プロセスあての送信の流れを次の図に示します。

図 2-6 自プロセスあての送信の流れ



（凡例）

：ユーザメッセージの流れ

：制御の流れ

OTQ ：出力キュー

ITQ ：入力キュー

説明

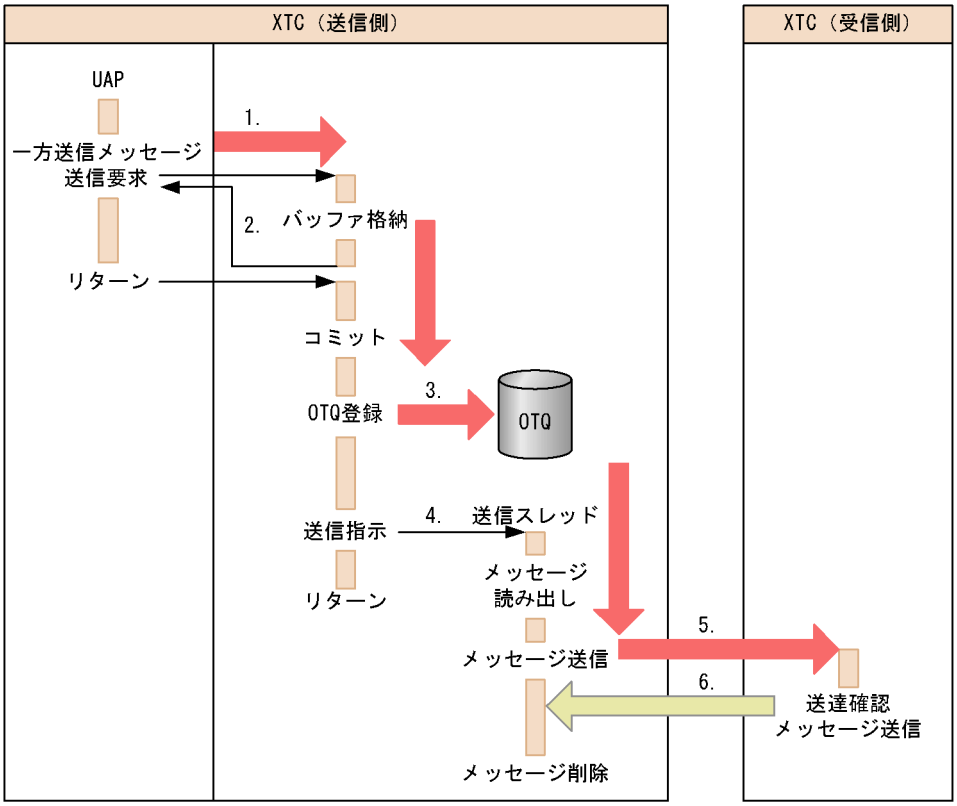
1. クライアント UAP から，自プロセスあての一方送信メッセージの送信要求を行います。
2. TP1/EE は，出力キューが閉塞状態でなければメッセージを UDP 用送信バッファに格納し，UAP にリターンします。
3. UAP リターン後，TP1/EE は同期点処理でのコミット確定後に，メッセージを入力キュー（ITQ）に登録します。
4. 入力キューに登録されたメッセージは処理スレッドによって引き出されます。
5. 処理スレッドがトランザクションを開始して，UAP にメッセージを引き渡します。

(b) トランザクション非同期の場合

ユニキャストまたはマルチキャストによる送信

トランザクション非同期の一方送信メッセージの送信で，ユニキャストまたはマルチキャストによる送信の流れを次の図に示します。

図 2-7　ユニキャストまたはマルチキャストによる送信の流れ（トランザクション非同期）



（凡例）

：ユーザメッセージの流れ

：制御メッセージの流れ

：制御の流れ

OTQ　：出力キュー

説明

1. クライアント UAP から，ユニキャストまたはマルチキャストによる一方送信メッセージの送信要求を行います。

2. TP1/EE は，出力キューが閉塞状態でなければメッセージを UDP 用送信バッファに格納し，UAP にリターンします。
3. UAP リターン後，TP1/EE は同期点処理でのコミット確定後にメッセージを出力キューに登録し，その際に通番を採番します。
4. 送信スレッドに対して，メッセージ送信指示を行います。
5. 送信スレッドは，未送信のメッセージを読み出し，指定された送信先にメッセージを送信します。
6. 送達確認メッセージを受信すると，出力キューからメッセージを削除します。読み出していないメッセージがある場合，メッセージ読み出し要求を行います。

自プロセスあての送信

トランザクション非同期の場合，自プロセスあての送信の流れは，基本的には「ユニキャストまたはマルチキャストによる送信」と同様です。

ただし，自プロセスあての送信であるため，送信スレッドからのメッセージの送信先は，「(1)(a)　トランザクション同期の場合」の「自プロセスあての送信」と同様，自プロセスの UAP となります。

(2) 一方送信メッセージの一括送信

同一出力キュー（OTQ）上にある他プロセスあてのトランザクション非同期の一方送信メッセージを，最大 1024 件を一括で送信できます。一括で送信できるメッセージの数は，高速メッセージ送信関連定義の mch_send_lump_count オペランドで指定します。

一括送信できるのは，属性が同じメッセージだけです。次の属性ごとにまとめて一括送信します。

- メッセージ送信先で登録された処理キューの中で優先的に起動するかどうか
（ee_mch_cmtsend 関数の priority 引数に EEMCH_HI または EEMCH_LOW を指定しているか）
- 通信方式がユニキャストかマルチキャストか
（ee_mch_cmtsend 関数の flags 引数に EENOFLAGS または EEMCH_MULTI_CL を指定しているか）
- メッセージが永続か非永続か
（ee_mch_cmtsend 関数の flags 引数に EEMCH_PERMANENCE を指定しているか）

なお，異なる属性のメッセージが出現した場合は，それまでにまとめたメッセージだけで一括送信します。

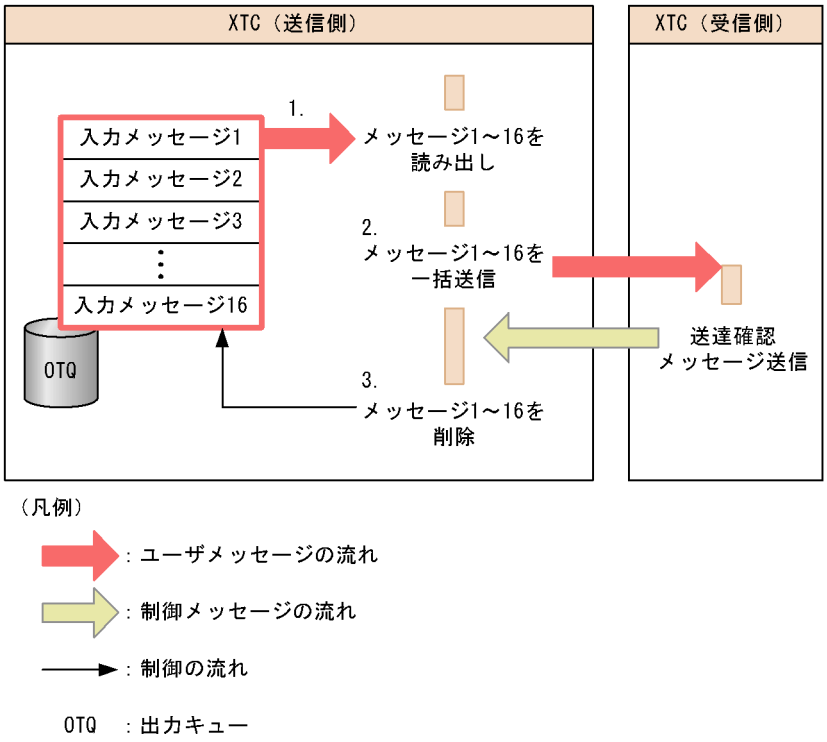
メッセージの一括送信は，出力キュー（OTQ）ごとの一方送信メッセージ数が多く，属性が同じ場合に活用できます。

(a) 一方送信メッセージを一括送信するときの流れ

一方送信メッセージを一括送信するときの流れについて説明します。ここでは，クライアント UAP からサーバへ非同期の一方送信メッセージを一括送信するときの流れを例に

示します。この例では，高速メッセージ送信関連定義の mch_send_lump_count オペランドの指定値は 16 です。

図 2-8　一方送信メッセージを一括送信するときの流れ



説明

1. 同一出力キュー（OTQ）に登録されている未送信メッセージを，16 件読み出してまとめます。未送信メッセージが 16 件よりも少ない場合は，すべての未送信メッセージをまとめます。
2. 指定された送信先に，まとめたメッセージを一括送信します。
3. 送達確認メッセージを受信すると，出力キューからメッセージを削除します。読み出していないメッセージがある場合，メッセージ読み出し要求を行います。

(b) 一括送信時の注意事項

一方送信メッセージを一括で送信する場合，クライアント側とサーバ側の XTC のバージョンが一致している必要があります。

注意事項

製品バージョンの変更は，すべてのシステムを停止してから実施してください。バージョンが 01-02 以降のクライアントが稼働中に送信先のバージョンを変更する場合，次のことに注意してください。

送信先のバージョンを 01-01 以前から 01-02 以降に変更する場合

メッセージを一括送信できるようになります。サービスグループレベルの出力キュー（OTQ）閉塞中に変更することをお勧めします。

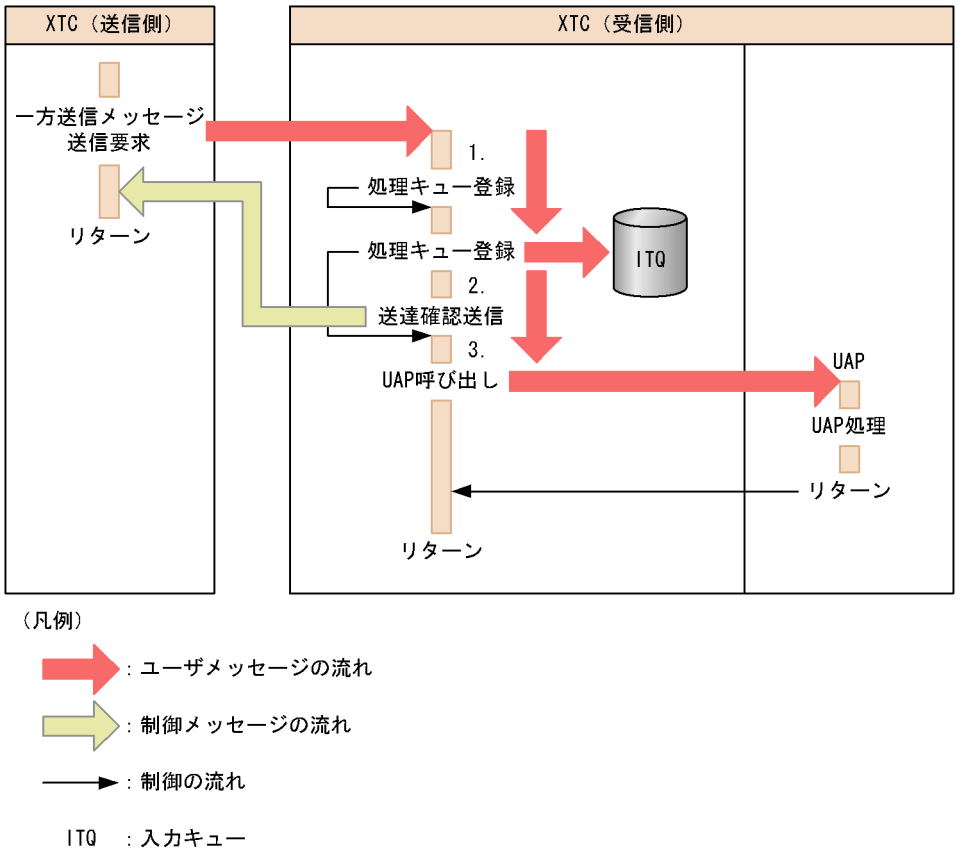
送信先のバージョンを 01-02 以降から 01-01 以前に変更する場合

一方送信メッセージの欠落が発生するおそれがあるため変更しないでください。

2.2.5 一方送信メッセージの受信

ここでは，クライアント側から一方送信メッセージを受信するときの流れを説明します。

図 2-9 一方送信メッセージの受信の流れ



説明

1. 一方送信メッセージを受信すると，送達確認用の処理キューを登録します。処理

キューはサービスごとにシリアライズされます。
2. 送信スレッドで処理キューが引き出され，クライアントからの一方送信メッセージを入力キュー（ITQ）に登録し，送達確認メッセージを送信します。
一括送信されたメッセージを受信した場合は，処理キューが引き出されてメッセージを入力キュー（ITQ）に登録する処理が，受信したメッセージの数だけ繰り返されます。
3. 処理キューに登録したサービスは，処理スレッドによって引き出され，トランザクションを開始して UAP にメッセージを引き渡します。

2.2.6 出力キューによるメッセージ管理

UAP から一方送信メッセージの送信要求があったときに，指定されたサービスごとに出力キュー（OTQ）の閉塞状態の管理ができるよう，出力キューは送信先サービスグループのサービス単位に管理されます。

なお，出力キューはメモリ上で管理されるため，サーバダウンによる再開始時は閉塞状態を引き継げません。ただし，CL サーバについては，出力キューの閉塞状態を実行系から待機系に転送することで，閉塞状態を保証（永続化）します。出力キューの閉塞状態の永続化については，「7.6.4 トランザクションと非同期に転送されるリソースの永続化」を参照してください。

(1) 出力キューの定義

出力キューは，送信元となる XTC の送信先サービス関連定義の eemchsrvdef 定義コマンドで定義します。送信先となる XTC のサービスグループごとに，サービスグループ名とサービス名を定義します。

(2) 出力キューの閉塞

出力キューが閉塞すると，次のような状態になります。

- UAP からの一方送信メッセージの送信要求が受け付けられない
- 出力キュー上の一方送信メッセージを送信しない

出力キューが閉塞する要因を次に示します。

- オンライン開始時に出力キューを自動閉塞するように定義されている場合
- API またはコマンドによる指示があった場合
- 自動閉塞の対象となる障害が発生した場合
自動閉塞の対象となる障害を次の表に示します。

表 2-2 出力キューの自動閉塞の対象となる障害

項番	自動閉塞の対象となる障害	システムの処理
1	一方送信メッセージの送信障害※	送信先サービスグループ下の全サービスに対応する出力キューを閉塞（サービスグループレベルの自動閉塞）

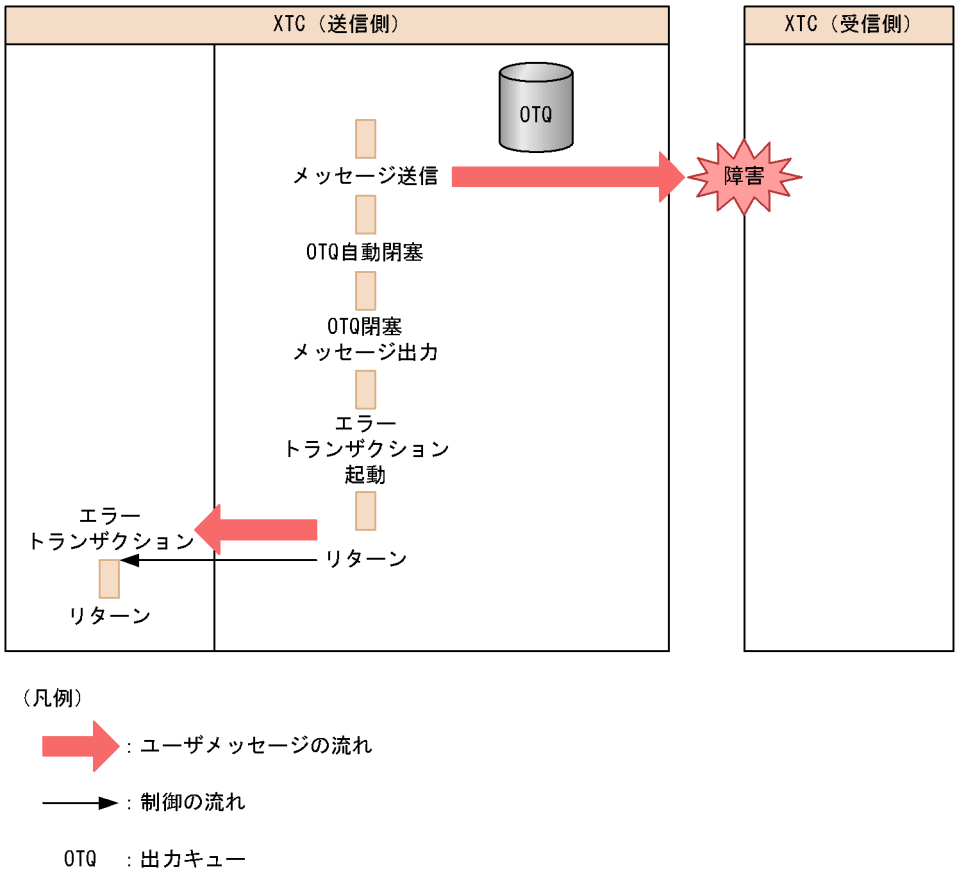
項番	自動閉塞の対象となる障害	システムの処理
2	一方送信メッセージの送信障害（パケットサイズの不正）	
3	一方送信メッセージの送達確認待ちタイムアウト※	
4	送達確認エラーメッセージ受信（CL同期エラー）※	
5	送達確認エラーメッセージ受信（送信先サービスグループ名の不正）	
6	送達確認エラーメッセージ受信（送信先サービスの閉塞）※	送信先サービスに対応する出力キューを閉塞（サービスレベルの自動閉塞）
7	送達確認エラーメッセージ受信（送信先サービス名の不正，メッセージ送信形態・オプションの不正）	

注※

自動閉塞するのは，送信先サービス関連定義の eemchsrvdef 定義コマンドで出力キューの自動閉塞を定義した場合だけです。

一方送信メッセージの送信障害時の出力キューの自動閉塞を次の図に示します。

図 2-10　一方送信メッセージの送信障害時の出力キューの自動閉塞



説明

一方送信メッセージの送信障害が発生すると，出力キューを自動閉塞します。その後，出力キューの閉塞を示すメッセージを出力して，エラートランザクションを起動します。

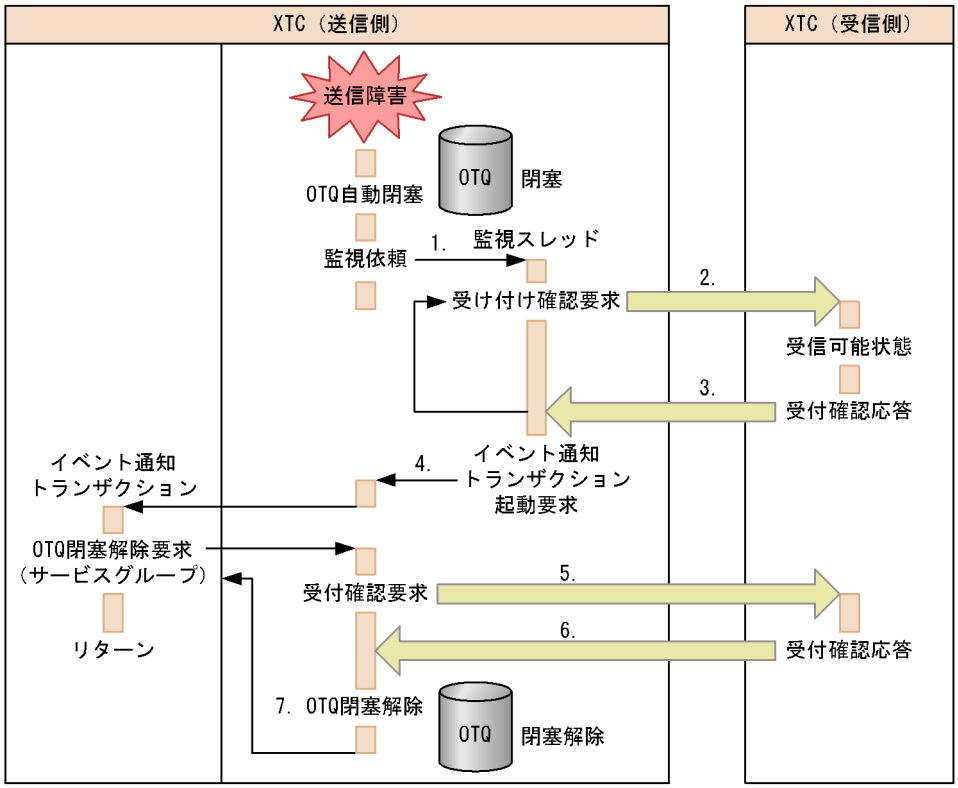
(3) 出力キューの閉塞状態の監視と閉塞解除

XTC は，出力キューの閉塞状態を監視します。監視時間の間隔は，高速メッセージ送信関連定義の mch_send_otqwatch_time_interval オペランドで定義します。

また，出力キューの閉塞解除は，API またはコマンドによる指示があったときにだけ行います。

出力キューの閉塞監視と閉塞解除の流れを次の図に示します。

図 2-11 出力キューの閉塞監視と閉塞解除の流れ



（凡例）

：制御メッセージの流れ

：制御の流れ

OTQ ：出力キュー

説明

1. メッセージ送信障害発生時，サービスグループ下の全サービスに対応する出力キューを自動閉塞し，送信先サービスグループの状態監視を依頼します。
2. 監視スレッドでは，送信先サービスグループに対して一定間隔で受付確認要求メッセージを送信します。
3. 受付確認要求メッセージを受信した受信側は，一方送信メッセージを受信できるかどうか判断し，受信可能であれば，受付確認応答（受付可）メッセージを送信します。受信できない場合は，受付確認応答（受付不可）メッセージを送信します。
4. 受付確認応答（受付可）メッセージを受信した送信側は，イベント通知トランザクション起動要求を行い，UAP に通信できることを通知します。受付確認応答（受付不可）メッセージを受信した場合は，2. に戻って処理を継続します。なお，イベント通知トランザクションは，一度起動すると，新たにサービスグループレ

ベルの自動閉塞が発生しないかぎり起動しません。

5. 出力キュー閉塞解除要求時，閉塞解除対象の送信先サービスグループに対し，受付確認要求メッセージを送信します。
6. 受付確認要求メッセージを受信した受信側は，一方送信メッセージが受信できるかどうかを判断し，受付確認応答（受付可）メッセージまたは受付確認応答（受付不可）メッセージを送信します。
7. 受付確認応答（受付可）メッセージを受信した場合は，出力キューの閉塞を解除します。出力キューの閉塞解除によって，出力キュー上の一方送信メッセージの送信を再開します。

（4） XTC 終了処理の待ち合わせ

XTC の終了処理時，出力キューに未送信メッセージが残っている間は，終了処理の待ち合わせを行います。終了処理の待ち合わせを行うのは，XTC の終了形態が正常終了または計画停止 A の場合です。計画停止 B，強制終了，またはプロセスダウンの場合，待ち合わせは行いません。

待ち合わせ方法は次の 3 とおりです。

- 出力キューの閉塞状態に関係なく，未送信メッセージがある間は XTC を終了しません。
- 出力キューが閉塞している場合を除いて，未送信メッセージがある間は XTC を終了しません。
- mch_send_end_otqwatch_time オペランドで指定した時間が経過するまで，XTC を終了しません。指定時間が経過すると，未送信メッセージが残っていても XTC を終了します。

待ち合わせ中であっても，eemchotqend コマンドを入力することで XTC を終了できます。

なお，待ち合わせ方法は高速メッセージ送信関連定義の mch_send_end_otqwatch オペランドで指定します。

2.2.7 送達確認待ちの時間監視

高速メッセージ送信制御では，クライアント側から一方送信メッセージを送信してからサーバ側で送達確認メッセージを受信するまでの時間（送達確認待ちの時間）を監視できます。送達確認待ちの時間を過ぎても応答がない場合は，定義に従ってメッセージを再送します。送達確認待ちの時間や再送のリトライ回数などは，高速メッセージ送信関連定義で指定します。

なお，メッセージを分割する場合は，最終パケットを送信してから監視を開始します。パケットレベルでの時間監視については，「2.4.1 UDP プロトコルを使用した通信」を参照してください。

2.2.8 障害処理

ここでは，高速メッセージ送信制御での主な障害発生時の流れと，障害発生時のシステムの処理について説明します。

(1) 障害発生時の処理の流れ

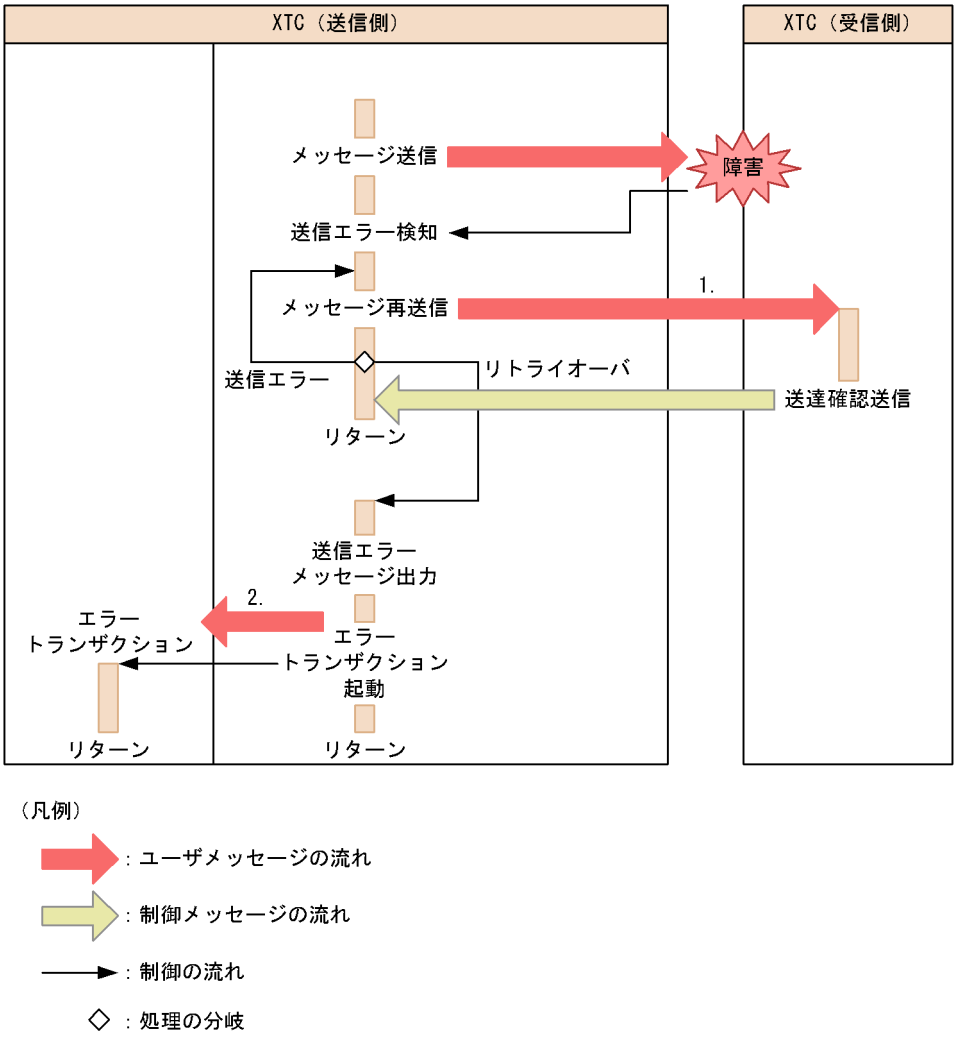
高速メッセージ送信制御では，主に次のような障害の発生が考えられます。

- メッセージ送信処理の障害
- メッセージ送信先サーバのリソース不足による障害
- メッセージ送信先のサービス閉塞による障害
- メッセージ送信先のサービスグループ名またはサービス名の不正による障害

(a) メッセージ送信処理の障害

メッセージ送信処理での障害時の流れを次の図に示します。

図 2-12　メッセージ送信処理での障害時の流れ



説明

1. メッセージ送信時の障害を検知すると，高速メッセージ送信関連定義に従ってメッセージを再送します。
2. リトライ回数が定義値を超えた場合は，エラーメッセージを出力し，エラートランザクションを起動します。

(b) メッセージ送信先サーバのリソース不足による障害

メッセージ送信先サーバのリソース不足による障害時の流れを次に示します。

1. サーバ側で，メッセージ受信時に処理キュー制御用バッファ（PCE）が不足している場合，送達確認エラーメッセージ（リソース不足）を応答し，受信メッセージを破棄します。

2. クライアント側で送達確認エラーメッセージ（リソース不足）を受信すると，高速メッセージ送信関連定義に従ってメッセージを再送します。
3. リトライ回数が定義値を超えた場合は，エラーメッセージを出力し，エラートランザクションを起動します。

(c) メッセージ送信先のサービス閉塞による障害

メッセージ送信先のサービス閉塞による障害時の流れを次に示します。

1. サーバ側で，メッセージ受信時に該当するサービスが閉塞中の場合，送達確認エラーメッセージ（サービス閉塞中）を応答し，受信メッセージを破棄します。
2. クライアント側で送達確認エラーメッセージ（サービス閉塞中）を受信すると，エラーメッセージを出力し，エラートランザクションを起動します（再送はしません）。

(d) メッセージ送信先のサービスグループ名またはサービス名の不正による障害

メッセージ送信先のサービスグループ名またはサービス名の不正による障害時の流れを次に示します。

1. サーバ側で，メッセージ受信時に受信メッセージ中のサービスグループ名またはサービス名が不正だった場合，送達確認エラーメッセージ（サービスグループ名不正またはサービス名不正）を応答し，受信メッセージを破棄します。
2. クライアント側で，送達確認エラーメッセージ（サービスグループ名不正またはサービス名不正）を受信すると，エラーメッセージを出力し，エラートランザクションを起動します（再送はしません）。

(2) 一方送信メッセージでの障害要因とシステムの処理

一方送信メッセージの送信で発生する障害の要因と，システムの処理を次の表に示します。

表 2-3 一方送信メッセージでの障害要因とシステムの処理

項番	障害要因	システムの処理
1	送信前に起きた，XDB の障害などによるロールバック決着	ロールバック決着で ERRTRNR を起動
2	送信障害（パケットサイズの不正などの，RPC 関連定義の不正）	• ERRTRNS を起動 • サービスグループレベルの出力キューの自動閉塞 • ほかの未送信メッセージの送信処理を継続※1
3	送信障害※2	• 送信エラーとなったメッセージの送信リトライ実施※3 • 送信リトライ失敗時，ERRTRNS を起動 • サービスグループレベルの出力キューの自動閉塞※4 • ほかの未送信メッセージの送信処理を継続※1
4	送達確認待ちタイムアウト※2	
5	送達確認エラーメッセージ受信（CL 同期エラー）	

項番	障害要因	システムの処理
6	送達確認エラーメッセージ受信（リソース不足（PCE，IBF））	• 送信エラーとなったメッセージの送信リトライ実施※3 • 送信リトライ失敗時，ERRTRNS を起動 • ほかの未送信メッセージの送信処理を継続※1
7	• 送達確認エラーメッセージ受信（送信先サービスの閉塞） • 自プロセスあてのサービス閉塞	• ERRTRNS を起動 • サービスレベルの出力キューの自動閉塞※4 • ほかの未送信メッセージの送信処理を継続※1
8	送達確認エラーメッセージ受信（送信先サービスグループ名の不正）	• ERRTRNS を起動 • サービスグループレベルの出力キューの自動閉塞 • ほかの未送信メッセージの送信処理を継続※1
9	送達確認エラーメッセージ受信（送信先サービス名の不正）	• ERRTRNS を起動 • サービスレベルの出力キューの自動閉塞 • ほかの未送信メッセージの送信処理を継続※1
10	送達確認エラーメッセージ受信（メッセージ送信形態，オプションの不正）	

（凡例）

PCE：処理キュー制御用バッファ

IBF：受信バッファ

注※1

トランザクション同期の一方送信メッセージの場合の動作です。トランザクション非同期の場合は該当しません。

注※2

メッセージが送信先に届いている場合があるため，UAP で送達確認を行う必要があります。

注※3

高速メッセージ送信関連定義で，送信リトライ回数および送信リトライ間隔を指定している場合にだけ実施します。

注※4

送信先サービス関連定義の eemchsrvdef 定義コマンドで，出力キューの自動閉塞を指定している場合にだけ実施します。

(3) 一方送信メッセージの送信に失敗したとき，または送信前にロールバック決着となったときに参照できる情報

一方送信メッセージの送信に失敗したとき，または送信前にロールバック決着となったときに参照できる情報を次の表に示します。

表 2-4 一方送信メッセージの送信に失敗したとき，または送信前にロールバック決着となったときに参照できる情報

項番	発生した障害	参照するエラートランザクション	参照できる情報
1	トランザクション同期の一方送信メッセージの送信失敗	ERRTRNS	• 送信先サービスグループ名 • 送信先サービス名 • 送信エラー要因（障害要因） • 出力キューの自動閉塞の有無 • 送信メッセージ
2		ERRTRNR	• 入力メッセージ • 送信先サービスグループ名※1 • 送信先サービス名※1 • 送信エラー要因（障害要因）※1 • 出力キューの自動閉塞の有無※1 • 送信メッセージ※1
3	トランザクション非同期の一方送信メッセージの送信失敗	ERRTRNS	• 送信先サービスグループ名 • 送信先サービス名 • 送信エラー要因（障害要因） • 出力キューの自動閉塞の有無 • 出力キュー通番 • 送信メッセージ※2
4	メッセージ送信前の XDB 障害などによるロールバック決着	項番 2 と同様です。	

注※ 1

ee_mch_cmtsend_get 関数によって一方送信メッセージ情報を取得することで，同一トランザクション内で一方送信メッセージ送信要求を行ったメッセージ情報を参照できます。

注※ 2

一括送信したメッセージの場合，参照できるのは送信に失敗した先頭メッセージだけです。

(4) トランザクション非同期の一方送信メッセージ送信障害後の再送

トランザクション非同期の一方送信メッセージの送信で障害が発生すると，メッセージの送信を中断して，出力キューで未送信メッセージとして管理します。

未送信メッセージを再送するタイミングは，次のとおりです。

■出力キューが閉塞した場合

出力キューの閉塞解除後に再送します。

■出力キューが閉塞していない場合

高速メッセージ送信関連定義の mch_send_otqschedule_interval オペランドで指定した間隔で送信を待ち合わせ，その後，再送します。障害要因が送信先サービス閉塞のときはサービス単位で，その他の障害要因のときはサービスグループ単位での待ち合わせとなります。待ち合わせ中に新たな送信要求があっても送信しません。また，出力キュー閉塞解除要求を行った場合にも再送します。

なお，一括送信したメッセージが未送信メッセージとして管理された場合は，1 メッセージずつ再送します。

(5) 注意事項

サーバ側では，一方送信メッセージの受信時に同一メッセージを受信しても同一トランザクションを起動しないように，メッセージをチェックします。ただし，次の場合についてはチェックの対象外となります。

- HA サーバがダウンして再起動した場合
- クライアントが永続指定でない一方送信メッセージを送信している間に，CL サーバ（実行系）がダウンして系切り替えが発生した場合

2.3 XTC の RPC 通信

ここでは，XTC の RPC 通信について説明します。

2.3.1 UDP プロトコルを使用した RPC 通信

OpenTP1 の通常の RPC 通信では，クライアントとサーバの間の通信を，TCP プロトコルを使用したコネクション経由で行います。XTC では，UDP プロトコルを使用した RPC 通信ができます。UDP プロトコルを使用した RPC 通信を行うと，クライアントとサーバの間をコネクションレスで通信できます。

(1) UDP プロトコルを使用した RPC 通信を行うための条件

UDP プロトコルを使用した RPC 通信を行う場合は，ee_rpc_call 関数の flags 引数に EERPC_UDP を指定します。

また，クライアントおよびサーバが TP1/EE（07-60 以降）であり，かつ TP1/EE サービス定義のプロセス関連定義の xtc_use オペランドに Y を指定していなければなりません。

(2) 定義との関連

UDP プロトコルを使用した RPC 通信（サービスグループ情報関連定義の eesvgdef 定義コマンド）と UDP 通信機能（UDP グループ情報関連定義の eeudpdef 定義コマンド）との関係を次の表に示します。

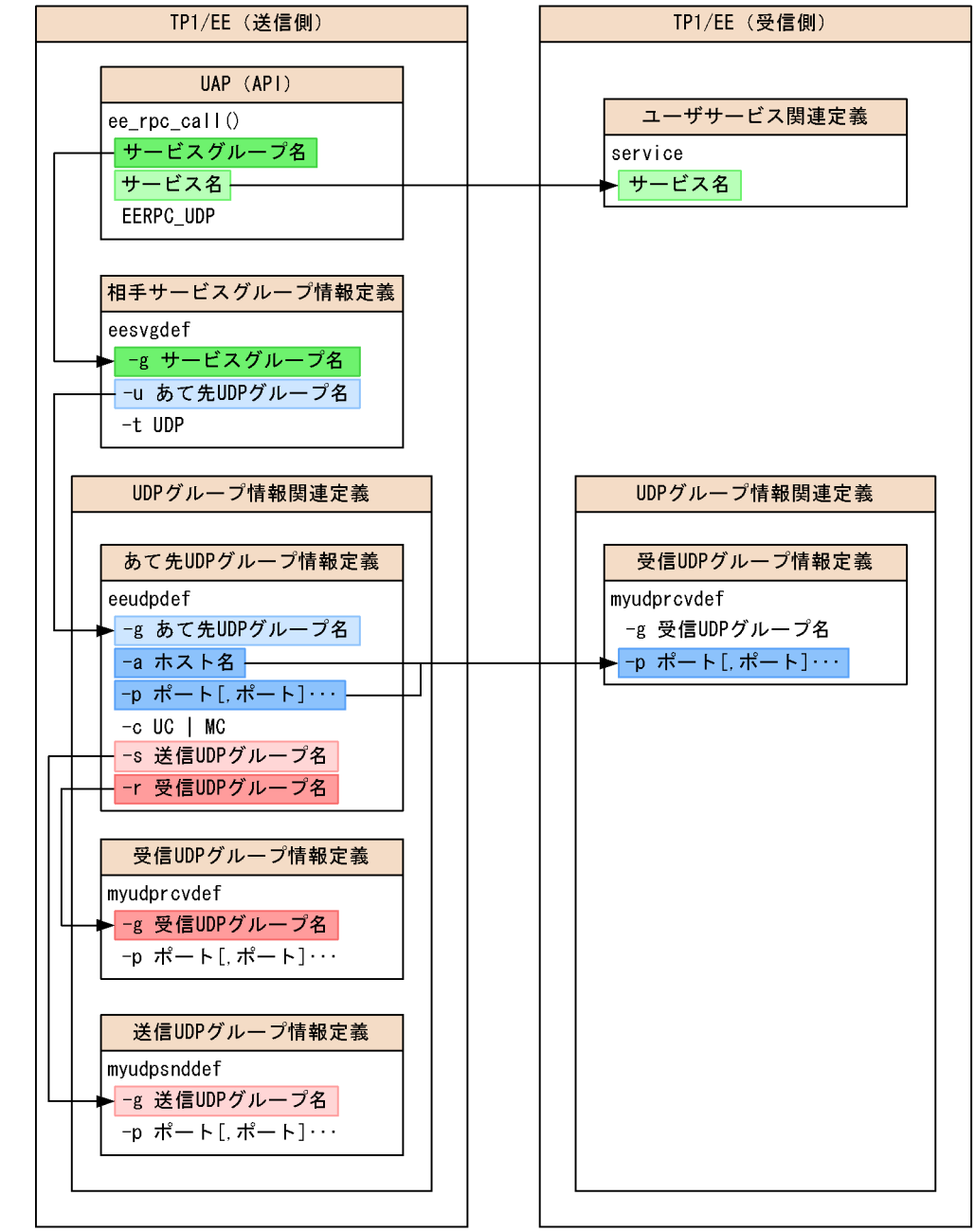
表 2-5 RPC 通信（サービスグループ情報関連定義）と UDP 通信機能（UDP グループ情報関連定義）との関係

サービスグループ情報関連定義	UDP グループ情報関連定義	内容
eesvgdef -g サービスグループ名 -u あて先UDPグループ名 -t UDP	eeudpdef -g あて先UDPグループ名 -a ホスト名 -p ポート番号[,ポート番号]… -c UC | MC	サービス要求先のあて先 UDP グループを指定します。サービス要求先ごとに定義する必要があります。

なお，eesvgdef 定義コマンドの -t オプションに UDP を指定した通信と，-t オプションに RPC を指定した通信を同時に行う場合，それぞれの eesvgdef 定義コマンドには同じサービスグループ名を指定できます。

同期応答型 RPC で通信する場合を例に，API および定義の関連を次の図に示します。

図 2-13　同期応答型 RPC で通信する場合の API および定義の関連



（凡例）

：関連する定義またはオプションを示します。

2.3.2 通信方式

UDP プロトコルを使用した RPC 通信では，ユニキャストおよびマルチキャストによる通信ができます。

RPC サービス要求メッセージは，クライアントからサーバに対して，マルチキャスト，またはユニキャストで送信します。一方，RPC サービス応答メッセージは，サーバからクライアントに対してユニキャストで送信します。

サービス要求先の構成によって，サービスが実行できる通信方式が異なります。RPC サービス要求メッセージの通信方式は，UDP グループ情報関連定義の eeudpdef 定義コマンドの -c オプションで指定します。通信方式とサービスの実行可否の組み合わせを次の表に示します。

表 2-6 通信方式とサービスの実行可否

項番	通信方式	サービス要求先	実行系／待機系	サービスの実行	備考
1	ユニキャスト	HA サーバ	－	できる	－
2		CL サーバ	実行系	できる	－
3			待機系	できない	サービス要求先でメッセージを破棄します。
4	マルチキャスト	HA サーバ	－	できる	－
5		CL サーバ	実行系	できる	－
6			待機系	できない※	サービス要求先でメッセージを破棄します。

（凡例）
－：該当しません。

注※
実行系のサーバでサービスが実行されます。

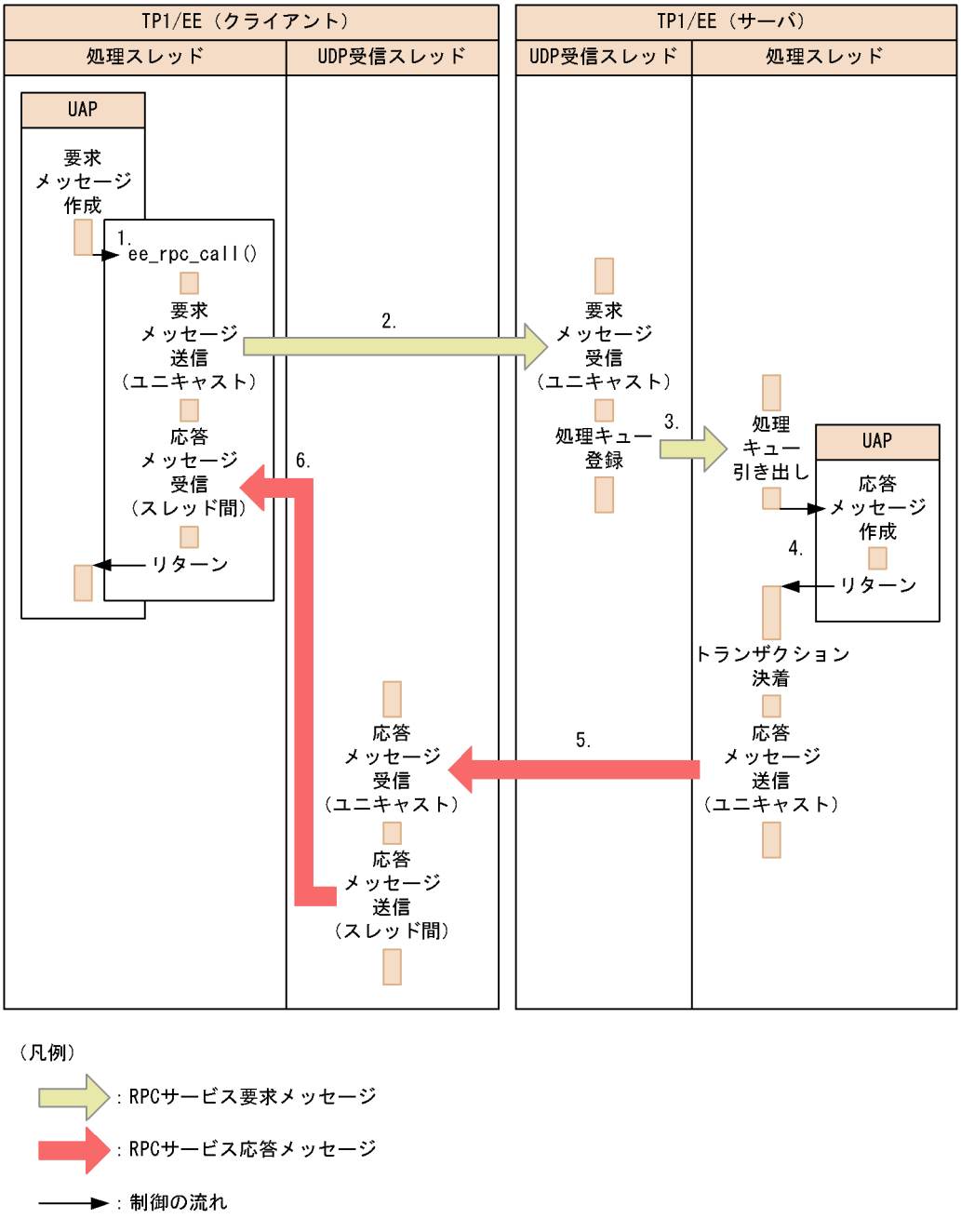
(1) ユニキャストでの通信

サーバに対して，ユニキャストで直接 RPC 要求メッセージを送信します。RPC 要求メッセージを受信したサーバは，サービスを実行します。

RPC 要求メッセージを受信したサーバが CL サーバ（待機系）の場合は，受信したメッセージを破棄します。

同期応答型 RPC によってユニキャストでサービスを要求する場合の処理の流れを次の図に示します。

図 2-14　同期応答型 RPC によってユニキャストでサービスを要求する場合の処理の流れ



説明

1. クライアント UAP によって ee_rpc_call 関数が呼び出されます。
2. 要求メッセージは，ユニキャストでサーバに送信されます。

3. サーバで受信した要求メッセージは，処理キューに登録されます。処理キューは，処理スレッドによって引き出されます。
4. UAP によって応答メッセージが作成され，処理スレッドにリターンします。
5. 処理スレッドは，応答メッセージをユニキャストでクライアントに送信します。
6. 受信スレッドで受信した応答メッセージは，処理スレッドの ee_rpc_call 関数に引き渡されます。

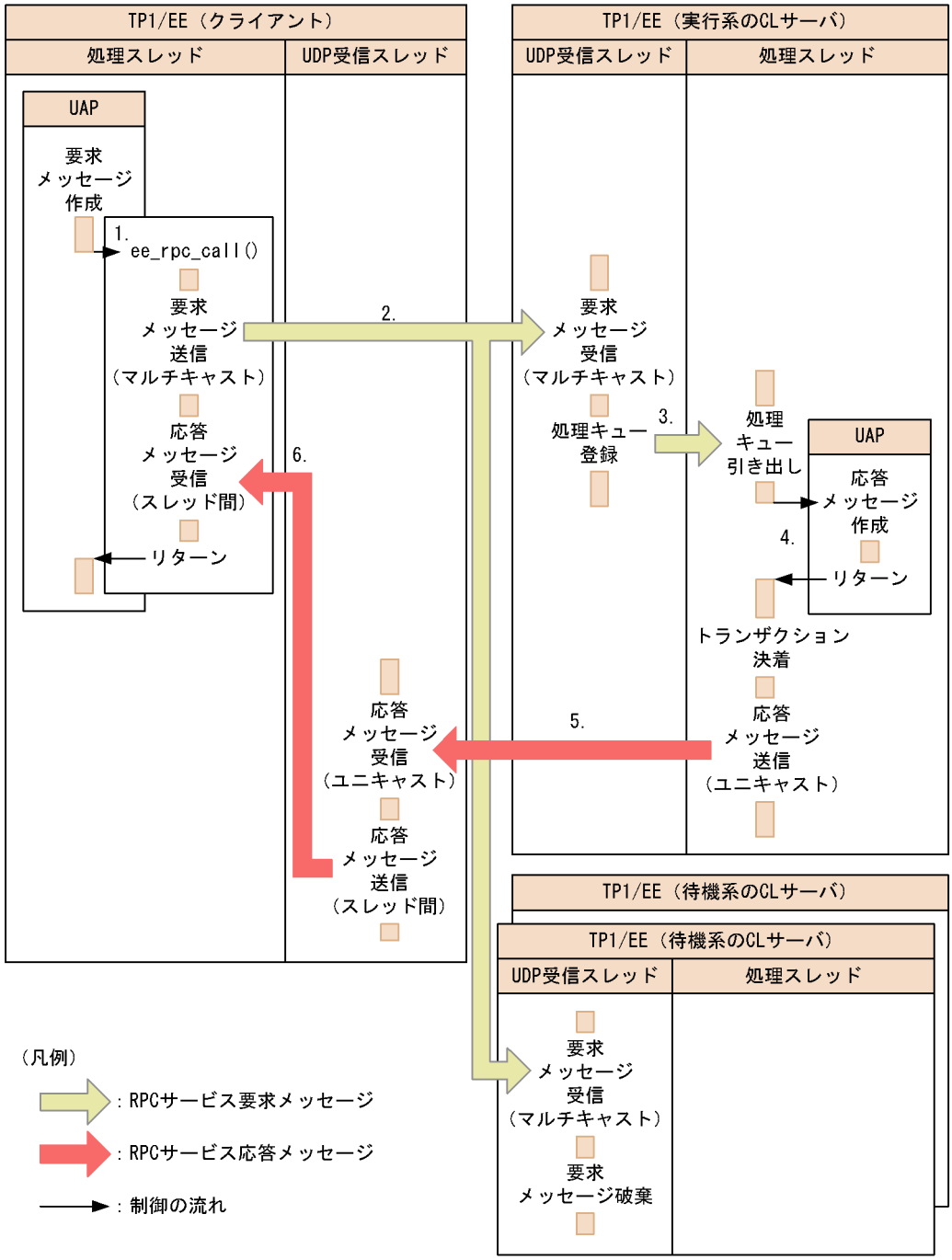
(2) マルチキャストでの通信

複数のサーバに対してマルチキャストで RPC 要求メッセージを送信します。

RPC 要求メッセージを受信したサーバが実行系の場合は，サービスを実行します。一方，待機系が RPC 要求メッセージを受信した場合は，メッセージを破棄します。

同期応答型 RPC によってマルチキャストでサービスを要求する場合の処理の流れを次の図に示します。

図 2-15　同期応答型 RPC によってマルチキャストでサービスを要求する場合の処理の流れ



説明

1. クライアント UAP によって ee_rpc_call 関数が呼び出されます。

2. 要求メッセージは，マルチキャストで CL サーバに送信されます。このとき，待機系の CL サーバでは，受信したメッセージを破棄します。
3. サーバで受信した要求メッセージは，処理キューに登録されます。処理キューは，サーバの処理スレッドによって引き出されます。
4. UAP によって応答メッセージが作成され，処理スレッドにリターンします。
5. 処理スレッドは，応答メッセージをユニキャストでクライアントに送信します。
6. 受信スレッドで受信した応答メッセージは，処理スレッドの ee_rpc_call 関数に引き渡されます。

2.3.3 TCP プロトコルを使用した RPC 通信との差異

TCP プロトコルを使用した RPC 通信と比較して，UDP プロトコルを使用した RPC 通信では，次に示す差異があります。

(1) RPC 通信機能の差異

UDP プロトコルを使用した RPC 通信を行う場合は，次に示す機能を使用できません。

表 2-7 UDP プロトコルを使用した RPC 通信を行う場合の RPC 通信機能の差異

項番	差異	影響するシステム
1	トランザクショナル RPC は使用できません。 ee_rpc_call 関数の flags 引数に EERPC_UDP を指定し，かつ論理和で EERPC_TPNOTRAN を指定しなかった場合，ee_rpc_call 関数は EERPCER_PARAM_FLAGS でエラーになります。	クライアント側
2	RPC 応答のトランザクションまたがり送信機能は使用できません。 使用した場合，ee_rpc_reply_suspend 関数が EERPCER_PROTO でエラーになります。ただし，通常の RPC 通信で起動されたサービス上で RPC 応答メッセージの送信を抑止（ee_rpc_reply_suspend 関数を発行）し，この機能によって起動されたサービス上で抑止解除（ee_rpc_reply_send 関数を発行）することはできます。	サーバ側
3	ERRTRNR による RPC 応答メッセージ送信機能は使用できません。 ERRTRNR による RPC 応答メッセージ送信機能を使用し（rpc_reply_errtrnr オペランドに Y を指定），かつ起動されたサービスから ERRTRNR が起動されたとしても，RPC 応答メッセージはトランザクション決着時に送信されます。	サーバ側
4	サービス要求先の決定方法として，ネームサービスからの検索は使用できません。 ee_rpc_call 関数の flags 引数に EERPC_UDP を指定した場合，RPC 関連定義の rpc_destination_mode オペランドの設定値は無視され，eesvgdef 定義コマンド指定値だけがサービス要求先になります。	クライアント側

項番	差異	影響するシステム
5	次に示すノード間負荷バランス機能は使用できません。 • サービス要求時のノード間負荷バランス機能（クライアント側） • サービス要求受付時のノード間負荷バランス機能（サーバ側） rpc_loadbalance オペランドの指定値は無視されます。 • 他サービスグループあてのサービス要求受付時の転送（サーバ側） rpc_transfer_othersvg オペランドの指定値は無視されます。	クライアント側およびサーバ側

(2) 使用するリソースの差異

UDP プロトコルを使用した RPC 通信を行う場合は，TCP プロトコルを使用した RPC 通信と比較して，使用するリソースに，次に示す差異があります。次に示すリソース以外については，TCP プロトコルを使用した RPC 通信を行う場合と共有します。

表 2-8 UDP プロトコルを使用した RPC 通信を行う場合の使用するリソースの差異

項番	リソース	差異
1	ファイルディスクリプタ	UDP 通信機能のファイルディスクリプタを使用するため，RPC 通信用としてファイルディスクリプタを使用しません。
2	ポート	UDP 通信機能のポートを使用するため，RPC 通信用のポートを使用しません。
3	ソケット送受信バッファサイズ	UDP 通信機能の定義で指定した UDP ソケット送受信バッファサイズになります。
4	フラグメントメッセージ用バッファ	UDP 通信機能のリソースを使用するため，RPC 通信用としてフラグメントメッセージ用バッファを使用しません。

2.3.4 UDP プロトコルを使用した RPC 通信で使用できる API

UDP プロトコルを使用した RPC 通信で使用できる API を次の表に示します。

表 2-9 UDP プロトコルを使用した RPC 通信で使用できる API

項番	API	内容	使用可否
1	ee_rpc_call	遠隔サービスの要求	○※
2	ee_rpc_cmtsend	トランザクションと同期して送信する非応答型 RPC 要求	×
3	ee_rpc_poll_any_replies	非同期応答型 RPC の応答受信	○
4	ee_rpc_discard_further_replies	すべての非同期応答型 RPC の応答受信拒否	○

項番	API	内容	使用可否
5	ee_rpc_discard_specific_reply	特定の非同期応答型 RPC の応答受信拒否	○
6	ee_rpc_get_error_descriptor	エラーが発生した非同期応答型 RPC の識別子取得	○
7	ee_rpc_set_watch_time	サービスの同期応答型 RPC の応答待ち時間の更新	○
8	ee_rpc_get_watch_time	サービスの同期応答型 RPC の応答待ち時間の参照	○
9	ee_rpc_reply_suspend	RPC 応答メッセージ送信の抑止	UDP プロトコルを使用する場合は，抑止できません（API エラーとなります)。
10	ee_rpc_reply_send	抑止していた RPC 応答メッセージの送信	×

（凡例）

○：使用できます。

×：使用できません。

注※

この機能を使用する場合，ee_rpc_call 関数の flags 引数に EERPC_UDP を指定します。

2.3.5 注意事項

UDP 通信機能の仕様上，ee_rpc_call 関数が EERPCER_NET_DOWN エラーとなった場合でも，サーバ側でサービスが実行されている可能性があります。問題がある場合は，UAP でサービス実行の有無を確認してください。

2.4 UDP 通信機能

ここでは，UDP 通信機能について説明します。

UDP 通信機能を使用する通信は，次に示すとおりです。UDP 通信機能の設定を変更する場合は，次に示した通信との整合性が保たれているかどうか確認してください。

- トランザクション同期の一方送信要求（ee_mch_cmtsend_sync 関数）またはトランザクション非同期の一方送信要求（ee_mch_cmtsend 関数）によって通信する場合
- 同一のクラスタグループ内で，サーバ間通信する場合
- UDP プロトコルを使用した RPC 通信を行う場合

2.4.1 UDP プロトコルを使用した通信

UDP 通信機能を使用すると，UDP プロトコルを使用した高速通信ができます。

UDP プロトコルを使用した通信では，メッセージをパケット単位で管理し，1 パケットを 1UDP データグラムとして送受信します。パケットのサイズは，RPC 関連定義の rpc_udp_packet_size オペランドで指定できます。

メッセージが rpc_udp_packet_size オペランドに指定したパケットサイズを超えている場合は，メッセージを複数のパケットに分割して送信します。受信側は，パケットを受信すると，分割された複数のパケットを元のメッセージに組み立てます。

UDP プロトコルを使用した通信では，パケット（分割されたメッセージ）の送信に対して，肯定応答（ACK），否定応答（NACK），およびタイマ監視を行います。これによって，回線障害，回線ビジーなどでメッセージがロストした場合でも，再送制御によってメッセージを保証します。

送信側は，最後のパケットを送信したあと，再送タイマを設定します。再送制御は，ここで設定した再送タイマを使用して行います。再送タイマは，UDP グループ情報関連定義の eeudpdef，clgrpdef，または myudpsnddef 定義コマンドの -w オプション（輻輳制御時は，-W オプション）によって指定します。

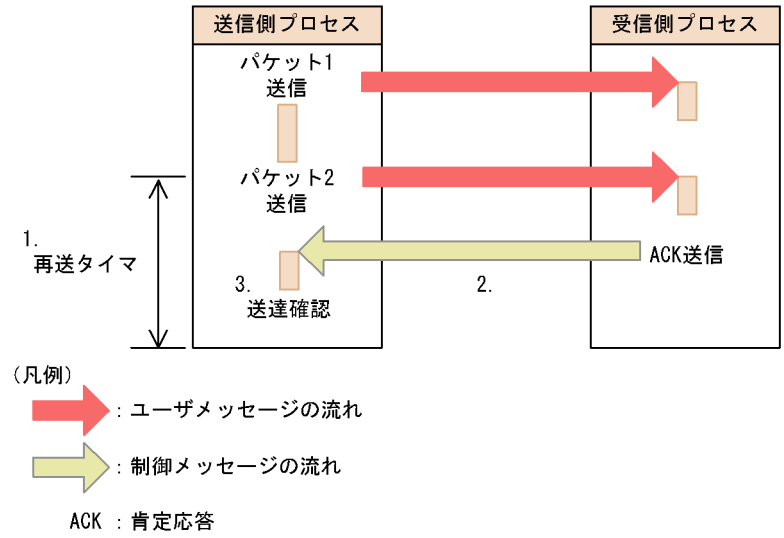
通信処理の流れを次に示すケースに分けて説明します。

- 正常ケース
- 正常ケース（受信側パケットロスでの再送発生）
- 正常ケース（再送タイマでの再送発生）
- 正常ケース（UDP 用受信バッファ（UIBF）の不足）
- 異常ケース（タイムアウト上限超過）

(1) 正常ケース

正常ケースでの通信の流れを次の図に示します。

図 2-16 正常ケースでの通信の流れ



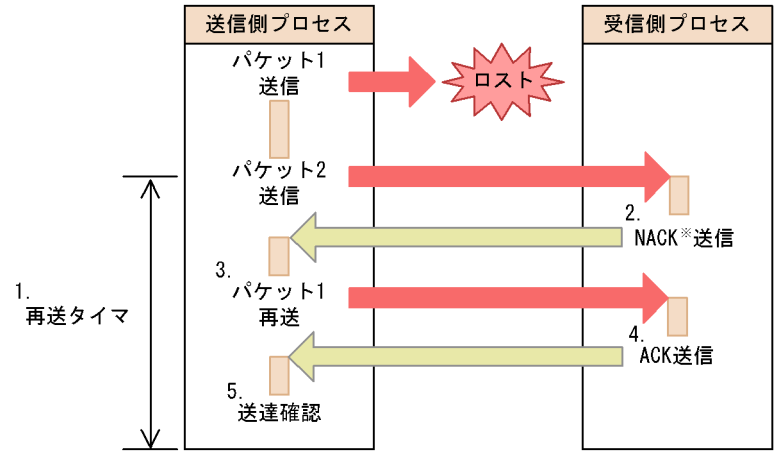
説明

1. 送信側は，パケットを送信したあと，再送タイマを設定します。
2. 受信側は，すべてのパケットを受信したあとに肯定応答を返信します。
3. 送信側は，肯定応答の受信によってメッセージの送達確認を行います。

(2) 正常ケース（受信側パケットロスでの再送発生）

正常ケース（受信側パケットロスでの再送発生）での通信の流れを次の図に示します。

図 2-17　正常ケース（受信側パケットロスでの再送発生）での通信の流れ



（凡例）

：ユーザメッセージの流れ

：制御メッセージの流れ

ACK　：肯定応答

NACK　：否定応答

注※

ロストしたパケットがあった場合，ロストしたパケットの一覧をNACKに格納します。

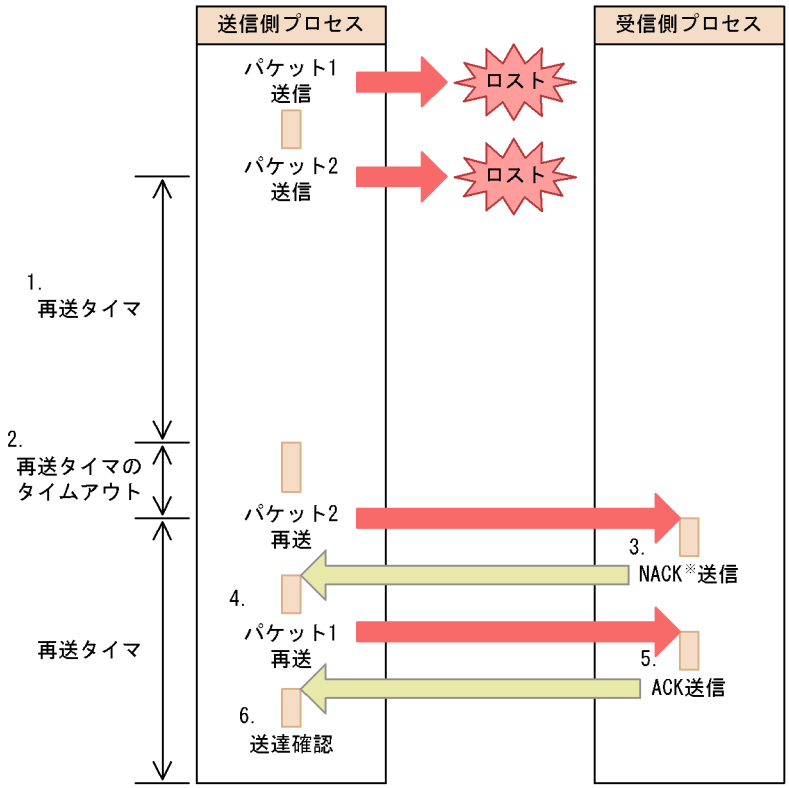
説明

1. 送信側は，パケットを送信したあと，再送タイマを設定します。
2. 受信側は，パケットのロストを検出した場合，否定応答を返信します。
3. 送信側は，否定応答で指示されたパケットを再送します。
4. 受信側は，送信側からの再送によってすべてのパケットを受信したあと，肯定応答を返信します。
5. 送信側は，肯定応答の受信によってメッセージの送達確認を行います。

（3）正常ケース（再送タイマでの再送発生）

正常ケース（再送タイマでの再送発生）での通信の流れを次の図に示します。

図 2-18　正常ケース（再送タイマでの再送発生）での通信の流れ



（凡例）

：ユーザメッセージの流れ

：制御メッセージの流れ

ACK　：肯定応答

NACK　：否定応答

注※

ロストしたパケットがあった場合，ロストしたパケットの一覧をNACKに格納します。

説明

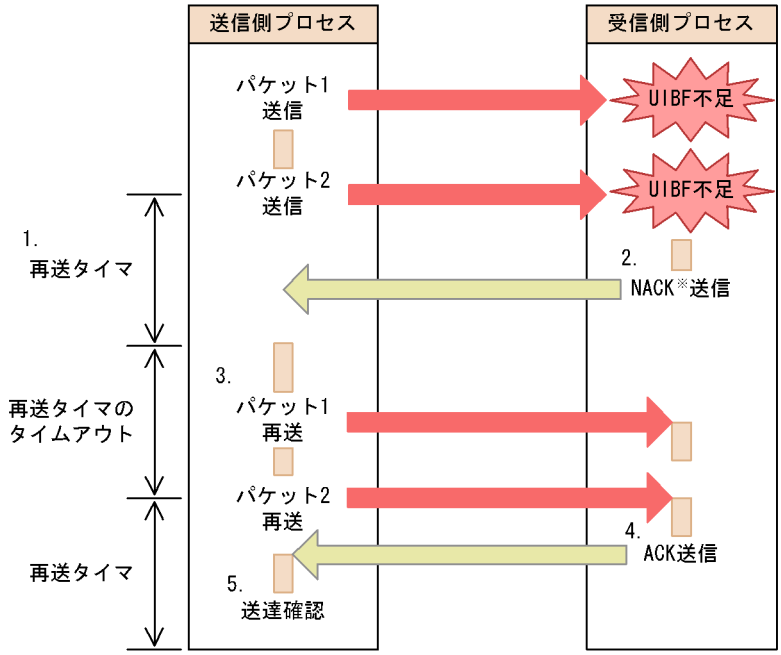
1. 送信側は，パケットを送信したあと，再送タイマを設定します。
2. 最後に送信したパケットがロストした場合（または送信先からの送達確認がロストした場合），送信側は，再送タイマで設定した時間まで待ったあとタイムアウトします。このとき，送信側は，最後に送信したパケットを再送して，再び再送タイマを設定します。
3. 受信側は，パケットのロストを検出した場合，否定応答を返信します。
4. 送信側は，否定応答で指示されたパケットを再送します。
5. 受信側は，送信側からの再送によってすべてのパケットを受信したあと，肯定応答を返信します。

6. 送信側は，肯定応答の受信によってメッセージの送達確認を行います。

(4) 正常ケース（UDP 用受信バッファ（UIBF）の不足）

正常ケース（UDP 用受信バッファ（UIBF）の不足）での通信の流れを次の図に示します。

図 2-19 正常ケース（UDP 用受信バッファ（UIBF）の不足）での通信の流れ



（凡例）

：ユーザメッセージの流れ

：制御メッセージの流れ

UIBF ：UDP用受信バッファ

ACK ：肯定応答

NACK ：否定応答

注※

UDP用受信バッファ不足によって受信できなかったパケットがあった場合，受信できなかったパケットの一覧をNACKに格納します。

説明

1. 送信側は，パケットを送信したあと，再送タイマを設定します。
2. 受信側は，UDP 用受信バッファ不足によって受信できなかったパケットがあった場合，否定応答を返信します。
3. 送信側は，否定応答を受信してもすぐには再送しないで，再送タイマのタイムア

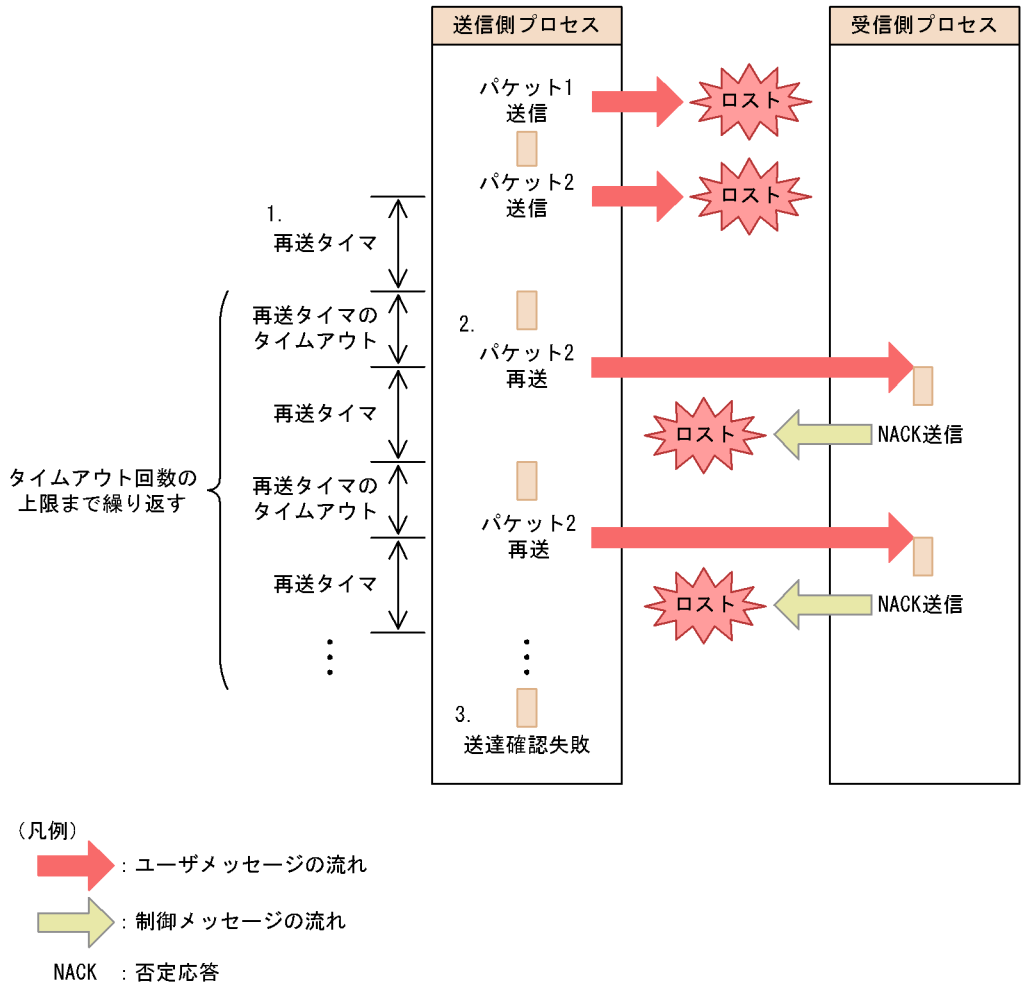
ウト時に否定応答で指示されたパケットを再送します。

4. 受信側は，送信側からの再送によってすべてのパケットを受信したあと，肯定応答を返信します。
5. 送信側は，肯定応答の受信によってメッセージの送達確認を行います。

(5) 異常ケース（タイムアウト上限超過）

異常ケース（タイムアウト上限超過）での通信の流れを次の図に示します。

図 2-20 異常ケース（タイムアウト上限超過）での通信の流れ



説明

1. 送信側は，パケットを送信したあと，再送タイマを設定します。
2. 最終パケットがロストした場合，または受信側からの肯定応答，否定応答がロストした場合，再送タイマがタイムアウトします。この場合，最終パケットの再送後，再度，タイマを設定します。

3. タイムアウト回数がタイムアウト上限に達した場合は送達確認失敗となり，送信エラーになります。

2.4.2 輻輳制御

通信回線上で輻輳が発生した場合は，ネットワーク機器のどれかのバッファがオーバフローを起こすため，受信側のバッファに格納されるはずの UDP パケットが破棄されます。

輻輳が頻繁に起こる場合は，受信側から見ると連続してパケットロスが発生するため，再送要求の回数が増えます。また，送信側がこの再送要求どおりに再送を行うと，ネットワークのトラフィックをさらに増加させるため，輻輳が解消できません。

そのため，XTC では送信側が輻輳制御を行うことで輻輳を解消します。なお，輻輳制御は，送信元スレッド単位に行われます。

(1) 輻輳制御の開始と終了

輻輳制御の開始と終了について説明します。

■輻輳制御の開始

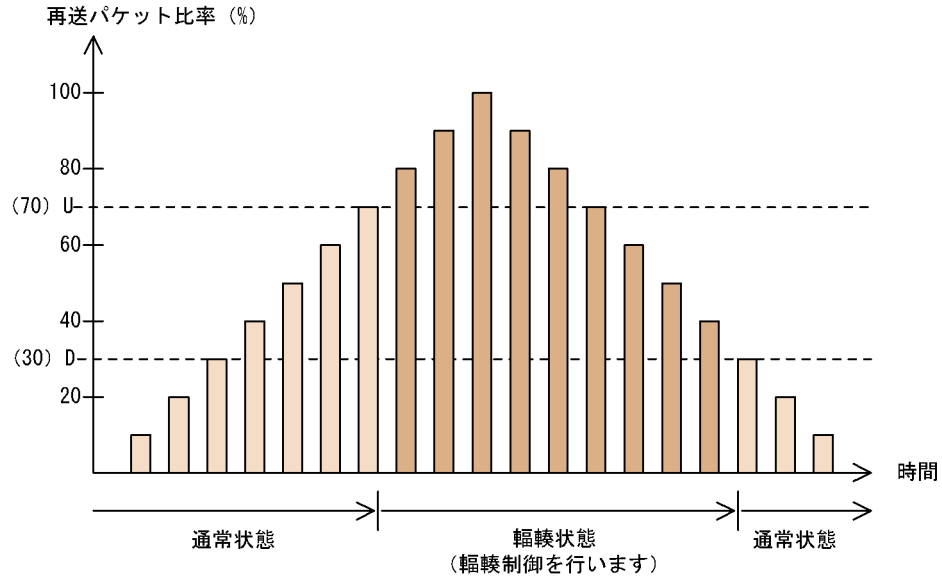
監視対象の範囲内（RPC 関連定義の rpc_udp_congestion_packet_count オペランドの指定値）に送信したパケットについて，再送パケットの比率がパケットロス上限（UDP グループ情報関連定義の myudpsnddef 定義コマンドの -L オプションの指定値）以上になった場合に輻輳制御を開始します。

■輻輳制御の終了

監視対象の範囲内（RPC 関連定義の rpc_udp_congestion_packet_count オペランドの指定値）に送信したパケットについて，再送パケットの比率がパケットロス下限（UDP グループ情報関連定義の myudpsnddef 定義コマンドの -l オプションの指定値）以下になった場合に輻輳制御を終了します。

再送パケット比率が 0% → 100% → 0% と遷移し，かつ，パケットロス上限に 70，パケットロス下限に 30 が指定された場合の輻輳状態の遷移例を次の図に示します。

図 2-21　輻輳状態の遷移例



（凡例）

U：パケットロス上限の比率

D：パケットロス下限の比率

（2）輻輳制御の方法

輻輳を検知した直後に輻輳制御を開始します。輻輳制御を開始した時点で仕掛かり中のメッセージについても輻輳制御によって再送します。輻輳制御では，次に示す方法によって，トラフィックを軽減します。

■肯定応答が未受信で連続送信できるパケット数の縮小

通常状態では，メッセージの最終パケットを送信した時点で再送タイマを設定します。しかし，輻輳状態では，最終パケットだけではなく，中間パケットを送信した時点でも再送タイマを設定します。

肯定応答待ちは，再送パケット比率を基に自動的に決定したパケット数のパケット送信ごとに行うようにします。肯定応答待ちを行うパケット数の最低値は UDP グループ情報関連定義の myudpsnddef 定義コマンドの -U オプションで指定します。

この制御によって，連続送信によるトラフィック増加を回避できます。

■否定応答の受信時の再送契機

通常状態では，否定応答を受信した場合，すぐに再送パケットを送信します。

輻輳状態では，否定応答を受信した場合，再送タイマのタイムアウト時に再送パケットを送信します。

■再送タイマの変更

通常状態の場合，再送タイマのタイマ値には，UDP グループ情報関連定義の

eeudpdef，clgrpdef，または myudpsnddef 定義コマンドの -w オプションの指定値を使用します。
輻輳状態の場合，再送タイマのタイマ値には，eeudpdef，clgrpdef，または myudpsnddef 定義コマンドの -W オプション指定値を使用します。ただし，-W オプションを指定していない場合は，タイマ値を変更しません。
それぞれの定義コマンドに異なる値を指定した場合の優先順位は次のとおりです。

- 同一のクラスタグループ内で通信する場合
  clgrpdef 定義コマンドの指定値＞ myudpsnddef 定義コマンドの指定値
- 同一のクラスタグループ外と通信する場合
  eeudpdef 定義コマンドの指定値＞ myudpsnddef 定義コマンドの指定値

2.4.3 複数メッセージの一括送受信機能

複数のメッセージを結合し，一つの論理メッセージとすることで，論理メッセージ単位での送達確認を行います。これを，複数メッセージの一括送受信機能といいます。複数のメッセージを一括して送受信することで，送達確認の頻度が減るため，通信時間を短縮できます。

通常のメッセージ送受信時の送達確認と，複数メッセージの一括送受信機能を使用した場合の送達確認との比較を次の図に示します。

図 2-22 通常のメッセージ送受信時の送達確認

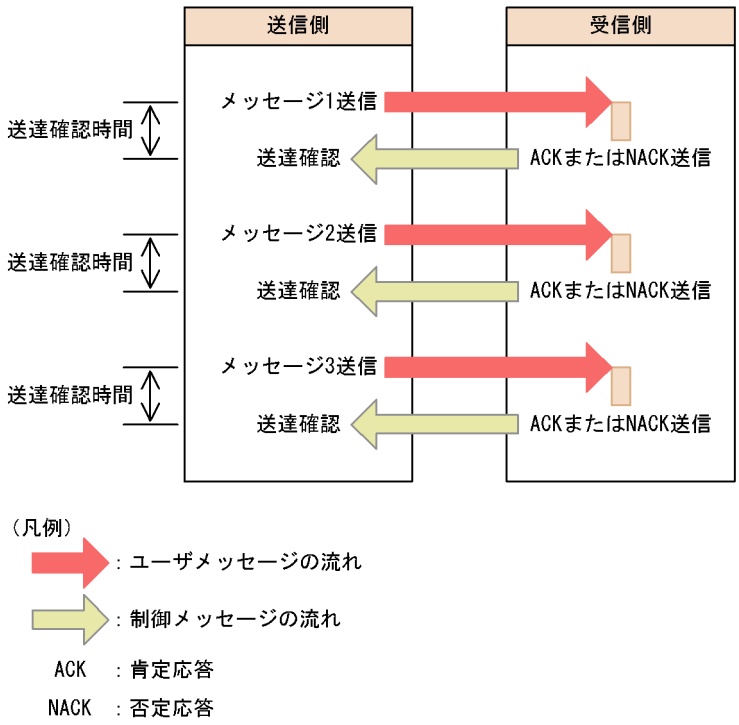
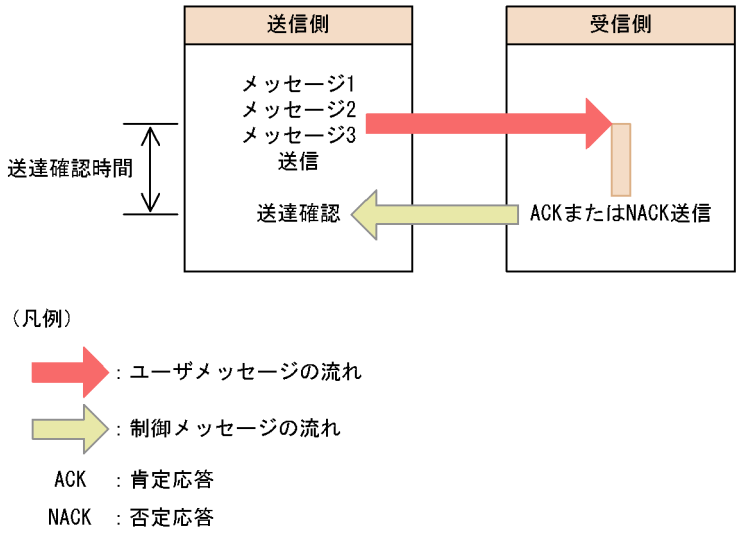


図 2-23　複数メッセージの一括送受信機能を使用した場合の送達確認



なお，一つの論理メッセージの最大サイズは，最大で 9 メガバイトです。メッセージの合計サイズが 9 メガバイトを超える場合は，複数の論理メッセージになります。一つのメッセージが複数の論理メッセージに分割されることはありません。

この機能は，次のときに使用できます。

- 一方送信メッセージの送受信時
  一方送信メッセージについては，「2.2　高速メッセージ送信制御」を参照してください。
- CL サーバでの CL 同期メッセージの送受信時
  CL 同期メッセージについては，「7.7.2　CL 同期メッセージの処理の流れ」を参照してください。

(1) 論理メッセージの送達確認範囲

論理メッセージの送達確認は，一定の範囲ごとに行われます。ここでは，送達確認範囲およびその見積もり方法について説明します。

論理メッセージの送達確認は，送達確認パケット数ごとに行われます。ただし，送達確認後の残りパケット数が，送達確認パケット数の 20% 未満となる場合は，残りのすべてのパケットに対して送達確認を行います。論理メッセージのパケット数が，送達確認パケット数以下の場合は，すべてのパケットに対して一度の送達確認を行います。

送達確認パケット数は，RPC 関連定義の rpc_udp_lmsg_deliverychk_size オペランドの指定値（省略時解釈値：512 キロバイト）を rpc_udp_packet_size オペランドの指定値（省略時解釈値：1472 バイト）で除算した値です。

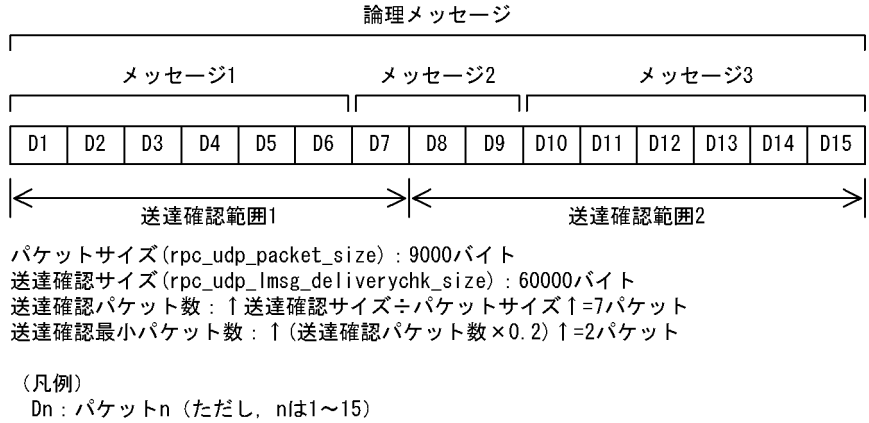
送達確認が完了した時点で，送達確認範囲で完成（最終パケットのパケット送信完了）

しているメッセージは，送受信完了となります。輻輳状態となった場合は，送達確認範囲単位に輻輳制御を行います。

送達確認範囲が小さい場合は，送達確認が多発するため，通信遅延の原因となります。一方，送達確認範囲が大きい場合は，パケットロスなどによる通信障害時に，再送による通信量が増加します。そのため，送信側の UDP グループ情報関連定義の myudpsnddef 定義コマンドの -b オプションの指定値と，受信側の myudprcvdef または clgrpdef 定義コマンドの -B オプションの指定値のうち，小さい方の値以下に指定することをお勧めします。

論理メッセージサイズが送達確認サイズを超過している場合の送達確認範囲を次の図に示します。

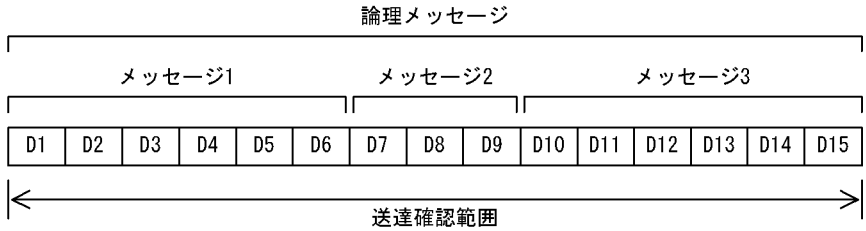
図 2-24　論理メッセージサイズが送達確認サイズを超過している場合



この図では，論理メッセージを構成するパケットを送達確認パケット数で区切った場合に，1 パケット（D15）が余ります。しかし，送達確認の最小パケット数は 2 であるため，D15 は送達確認範囲 2 に含めます。

次に，論理メッセージサイズが送達確認サイズ以下の場合の送達確認範囲を次の図に示します。

図 2-25　論理メッセージサイズが送達確認サイズ以下の場合



パケットサイズ(rpc_udp_packet_size)：9000バイト
送達確認サイズ(rpc_udp_lmsg_deliverychk_size)：600000バイト
送達確認パケット数：↑送達確認サイズ÷パケットサイズ↑＝67パケット
送達確認最小パケット数：↑(送達確認パケット数×0.2)↑＝14パケット

（凡例）
Dn：パケットn（ただし，nは1～15）

この図では，論理メッセージが送達確認パケット数以下であるため，一度に送達確認できます。

(2) 単メッセージの送達確認範囲の変更機能

単メッセージとは，一つのメッセージから成る論理メッセージです。

通常，送達確認はパケットの送信後に行われます。しかし，メッセージサイズより送信先のソケット受信バッファサイズが小さい場合，送信先の UDP ソケット受信バッファ不足が発生し，パケットロスするおそれがあります。ロストしたパケットは，再送処理によって保証されます。しかし，送信するパケット数が増加するため，通信遅延が発生したり，CPU やネットワークの負荷が増大したりします。そのため，XTC では，単メッセージの場合の送達確認範囲を変更できます。

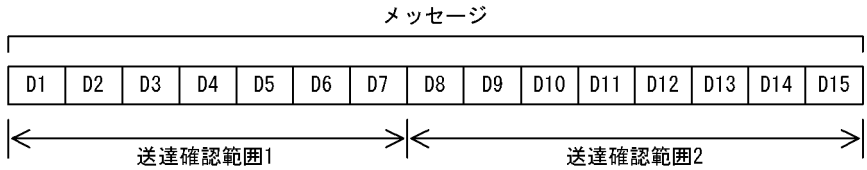
送達確認範囲を変更する場合，RPC 関連定義の rpc_udp_msg_deliverychk_size オペランドの指定値（省略時解釈値：2 メガバイト）を RPC 関連定義の rpc_udp_packet_size オペランドの指定値（省略時解釈値：1472 バイト）で除算した値を送達確認パケット数として，送達確認パケット数ごとに送達確認を行います。ただし，送達確認後の残パケット数が，送達確認パケット数の 20% 未満となる場合は，残りのすべてのパケットに対して送達確認を行います。メッセージのパケット数が，送達確認パケット数以下の場合は，すべてのパケットに対して一度に送達確認を行います。これによって，連続送信するパケット数が減少し，パケットのロスト頻度も減少します。輻輳状態となった場合は，送達確認範囲単位に輻輳制御を行います。

送達確認範囲が小さい場合は，送達確認が多発するため通信遅延の原因となります。一方，送達確認範囲が大きい場合は，パケットロスなどによる通信障害時に，再送による通信量が増加します。そのため，送信元の UDP グループ情報関連定義の myudpsnddef 定義コマンドの -b オプションの指定値，および送信先の myudprcvdef 定義コマンド，または clgrpdef 定義コマンドの -B オプションの指定値のうち，小さい方の値以下に指定

することをお勧めします。

メッセージサイズが送達確認サイズを超過している場合の送達確認範囲を次の図に示します。

図 2-26　メッセージサイズが送達確認サイズを超過している場合



パケットサイズ(rpc_udp_packet_size)：9000バイト
送達確認サイズ(rpc_udp_msg_deliverychk_size)：60000バイト
送達確認パケット数：↑送達確認サイズ÷パケットサイズ↑＝7パケット
送達確認最小パケット数：↑(送達確認パケット数×0.2)↑＝2パケット

（凡例）
Dn：パケットn（ただし，nは1～15）

この図では，メッセージを構成するパケットを送達確認パケット数で区切った場合に，1パケット（D15）が余ります。しかし，送達確認の最小パケット数は 2 であるため，D15は送達確認範囲 2 に含めます。

次に，メッセージサイズが送達確認サイズ以下の場合の送達確認範囲を次の図に示します。

図 2-27　メッセージサイズが送達確認サイズ以下の場合

パケットサイズ(rpc_udp_packet_size)：9000バイト
送達確認サイズ(rpc_udp_msg_deliverychk_size)：600000バイト
送達確認パケット数：↑送達確認サイズ÷パケットサイズ↑＝67パケット
送達確認最小パケット数：↑(送達確認パケット数×0.2)↑＝14パケット

（凡例）
Dn：パケットn（ただし，nは1～15）

この図では，メッセージが送達確認サイズ以下であるため，一度に送達確認できます。

2.4.4　W-send 機能

XTC では，ネットワーク経路を二重化した構成で hbonding ドライバ（HA Network

Driver Mirroring Edition for Linux）を適用し，二つの経路から同一のパケットを送受信できます。これを W-send 機能といいます。W-send 機能を使用するためには，hbonding ドライバのバージョンが 02-01 以降であり，hbonding ドライバの動作モードに broadcast モードを選択する必要があります。

W-send 機能を使用した場合のメリットは次のとおりです。

- ネットワーク切り替え時間の短縮
- UDP パケットロスの補償

W-send 機能を使用した場合，受信側では，二つの経路から同一のパケットを重複して受信します。このとき，重複して受信したパケット（重複受信パケット）は，hbonding ドライバまたは XTC によって破棄されます。

hbonding ドライバで重複受信パケットを破棄する場合は，hbonding ドライバの重複受信パケットフィルタリング機能を使用します。XTC で受信パケットを破棄する場合と比較して次のメリットがあるため，hbonding ドライバで重複受信パケットを破棄することをお勧めします。

- CPU 負荷の軽減
- UDP ソケット受信バッファの使用量の削減

hbonding ドライバの重複受信パケットフィルタリング機能を使用しない場合，または重複受信パケットフィルタリング機能で破棄できなかった場合は，XTC によって重複受信パケットが破棄されます。

2.4.5 UDP 通信機能で使用するリソース

UDP 通信機能で使用するリソースについて説明します。

(1) ポート

UDP 通信機能で使用するポートは，目的に応じて送信用ポート，または受信用ポートと呼ばれます。

ポートごとに異なるソケット属性を指定する場合（例えば，UDP を使用した RPC 通信と高速メッセージ送信制御でポート番号を切り替えたい場合など）は，UDP グループを複数定義して，異なる属性を指定します。

ポート種別による定義の違いを次の表に示します。

表 2-10 ポート種別による定義の違い

項番	ポート種別	内容
1	送信用ポート	メッセージの送信に使用し，送信 UDP グループとして管理します。UDP グループ情報関連定義の myudpsnddef 定義コマンドで定義します。

項番	ポート種別	内容
2	受信用ポート	メッセージの受信，ならびに肯定応答および否定応答の送受信に使用し，受信 UDP グループとして管理します。UDP グループ情報関連定義の myudprcvdef 定義コマンドまたは clgrpdef 定義コマンドで定義します。

(2) UDP 用送受信バッファ

■ UDP 用送信バッファ

UDP 通信機能が使用する UDP 用送信バッファのサイズと面数は，メモリ関連定義の udp_send_message_buf_size オペランドおよび udp_send_message_buf_cnt オペランドで指定します。

■ UDP 用受信バッファ

UDP 通信機能が使用する UDP 用受信バッファのサイズと面数は，メモリ関連定義の udp_recv_message_buf_size オペランドおよび udp_recv_message_buf_cnt オペランドで指定します。

(3) UDP ソケット送受信バッファ

■ UDP ソケット送信バッファ

送信 UDP データグラムを一時的に格納する OS のバッファです。送信パケットサイズ（RPC 関連定義の rpc_udp_packet_size オペランド指定値 + 28）以上を指定します。また，次の計算式を満たすサイズに指定することをお勧めします。

UDPソケット送信バッファ ≧ RPC関連定義のrpc_udp_lmsg_deliverychk_sizeオペランド，または rpc_udp_msg_deliverychk_sizeオペランドの指定値

■ UDP ソケット受信バッファ

受信 UDP データグラムを一時的に格納する OS のバッファです。受信パケットサイズ（RPC 関連定義の rpc_udp_packet_size オペランド指定値 + 28）以上を指定します。また，次の計算式を満たすサイズに指定することをお勧めします。

UDPソケット受信バッファ ≧ 送信するユーザデータのサイズ※1 **× 1.1 × 送信元スレッド数**※2

注※ 1

CL 同期メッセージの送受信だけで使用するソケットの場合に，ユーザデータのサイズが RPC 関連定義の rpc_udp_lmsg_deliverychk_size オペランドの指定値以上になるときは，オペランド指定値はユーザデータのサイズになります。
上記以外のメッセージの場合に，ユーザデータのサイズが RPC 関連定義の rpc_udp_msg_deliverychk_size オペランドの指定値以上になるときは，オペランド指定値はユーザデータのサイズになります。

注※ 2

UDP ソケットに対して，同時送信を行う送信元のスレッド数です。

輻輳によるパケットロスを減らすには，UDP ソケット受信バッファを大きく設定する必要があります。各送信元スレッドからのメッセージ一つ分を格納するソケット受信バッファを設定することをお勧めします。

なお，W-send 機能を使用している場合に hbonding ドライバで重複受信パケットを破棄しないときは，UDP ソケット受信バッファを上記の計算式で算出されるサイズの 2 倍に設定することをお勧めします。

2.4.6 ポート構成

通信相手が同一のクラスタグループ内のサーバであるかどうかによって，ポート構成が異なります。ここでは，同一のクラスタグループ内のサーバと通信する場合，および同一のクラスタグループ外のサーバと通信する場合について，ポート構成例を説明します。

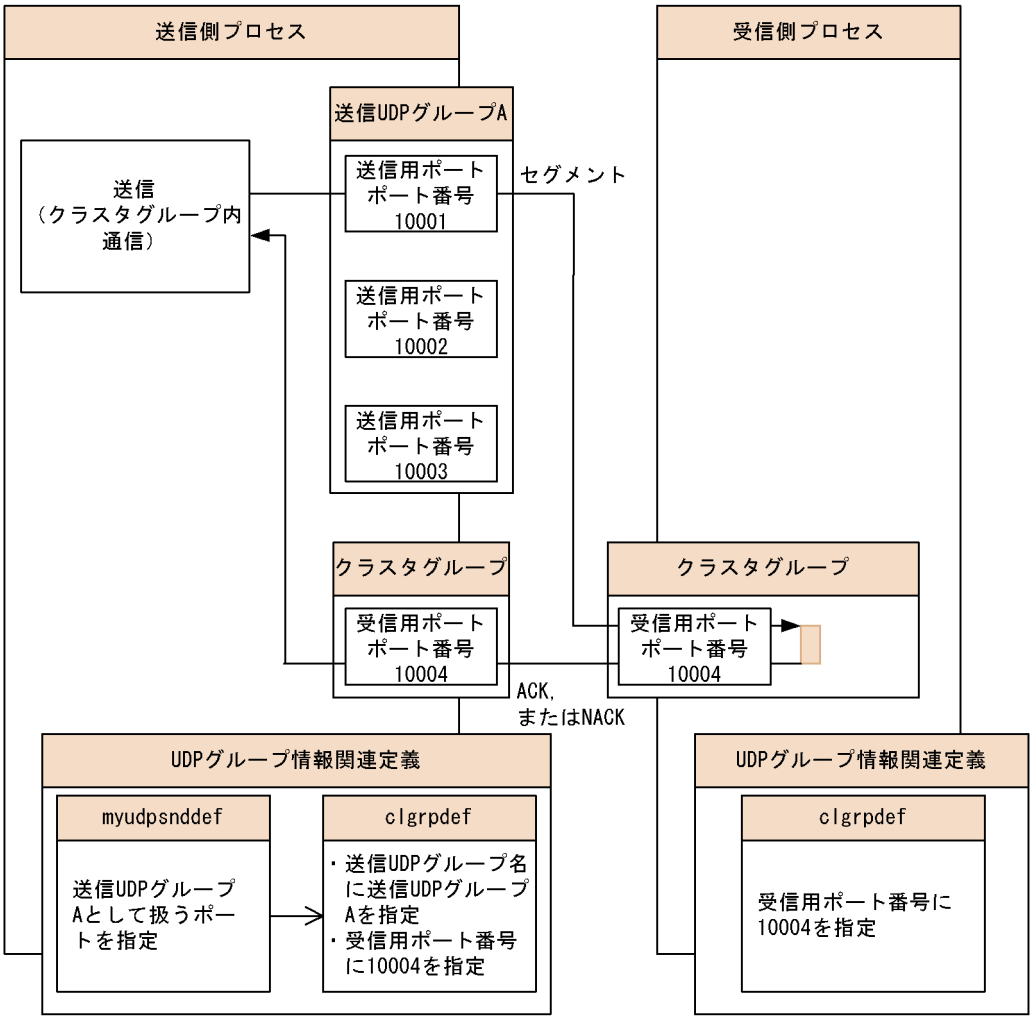
(1) 同一のクラスタグループ内のサーバと通信する場合の構成

クラスタグループ内の各サーバには，クラスタグループ定義の clgrpdef 定義コマンドを定義する必要があります。クラスタグループ定義では，同一のクラスタグループ内での通信で使用するポート番号，マルチキャストアドレス，送信 UDP グループなどを指定します。各サーバのクラスタグループ定義は，次の項目を除いて同じ値を指定します。

- 送信 UDP グループ
- LAN アダプタのホスト名

同一のクラスタグループ内で通信する場合の定義例を次の図に示します。

図 2-28　同一のクラスタグループ内で通信する場合の定義例



（凡例）

：クラスタグループ内通信の流れ

：定義の関連

ACK　：肯定応答

NACK　：否定応答

説明

送信側プロセスでは，次のように定義します。

- UDP グループ情報関連定義の myudpsnddef 定義コマンドに，送信 UDP グループ A として扱うポート番号，10001，10002，および 10003 を指定します。
- UDP グループ情報関連定義の clgrpdef 定義コマンドの送信 UDP グループ名に，送信 UDP グループ A を指定します。また，受信用ポートのポート番号として

10004 を指定します。

受信側プロセスでは，次のように定義します。

- clgrpdef 定義コマンドで受信用のポート番号として 10004 を指定します。

(2) 同一のクラスタグループ外のサーバと通信する場合の構成

同一のクラスタグループ外のサーバと通信する場合（例えば，HA サーバと CL サーバ間，CL サーバと別の CL サーバ間での通信）は，通信先情報をあて先 UDP グループ（受信側プロセスにとっての受信 UDP グループ）として，eeudpdef 定義コマンドで定義します。

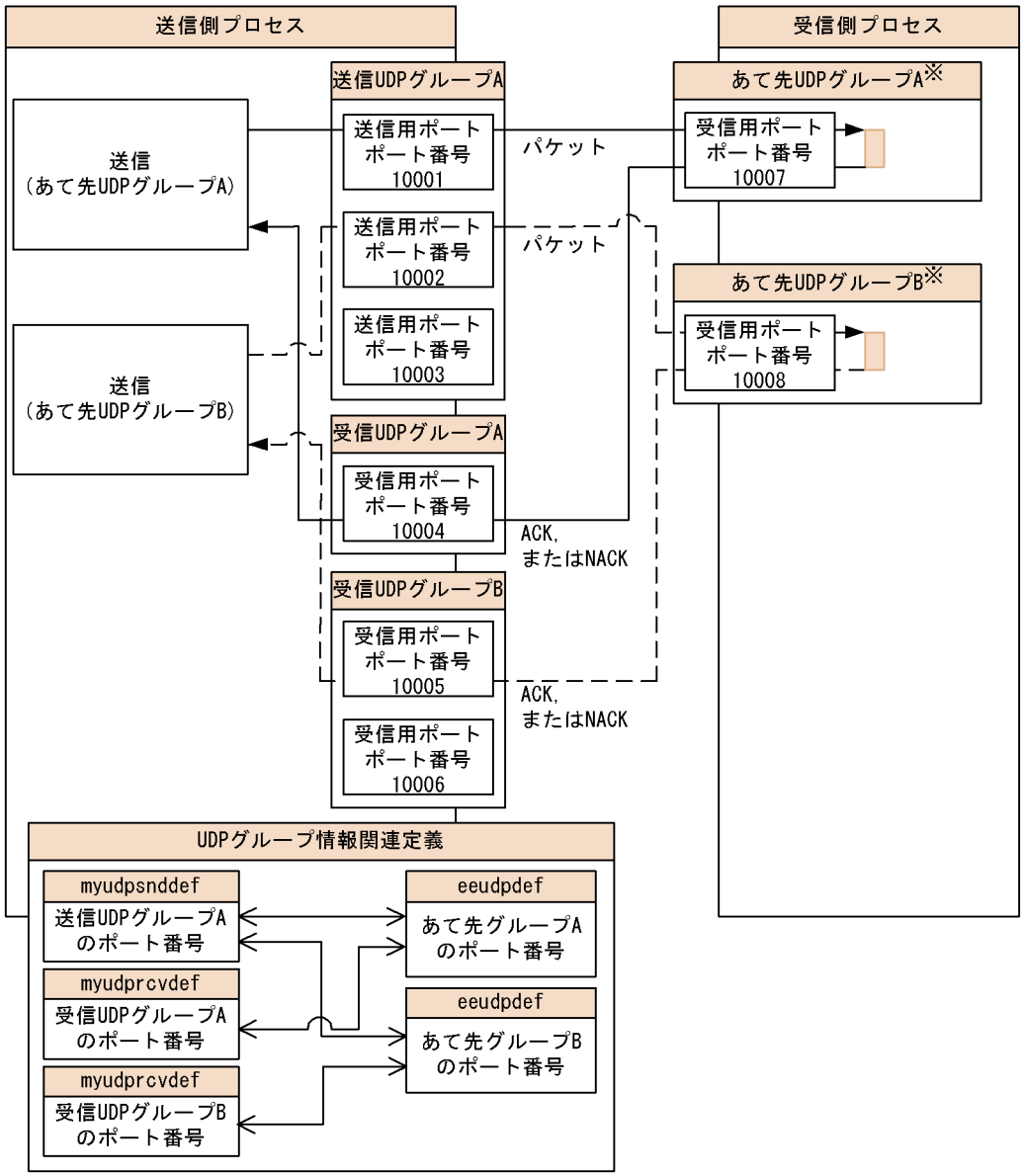
あて先 UDP グループを定義する場合，該当するあて先グループへのメッセージ送信に使用する送信 UDP グループ，および該当するあて先グループからの送達確認受信で使用する受信 UDP グループを定義します。省略した場合は，デフォルトの送信 UDP グループ，および受信 UDP グループが使用されます。

同一のクラスタグループ外のサーバと通信する場合の定義の組み合わせ例を，次の表および図に示します。

表 2-11 同一のクラスタグループ外のサーバと通信する場合の定義の組み合わせ例

組み合わせ	あて先 UDP グループ名	送信 UDP グループ名	受信 UDP グループ名
1	あて先 UDP グループ A	送信 UDP グループ A	受信 UDP グループ A
2	あて先 UDP グループ B	送信 UDP グループ A	受信 UDP グループ B

図 2-29　同一のクラスタグループ外のサーバと通信する場合の定義の組み合わせ例



（凡例）

——►：組み合わせ1の場合のあて先UDPグループとの通信の流れ

– – ►：組み合わせ2の場合のあて先UDPグループとの通信の流れ

——>：定義内でのポート番号の関連（それぞれの定義のポート番号を合わせる必要があります）

ACK　：肯定応答

NACK　：否定応答

注※

受信側プロセス上では，受信UDPグループになります。

説明

送信側プロセスでは，次のように定義します。

- UDP グループ情報関連定義の myudpsnddef 定義コマンドに，送信 UDP グループ A として扱うポート番号 10001，10002，および 10003 を指定します。
- UDP グループ情報関連定義の myudprcvdef 定義コマンドを二つ定義し，それぞれの定義に受信 UDP グループ A としてポート番号 10004，受信 UDP グループ B としてポート番号 10005，および 10006 を指定します。
- UDP グループ情報関連定義の eeudpdef 定義コマンドを二つ定義し，それぞれの定義のあて先 UDP グループ名に，あて先 UDP グループ A，およびあて先 UDP グループ B を指定します。また，それぞれのあて先 UDP グループに対して，送受信に利用する送信 UDP グループおよび受信 UDP グループを指定します。

受信側プロセスでは，次のように定義します。

- eeudpdef 定義コマンドで受信用のポート番号として 10007 および 10008 を指定します。

(3) マルチキャストループバック受信（同一のクラスタグループ内での自マシンに対するマルチキャスト送信）有無

マルチキャストループバック受信とは，マルチキャストでメッセージを送信する場合に，自マシンを送信対象とすることで，自マシンから送信したメッセージを自マシンで受信することをいいます。マルチキャストループバック受信を行うかどうかは，UDP グループ情報関連定義の myudpsnddef 定義コマンドで選択できます。

■マルチキャストループバック受信をありに設定する場合

マルチキャストによるサービス要求時に，該当するサービスを処理する XTC が同一マシン上に存在する場合は，ループバック受信をあり（-k オプションを指定）に設定してください。

■マルチキャストループバック受信をなしに設定する場合

CL サーバの場合，ループバック受信をありに設定すると，自マシンから送信したメッセージを自マシンで受信し，破棄するという不要な受信処理が発生します。そのため，ループバック受信をなし（-k オプションを省略）に設定することをお勧めします。

上記の二つの条件が重なった場合，次のどちらかの対策をする必要があります。

- ループバック受信ありの送信 UDP グループを一つ定義し，サービス要求送信用および CL サーバ間での通信用の定義として共有する。
- ループバック受信ありとループバック受信なしの送信 UDP グループを二つ以上定義し，ループバック受信ありはサービス要求送信用，ループバック受信なしは CL サーバ間での通信用というように用途ごとに割り当てる。

2.4.7 UDP 通信機能で使用するタイマ

UDP 通信機能で使用するタイマを，次の表に示します。

表 2-12 UDP 通信機能で使用するタイマ

項番	タイマ種別	タイマ設定側	内容
1	再送タイマ	送信側	メッセージを構成するすべてのパケットを送信してから送達確認を受信するまでのタイマです。タイムアウト時は再送制御を行います。タイムアウト回数が定義した値を超過した場合はエラーとなります。

2.4.8 負荷分散機能

XTC には，送信用ポート，受信用ポート，および通信回線に対する負荷を分散するための負荷分散機能があります。ここでは，XTC の負荷分散機能について説明します。

(1) 送信用ポートの共有と占有

送信用ポート数が少ない場合，送信用ポートをスレッド間で共有します。送信用ポートを共有した場合に，スレッド間で送信処理が競合したときは，送信用ポートの空き待ちを行います。そのため，競合するスレッド数が増えるほど，空き待ち時間が長くなります。

一方，送信用ポート数が多い場合は，送信用ポートをスレッドが占有します。この場合，スレッド間での待ち時間は発生しません。

送信用ポートの共有または占有は，次の表に示す規則で決定します。なお，送信 UDP グループごとに，このルールが適用されます。そのため，例えば，送信 UDP グループ A は共有，送信 UDP グループ B は占有という構成も考えられます。

表 2-13 送信用ポートの共有または占有を決定するためのルール

項番	条件	結果
1	送信用ポート数 ≧ メインスレッドを除いたスレッド数	すべてのスレッドは占有
2	送信用ポート数 ＞ 処理スレッド数 + 送信スレッド数	処理スレッド，および送信スレッドは占有，それ以外のスレッドは共有
3	項番 1 および項番 2 以外の場合	すべてのスレッドは共有

(2) 受信用ポート（あて先ポート）の分散

受信 UDP グループに複数のポート番号を指定した場合，複数の UDP 受信スレッドが同時に受信処理を行うことで，負荷分散を行います。

あて先 UDP グループにポート番号が複数指定されている場合，送信元プロセスは，あて先ポート番号をランダムに選択します。

2.5 トランザクション制御

ここでは，TP1 キャッシュ機能使用時の CL サーバでのトランザクション制御について説明します。

2.5.1 TP1 キャッシュ機能使用時の CL サーバでのトランザクション

TP1 キャッシュ機能使用時の CL サーバでのトランザクションは，ルートブランチだけで構成されます。

TP1 キャッシュ機能使用時にトランザクション連携できるリソースマネジャは，CL サーバの XDB だけです。TP1 キャッシュ機能使用時の XDB の呼び出しタイミングを次の表に示します。

表 2-14 TP1 キャッシュ機能使用時の XDB の呼び出しタイミング

項番	タイミング	実施内容
1	トランザクション開始時	1. XTC から XDB に対して，トランザクション ID を通知（中央処理通番） 2. XDB から XTC に対して，コネクトコンテキストを通知
2	トランザクションコミット時	1. XTC から XDB に対して，トランザクション ID を通知（中央処理通番） 2. XDB が更新ログ作成 3. DB 更新
3	更新ログサイズ取得時	XTC から XDB に対し，更新ログサイズを問い合わせ
4	トランザクションロールバック時	1. XTC からトランザクション ID を通知（中央処理通番） 2. 更新データを更新する前に戻す
5	待機系での更新ログ反映時	XDB が更新ログ反映

また，XDB とのインタフェースには XA インタフェースを使用しないで，独自のインタフェースによって連携します。HA サーバでは，データベースとして HiRDB を使用できますが，TP1 キャッシュ機能とのトランザクション連携はできません。

TP1 キャッシュ機能を使用する場合と TP1 キャッシュ機能を使用しない場合でのトランザクション制御の違いを，次の表に示します。

表 2-15 TP1 キャッシュ機能を使用する場合と使用しない場合でのトランザクション制御の違い

項番	項目	TP1 キャッシュ機能を使用しない場合	TP1 キャッシュ機能を使用する場合	
			HA サーバ	CL サーバ
1	トランザクション構成	複数のトランザクションブランチで構成	複数のトランザクションブランチで構成	ルートトランザクションブランチだけで構成
2	トランザクショナル RPC	サポート	サポート	未サポート
3	連携リソースマネジャ	HiRDB，Oracle（XA インタフェースに準拠したリソースマネジャ）	HiRDB，Oracle（XA インタフェースに準拠したリソースマネジャ）	XDB（独自のインタフェース）
4	コミット	二相コミット	二相コミット	一相コミット

2.5.2 関連する API

TP1 キャッシュ機能使用時に使用できる同期点取得に関連する API を，次の表に示します。

表 2-16 TP1 キャッシュ機能使用時の同期点取得に関連する API

項番	API 項目	
	C 言語関数名	thkind の指定値
1	ee_trn_chained_rollback	EETRN_PRCS
		EETRN_KILL
		EETRN_ABRT
		EETRN_KEEP
2	ee_trn_rollback_mark	なし

2.5.3 トランザクショナル RPC の抑止（CL サーバ）

TP1 キャッシュ機能使用時，CL サーバでは，トランザクショナル RPC を抑止する必要があります。そのため，CL サーバでは，トランザクション連携定義の trn_transactional_rpcless_use オペランドに Y を指定し，トランザクショナル RPC の抑止を有効にしてください。

2.5.4 RPC 応答の送信タイミング変更機能

TP1 キャッシュで CL 連携による系切り替え機能使用時，コミットと同期して行う RPC 応答メッセージの送信処理のタイミングを CL 連携の転送処理の結果と同期するための機能です。

TP1 キャッシュの CL サーバで，RPC 応答と転送処理のタイミングを変更するものです。

この機能の使用有無は，トランザクション関連定義の trn_cl_rpc_reply_timing オペランドで指定できます。

この機能を使用するときに変更できる処理を次に示します。

送信タイミングの変更

永続化しない（RPC 関連定義の rpc_recv_permanence オペランドに Y 指定なし）同期応答型 RPC/ 非同期応答型 RPC の応答送信を，永続化する（RPC 関連定義の rpc_recv_permanence オペランドに Y 指定）ときの応答送信と同様に，転送処理によってリソースの永続化が完了したあとに行います。

実行系孤立状態を検知したときの動作の変更

転送失敗によって実行系孤立状態を検知した場合，RPC 応答メッセージの送信はロールバックしたときと同様の動作をするように変更します。ただし，ERRTRNR は起動しません。

次の条件をすべて満たした場合，トランザクション結果に加え，転送処理の結果を RPC 応答メッセージに反映できます。

- rpc_reply_tp1mode オペランドに N を指定
- rpc_reply_errtrnr オペランドを省略，または N を指定

RPC 応答メッセージの内容を，次の表に示します。

表 2-17 RPC 応答メッセージの内容

トランザクション結果	転送処理の結果	RPC 応答メッセージ
コミット	正常	RPC 正常応答
	転送失敗 (孤立検知) ※	RPC エラー応答（EERPCER_TRNCHK_EXTEND）
ロールバック	―	RPC エラー応答（EERPCER_TRNCHK_EXTEND）

注※

系切り替えが発生すると，転送結果が反映されている可能性があります。

(1) RPC 要求受信時の処理シーケンス

(a) RPC 永続化指定なし（RPC 関連定義 rpc_recv_permanence に Y 指定なし）の場合

永続化しない RPC 要求メッセージを受信した場合について，次のケースのそれぞれの処理シーケンスを示します。

- RPC 応答の送信タイミング変更機能を使用するケース（trn_cl_rpc_reply_timing=Y）
- 機能未使用のケース

図 2-30 RPC 応答の送信タイミング変更機能使用時（trn_cl_rpc_reply_timing=Y）の処理シーケンス

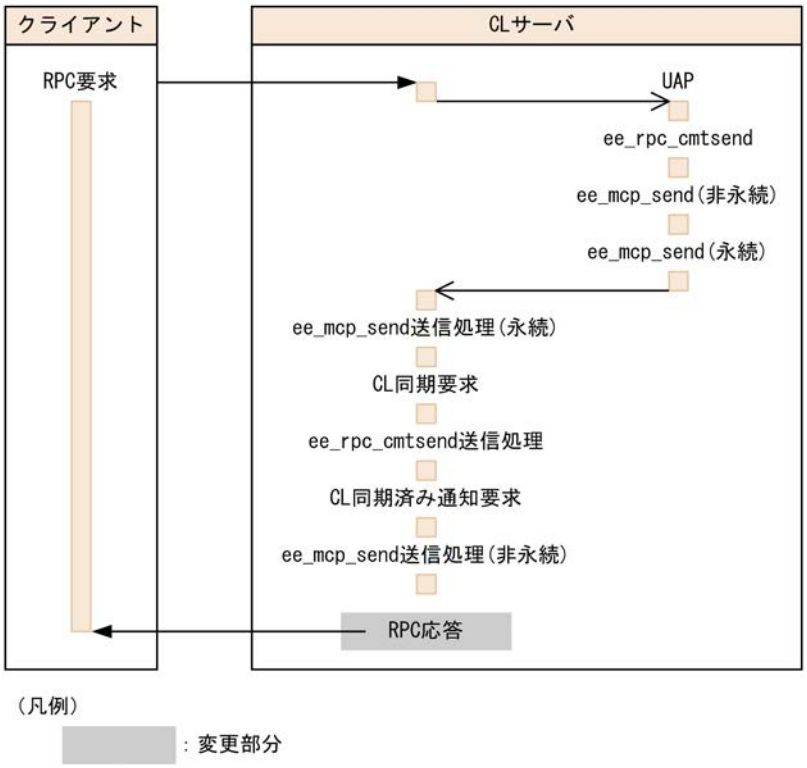
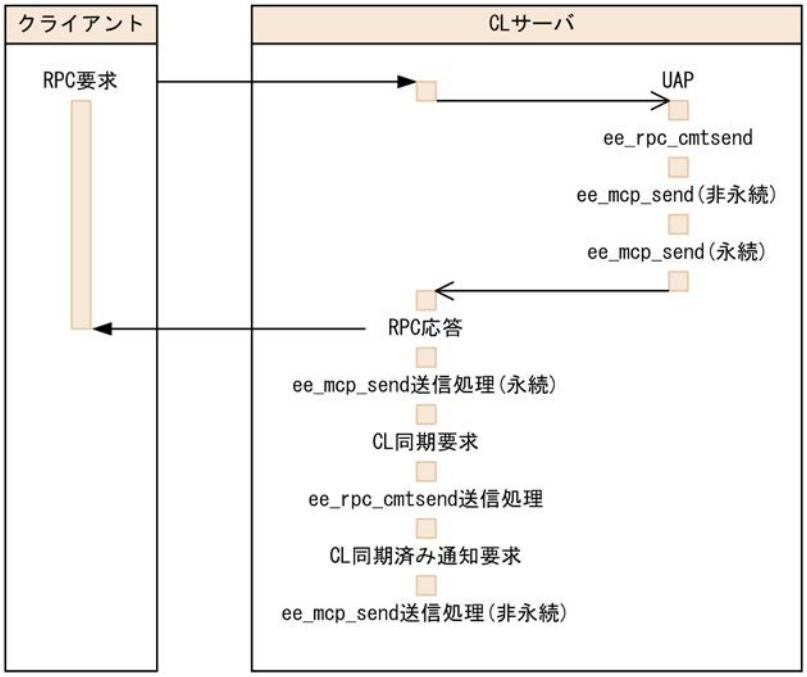


図 2-31　この機能を使用しないときの処理シーケンス



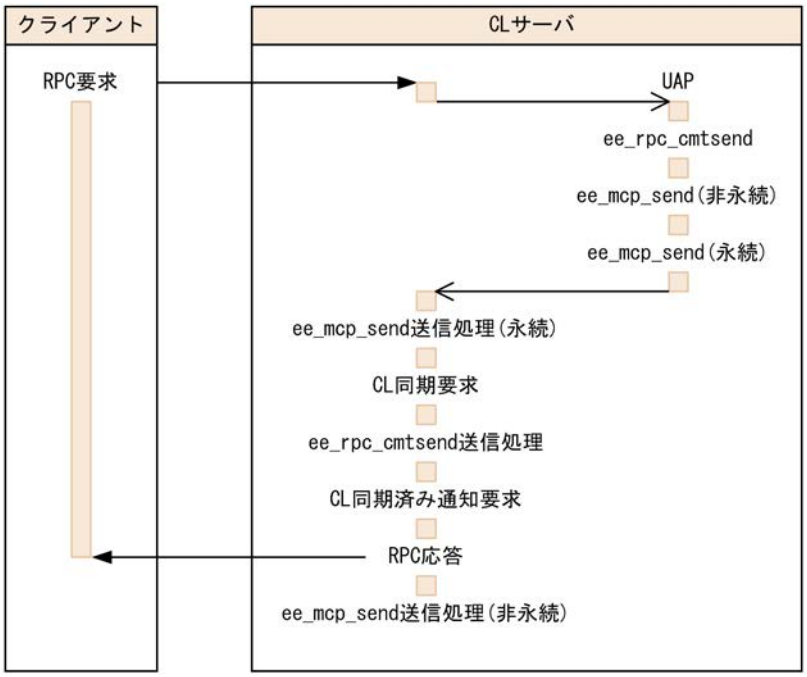
(b) RPC 永続化指定あり（RPC 関連定義 rpc_recv_permanence に Y 指定）の場合

永続化する RPC 要求メッセージを受信した場合について，次のケースの処理シーケンスを示します。

- RPC 応答の送信タイミング変更機能を使用するケース（trn_cl_rpc_reply_timing=Y）
- 機能未使用のケース

なお，RPC 永続指定ありの場合は，各ケースで処理シーケンスに変更はありません。

図 2-32 RPC 応答の送信タイミング変更機能使用時（trn_cl_rpc_reply_timing=Y）の処理シーケンス，およびこの機能を使用しないときの処理シーケンス



(2) 実行系孤立状態の実行系／待機系のメモリ DB

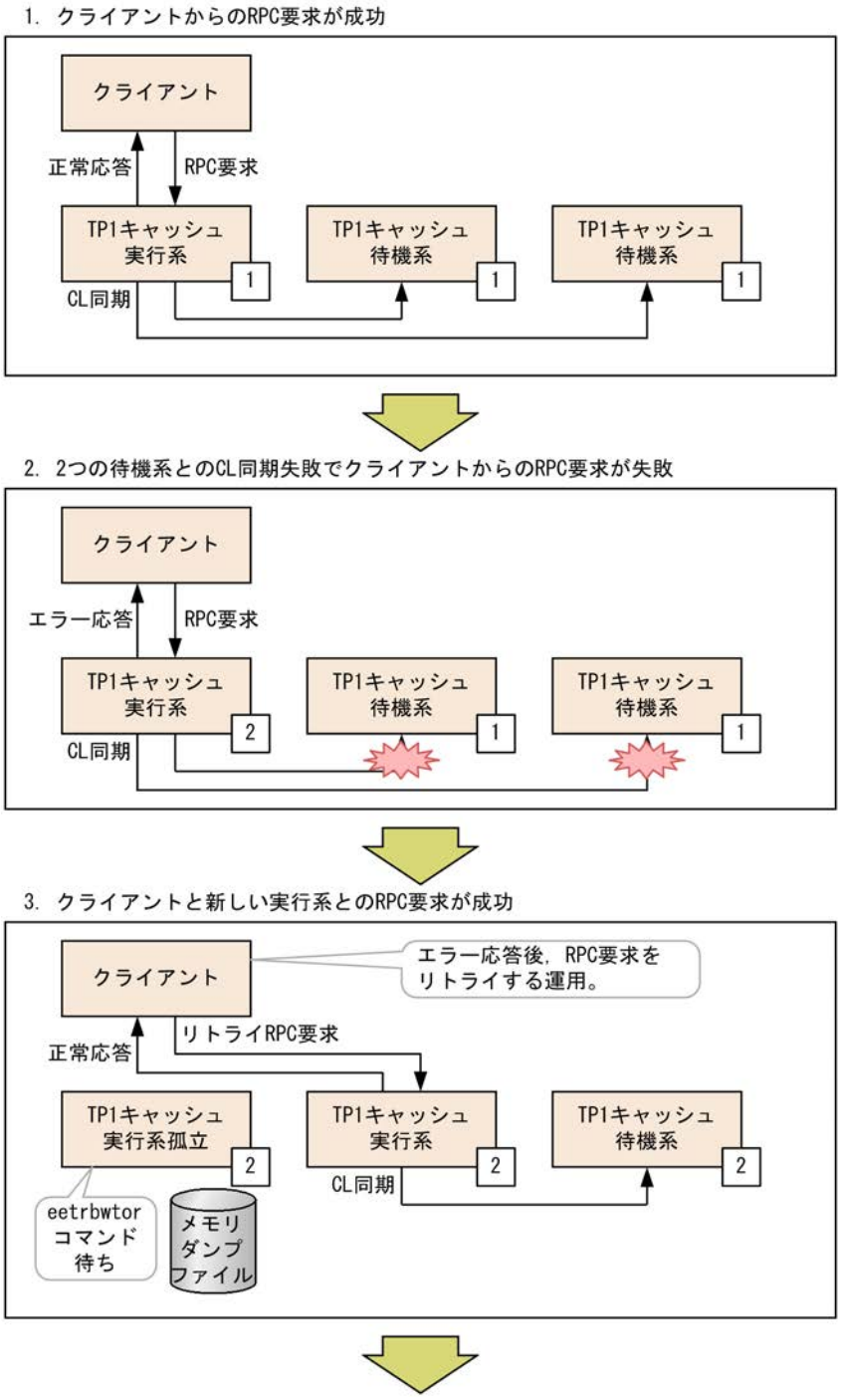
この機能使用時，転送処理失敗により実行系孤立状態を検知した場合には，クライアントからの RPC 要求はエラー応答となり，待機系に転送処理が反映されているか不明となります。このため，TP1 キャッシュのメモリ DB は実行系と待機系で不一致となることがあります。

RPC 要求をリトライする場合，2 重実行されても問題ないような UAP を作成する必要があります。

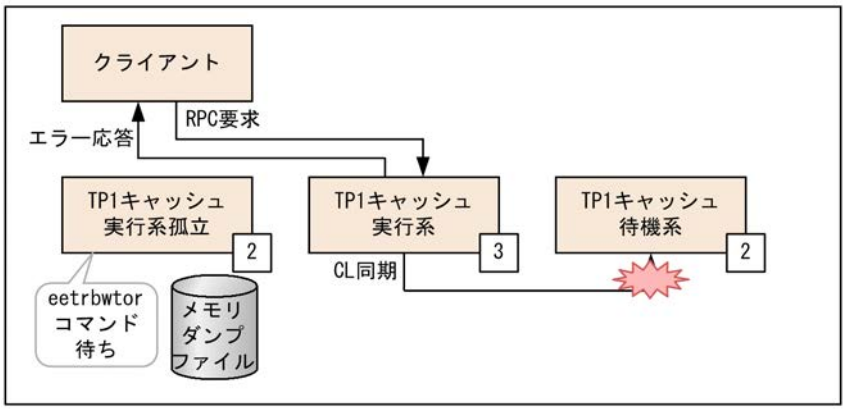
メモリ DB の不一致となるケースの例を次に記します。

(a) すべての待機系が転送失敗となった場合の流れ

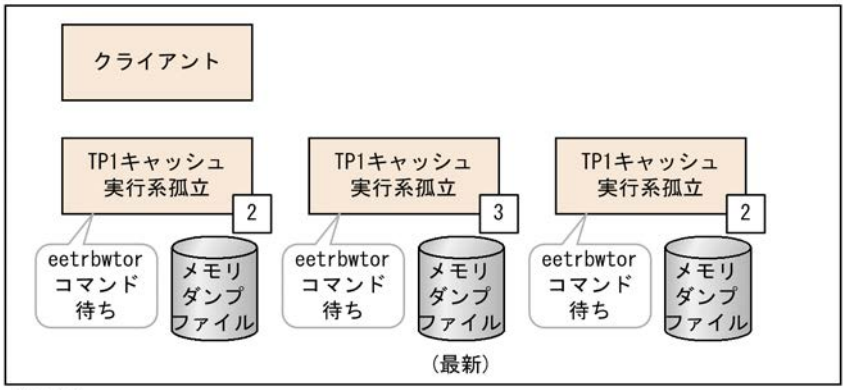
図 2-33 すべての待機系が転送失敗となった場合の流れ



4. 待機系とのCL同期失敗でクライアントからのRPC要求が失敗



5. すべてのTP1キャッシュでメモリダンプファイルを出力し，eetrbwtor待ちとなる

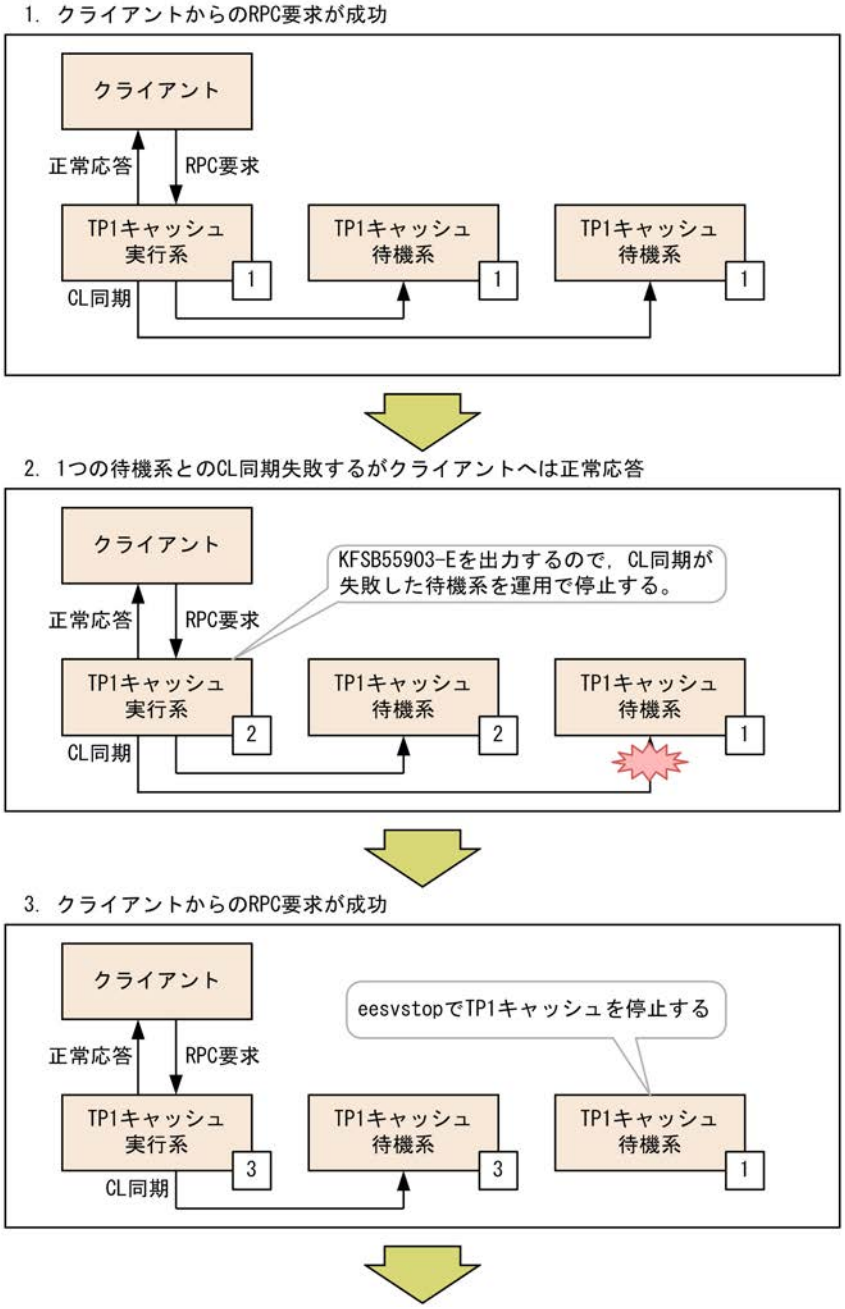


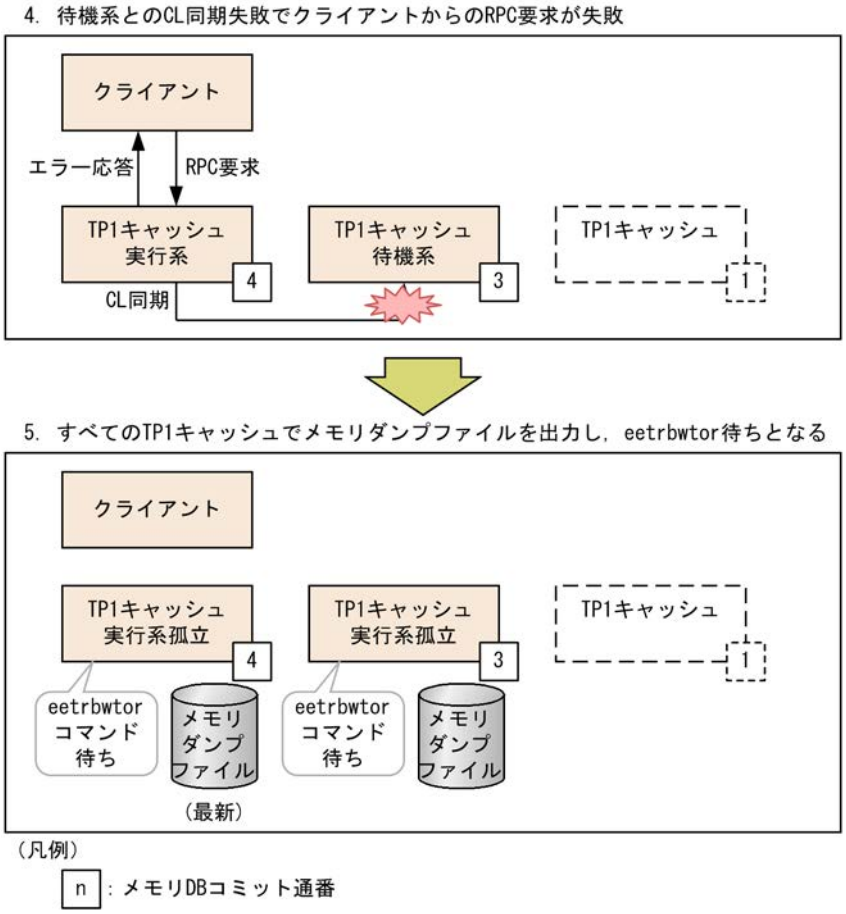
（凡例）

n ：メモリDBコミット通番

(b) 1 台ずつ転送失敗となった場合の流れ

図 2-34 1 台ずつ転送失敗となった場合の流れ





（3） 実行系孤立状態になった場合の対処

実行系孤立時の eetrbwtor **コマンド待ち**

孤立モード B を選択してください。孤立モード A を選択した場合，メモリダンプファイル出力後に受け付け済みサービスが動作するため，実行系孤立後のメモリ DB 更新の内容はメモリダンプファイルへ出力されません。また，クライアントへの RPC 応答は失敗します。

2.6 処理キューの制御

処理キューの制御は TP1/EE によって行われますが，TP1 キャッシュ機能使用時は，使用するキューや実現できる機能に一部異なる点があります。ここでは，TP1 キャッシュ機能使用時の処理キューの制御方式について説明します。

2.6.1 TP1 キャッシュ機能使用時の処理キューの制御方式

TP1/EE では，処理キューの制御に処理キュー制御用バッファ（PCE）および入力バッファ（IBF）を使用します。TP1 キャッシュ機能使用時はそれに加え，入力キュー（ITQ）を使用して処理キューを制御します。

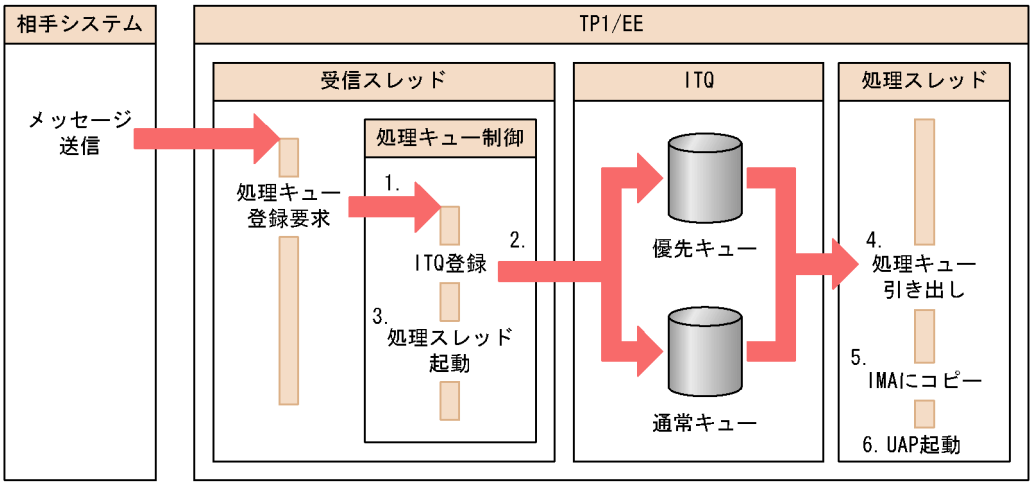
TP1 キャッシュ機能を使用する場合と TP1 キャッシュ機能を使用しない場合での処理キューの制御方式の違いを，次の表に示します。

表 2-18 TP1 キャッシュ機能を使用する場合と使用しない場合での処理キューの制御方式の違い

項目	制御方式の違い	
	TP1 キャッシュ機能を使用しない場合	TP1 キャッシュ機能を使用する場合
メッセージの流れ	メッセージ受信時に入力バッファに格納し，UAP 起動時に入力メッセージ引き渡しエリア（IMA）にコピーしてから，入力バッファを解放します。	メッセージ受信時に入力バッファに格納し，処理キュー登録時にそのまま入力キューに登録します。
処理キュー制御用バッファおよび入力バッファの解放	トランザクション起動時にすべて解放します。	トランザクション終了まで保持します。

TP1 キャッシュ機能使用時の処理キューの制御方式を，メッセージの受信から UAP 起動までの流れを例にして，次の図に示します。

図 2-35　処理キューの制御方式



説明

1. 相手システム（送信元）から受信した入力メッセージを，処理キューへ登録要求します。
2. 処理キューに登録された入力メッセージおよび処理キュー制御用バッファ情報を，入力キューに書き込みます。優先キューと通常キューのどちらに書き込むかは送信元の指示に従います。優先キューに書き込まれたメッセージは，優先して処理されます。通常キューに書き込まれたメッセージは，優先キューのメッセージをすべて処理したあとで，処理されます。
3. 入力メッセージの読み出しを処理スレッドで処理させるために，処理スレッドを起動します。
4. 起動した処理スレッドは，該当するサービスから処理キューの引き出し要求をします。
5. 入力メッセージ引き渡しエリアにメッセージをコピーします。
6. UAP を起動して，メッセージを引き渡します。

2.6.2　メッセージの同時引き出しの有無

メッセージは，該当するサービスに接続されている入力キューの先頭から毎回読み出されることでスケジュールされます。

このとき，入力キューのメッセージを読み出すトランザクションを一つずつ起動する（メッセージの同時引き出しなし）か，複数のトランザクションを起動する（メッセージの同時引き出しあり）かを service_attr 定義コマンドの -e オプションで指定できます。

なお，メッセージの同時引き出しなしを指定した場合に，起動中のトランザクションがあったときは，そのトランザクションが終了してから次のトランザクションを起動します。

メッセージの同時引き出しの有無によって，次に示す差異があります。

表 2-19　メッセージの同時引き出しの有無よる差異

項番	項目	メッセージの同時引き出し	
		なし	あり
1	メッセージの同時引き出し限界数	1	1 ～ 255 ※1
2	メッセージの同時引き出し限界数の変更	変更できない	eelspce コマンドで変更できる
3	複数メッセージの読み出し	使用できる ※2	使用できない
4	読み出しメッセージの差し戻し	使用できる ※2	使用できない
5	ロールバック時の読み出しメッセージの扱い	リトライする ※3	破棄する

注※ 1

同時引き出し限界数は，TP1/EE サービス定義のユーザサービス関連定義の service オペランドで指定します。

注※ 2

連鎖モード連携機能による再起動トランザクションでは使用できません。連鎖モード連携機能については，「2.6.8　連鎖モード連携機能」を参照してください。

注※ 3

次に示すメッセージ，およびトランザクションの場合は破棄します。

・応答型 RPC
・トランザクショナル RPC
・RAP メッセージ
・エラートランザクション
・MCP 後処理トランザクション（RL）
・イベント通知トランザクション（MV）

2.6.3　後続メッセージの読み出し

TP1 キャッシュ機能使用時は，トランザクション実行中（UAP 処理中）に，該当するサービスに滞留している後続メッセージを読み出せます。後続メッセージを読み出すには，ee_scd_msg_receive 関数を発行します。なお，この関数を発行できるのは，ユーザサービス関連定義の service オペランドの同時処理限界数に 1 が指定されたサービスだけです。

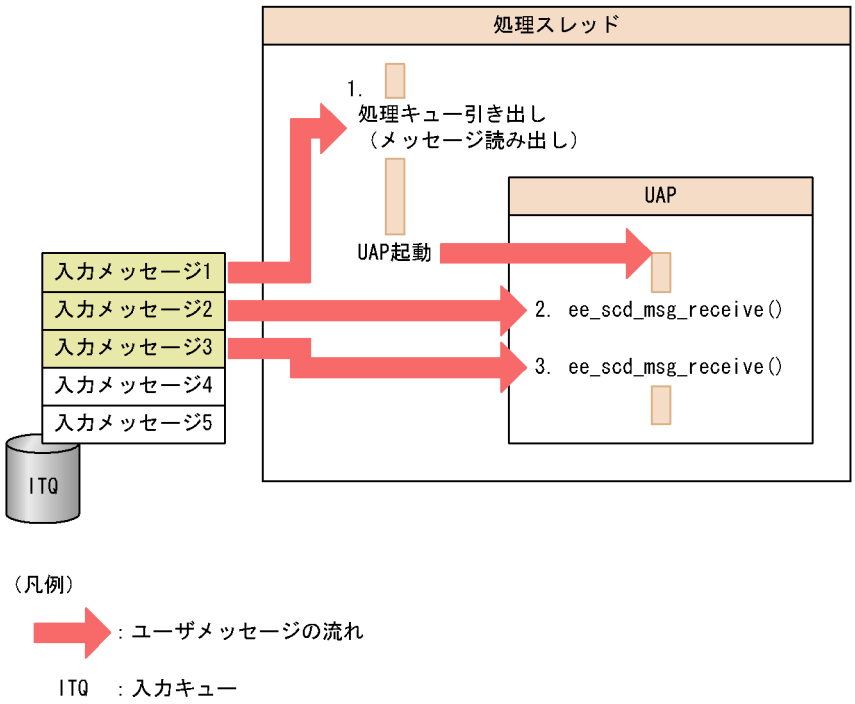
同時処理限界数に 1 を指定した場合は，一つのトランザクションで複数のメッセージを読み出せます。読み出せるメッセージは，トランザクション起動時に入力キューに滞留

しているメッセージです。トランザクション実行中に新たに書き込まれたメッセージは，次のトランザクションから読み出せます。

一つのトランザクションで読み出せるのは，優先キューと通常キューのどちらか一方に滞留しているメッセージです。サービストランザクション起動時に読み出したキューと，同じ（優先または通常）キューに滞留しているメッセージを読み出せます。

後続メッセージの読み出し例を次の図に示します。

図 2-36　後続メッセージの読み出し例



説明

1. 処理キュー引き出し後のメッセージ読み出しで，入力メッセージ 1 を読み出し，UAP を起動します。
2. ee_scd_msg_receive 関数によって，入力メッセージ 2 を読み出します。
3. ee_scd_msg_receive 関数によって，入力メッセージ 3 を読み出します。

(1) 読み出しできる後続メッセージの条件

ee_scd_msg_receive 関数で，読み出すメッセージの種別を API 引数で指定できます。ee_scd_msg_receive 発行時に，後続メッセージの読み出し方法を「メッセージ種別を指定しない方式」および「メッセージ種別を指定する方式」から選択できます。

(a) メッセージ種別を指定しない方式

ee_scd_msg_receive 関数で読み出せる後続メッセージ種別は，トランザクション起動時と同じメッセージ種別で，RPC メッセージ，MCP メッセージ，タイマメッセージ，および MCH メッセージです。

ただし，RPC メッセージの場合，応答型 RPC，トランザクショナル RPC，および RAP は読み出せません。また，タイマメッセージの場合，永続か非永続かによって異なるメッセージ種別として扱います。

(b) メッセージ種別を指定する方式

メッセージ種別を指定する方式は，トランザクション起動時と異なるメッセージ種別の読み出しができます。次のメッセージ種別を指定できます。

- 端末入力（MCP）
- RPC（非応答型 RPC の場合だけ読み出しができます。トランザクショナル RPC は読み出しができません）
- タイマ（永続／非永続タイマは別メッセージ種別として扱います）

指定したメッセージ種別だけの読み出しができ，起動時のメッセージ種別も指定しない場合は，読み出しはできません。例えば，MCP メッセージでトラザクション起動した場合，MCP のメッセージ種別を指定しなければ MCP メッセージの読出しはできません。

起動時に読み出したメッセージ種別と後続メッセージで読み出しできるメッセージ種別の組み合わせを示します。

表 2-20　起動時のメッセージ種別と後続メッセージの組み合わせ

後続メッセージ	起動時のメッセージ					
	MCP	mach	RPC	タイマ（永続）	タイマ（非永続）	左記以外
MCP	○	×	○	○	○	×
mach	×	×	×	×	×	×
RPC	○	×	○	○	○	×
タイマ（永続）	○	×	○	○	○	×
タイマ（非永続）	○	×	○	○	○	×
上記以外	×	×	×	×	×	×

（凡例）
○：メッセージ種別指定時，読み出しができます。
×：読み出しができません。

(2) 読み出しできる後続メッセージの上限

ee_scd_msg_receive 関数は，トランザクション起動時に登録済みのメッセージ数以上の

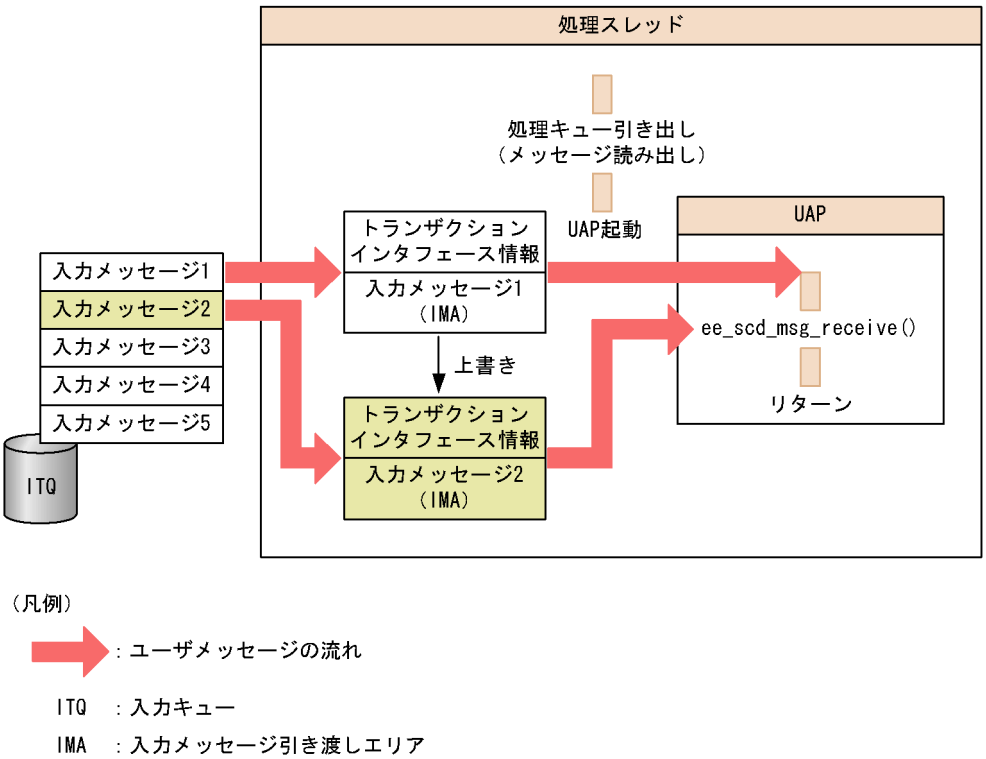
回数を発行できません。上限を超えて ee_scd_msg_receive 関数を発行した場合は，「読み出し上限を超えた」としてエラーになります。

(3) 後続メッセージ読み出し時の格納領域の上書き

後続メッセージは，トランザクション起動時と同様，入力メッセージ引き渡しエリア（IMA）にコピーされます。このため，ee_scd_msg_receive 関数を発行すると，前に読み込んだメッセージやトランザクション起動時のインタフェース情報は上書きされます。受信側の UAP から見ると，ee_scd_msg_receive 関数を発行することで新しいトランザクションを起動することになります。

後続メッセージを読み出すときの格納領域の上書きを次の図に示します。

図 2-37　後続メッセージを読み出すときの格納領域の上書き



ee_scd_msg_receive 関数を発行すると直前のメッセージを参照できなくなるため，必要に応じて事前に退避してください。

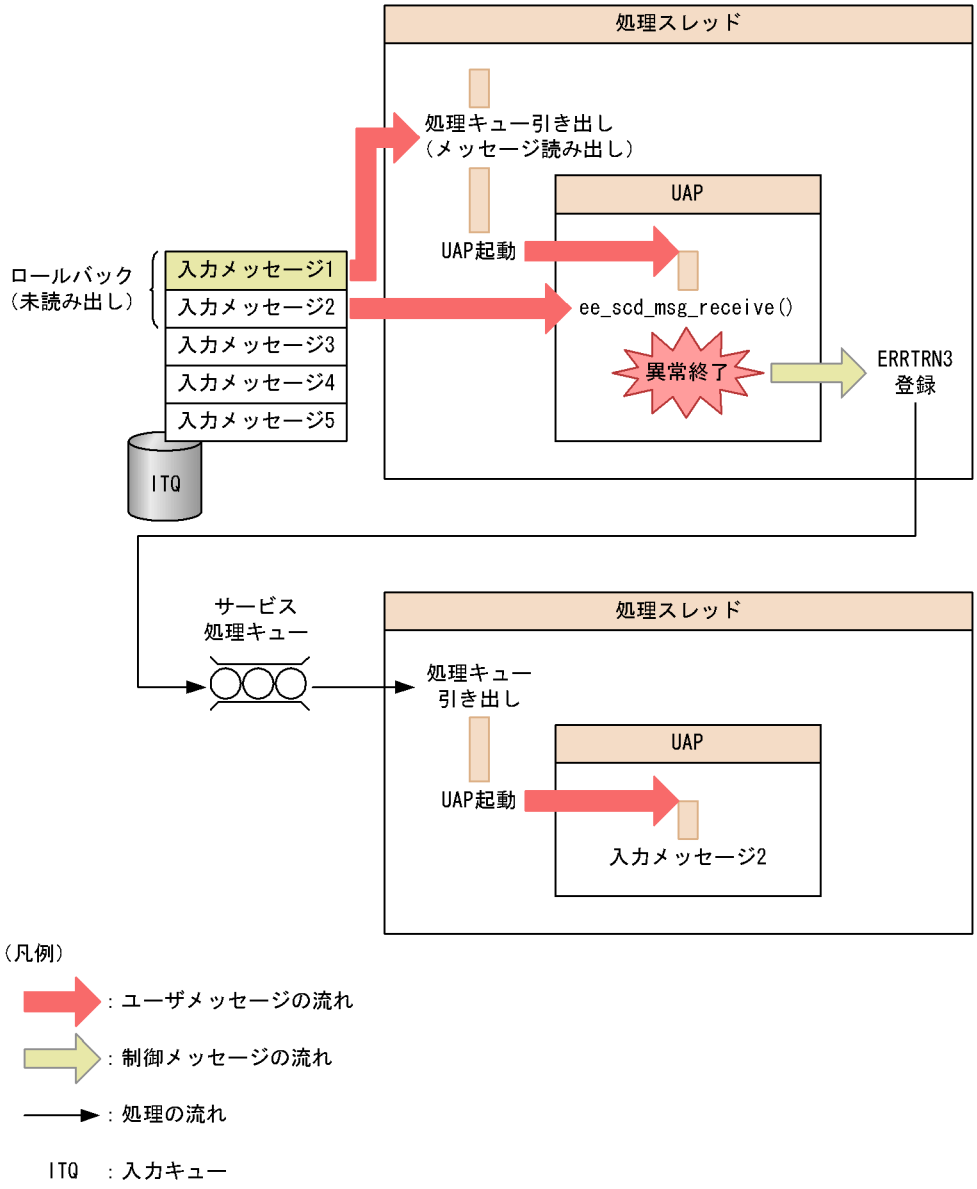
(4) コミット決着失敗通知（ERRTRNR）時および UAP 異常終了通知（ERRTRN3）時の動作

コミット決着に失敗するとエラートランザクションの ERRTRNR が起動します。また，UAP が異常終了すると，エラートランザクションの ERRTRN3 が起動します。

通常は，トランザクション起動時に UAP に引き渡した入力メッセージを，ERRTRNR または ERRTRN3 の入力メッセージとして引き渡します。しかし，ee_scd_msg_receive 関数を使用して後続メッセージを読み出している場合は，最後に発行した ee_scd_msg_receive 関数で読み出した入力メッセージを ERRTRNR または ERRTRN3 の入力メッセージとして引き渡します。

UAP 異常終了通知（ERRTRN3）時の動作を次の図に示します。

図 2-38　UAP 異常終了通知（ERRTRN3）時の動作



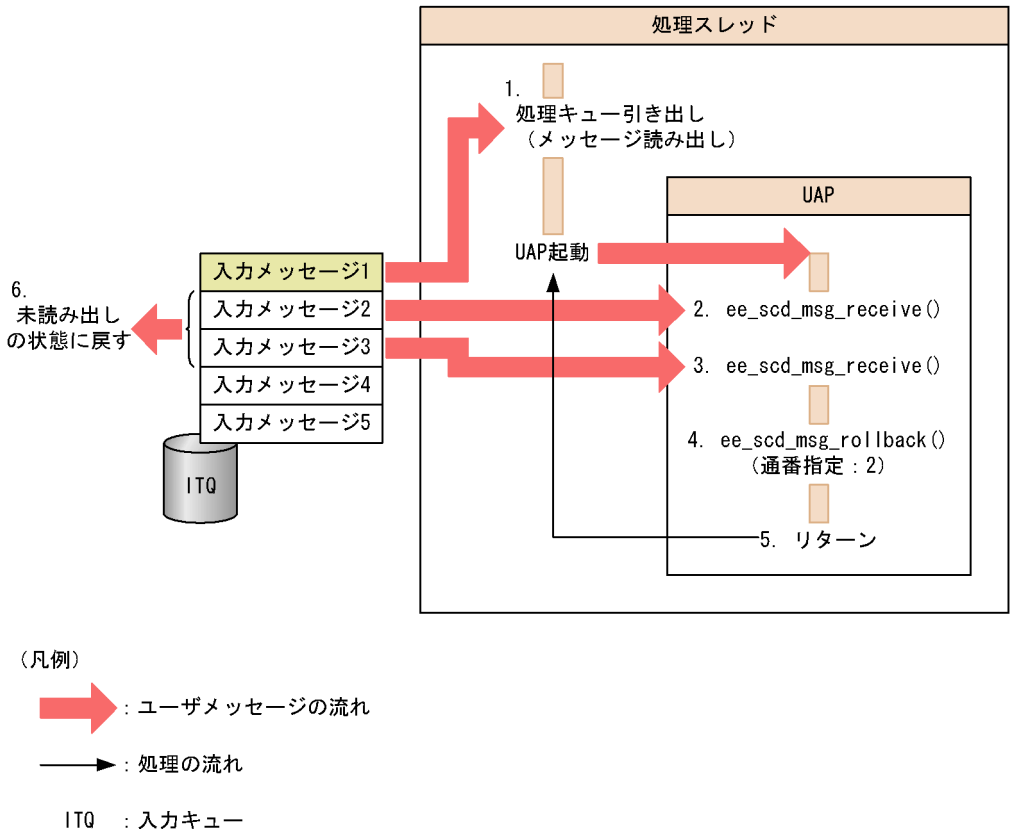
UAP が異常終了すると，メッセージをロールバックして，入力キューを未読み出しの状態にします。そのため，ERRTRN3 で引き渡された入力メッセージは，ERRTRN3 終了後に，通常のトランザクションまたは ERRTRN2 で再度 UAP に引き渡されます。

2.6.4　読み出しメッセージの差し戻し

メッセージの同時引き出しをなしに指定した場合，ee_scd_msg_receive 関数によって読み出したメッセージは，ee_scd_msg_rollback 関数によって差し戻すことができます。ee_scd_msg_rollback 関数を発行すると，トランザクションの終了状態に関係なく，読み出し済みのメッセージをロールバックして，メッセージを未読み出しの状態にします。

読み出しメッセージの差し戻し例を次の図に示します。

図 2-39　読み出しメッセージの差し戻し例



説明

1. 入力メッセージ 1 を読み出します。
2. ee_scd_msg_receive 関数によって，入力メッセージ 2 を読み出します。
3. ee_scd_msg_receive 関数によって，入力メッセージ 3 を読み出します。

4. ee_scd_msg_rollback 関数によって，入力メッセージ 2 以降を差し戻します。
5. UAP を終了します（リターンします）。
6. 入力メッセージ 2，3 が未読み出しの状態に戻ります。

トランザクション起動時に読み出したメッセージは，ee_scd_msg_rollback 関数では差し戻せません。トランザクション起動時に読み出したメッセージを差し戻したい場合は，トランザクションをロールバックさせる必要があります。

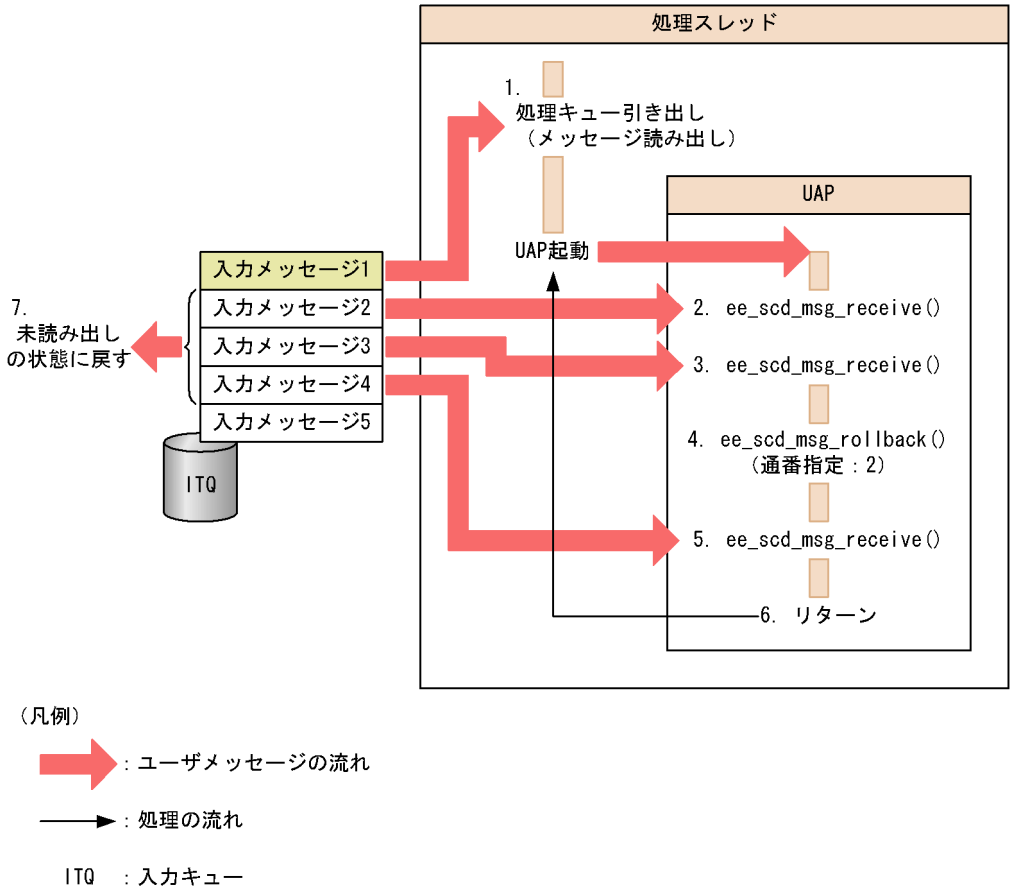
（1） 読み出しメッセージの差し戻しが有効となるタイミング

ee_scd_msg_rollback 関数によってメッセージが未読み出しの状態となるのは，ee_scd_msg_rollback 関数を発行して UAP がリターン（コミット）したタイミングです。

ee_scd_msg_receive 関数発行後，ee_scd_msg_rollback 関数を発行し，再度 ee_scd_msg_receive 関数を発行した場合は，次のメッセージを読み出します。

読み出しメッセージの差し戻しが有効となるタイミングを，次の図に示します。

図 2-40　読み出しメッセージの差し戻しが有効となるタイミング



説明

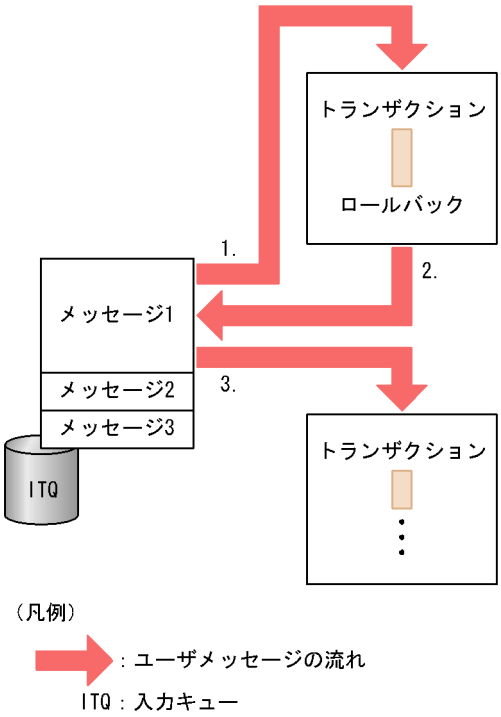
1. 入力メッセージ 1 を読み出します。
2. ee_scd_msg_receive 関数によって，入力メッセージ 2 を読み出します。
3. ee_scd_msg_receive 関数によって，入力メッセージ 3 を読み出します。
4. ee_scd_msg_rollback 関数によって，入力メッセージ 2 以降を差し戻します。
5. ee_scd_msg_receive 関数によって，入力メッセージ 4 を読み出します。
6. UAP 処理を終了します（リターンします）。
7. 入力メッセージ 2，3，4 が未読み出しの状態に戻ります（ee_scd_msg_rollback 関数の発行後に読み出した入力メッセージ 4 も，未読み出しの状態に戻ります）。

なお，ee_scd_msg_rollback 関数を複数回発行した場合は，最後に ee_scd_msg_rollback 関数で指定したメッセージ通番が有効となります。

2.6.5 ロールバック時の読み出しメッセージの扱い（ロールバックリトライ）

トランザクションがロールバックで決着して終了したときは，読み出したメッセージをすべてキューに戻し，再度同じメッセージを読み出すトランザクションを起動します。これをロールバックリトライといいます。ロールバックリトライは，メッセージの同時引き出しをなしに指定した場合だけできます。ロールバックリトライ時の処理の流れを次の図に示します。

図 2-41 ロールバックリトライ時の処理の流れ



説明

1. メッセージ 1 を読み出します。
2. メッセージ 1 のトランザクションがロールバックで決着して終了したため，メッセージ 1 は入力キューに差し戻されます。
3. メッセージ 1 が再度読み出され，トランザクションを開始します。トランザクションがコミットで決着するまで処理が繰り返されます。ただし，処理が繰り返されるのは，ロールバックリトライ回数の上限までです。

(1) ロールバック回数のカウント

ロールバック回数は，入力キューごとにカウントします。優先キューと通常キューはそれぞれ別にカウントされます。

ロールバック回数のカウント対象になるのは，UAP が明示的にロールバックを指示した場合と，同期点処理の通信障害などで TP1/EE が暗黙的にロールバックした場合です。

ロールバック回数は，次の場合にクリアされ，0 になります。

- トランザクションのコミット決着時
- メッセージ差し戻し指示のコミット決着時
- ee_scd_clear_rollback_cnt 関数の発行時

また，TP1/EE では，ロールバック回数を UAP に通知します。

(2) ロールバックリトライの監視

ロールバックリトライの監視は，TP1/EE で行うか，UAP で行うか選択できます。

TP1/EE でロールバックリトライを監視する場合は，ロールバックリトライ回数の上限をサービス属性定義の service_attr 定義コマンドの -f オプションで定義します。また，ロールバックリトライ回数が上限を超えたときに，該当するサービスを閉塞するか，プロセスダウンするかをユーザサービス関連定義の scd_rollback_retry_mode オペランドで定義します。

UAP で監視する場合は，ロールバック回数が一定の回数を超えたときに，次に示すような対応をします。

- 業務処理を行わない意味のないコミットとする
- サービス閉塞要求とする

2.6.6 プロセス終了時の未読み出しメッセージの扱い

サービスが閉塞している場合，サービス定義の forbid_draw_service オペランドに Y を指定していると，該当するサービスの入力キュー（ITQ）に未読み出しメッセージが滞留することがあります。

プロセス終了時，サービス閉塞によって入力キューに未読み出しメッセージが滞留していると，プロセスを終了できません。そのため，プロセス終了時に未読み出しメッセージがある場合は，1 分おきに警告のメッセージログを出力します。未読み出しメッセージの監視は，未読み出しメッセージがなくなるまで続きます。

プロセスを終了させたい場合は，次のどちらかの方法を用いてください。

- eelspcenum コマンドで未読み出しメッセージが滞留しているサービスを確認し，eepceskip コマンドで滞留メッセージを破棄する
- eeactsv コマンドでサービスの閉塞を解除して，メッセージを読み出せる状態にする

2.6.7 サービス先行解放

サービス先行解放とは，同期点でのメッセージ送信処理前にサービスを解放し，次のト

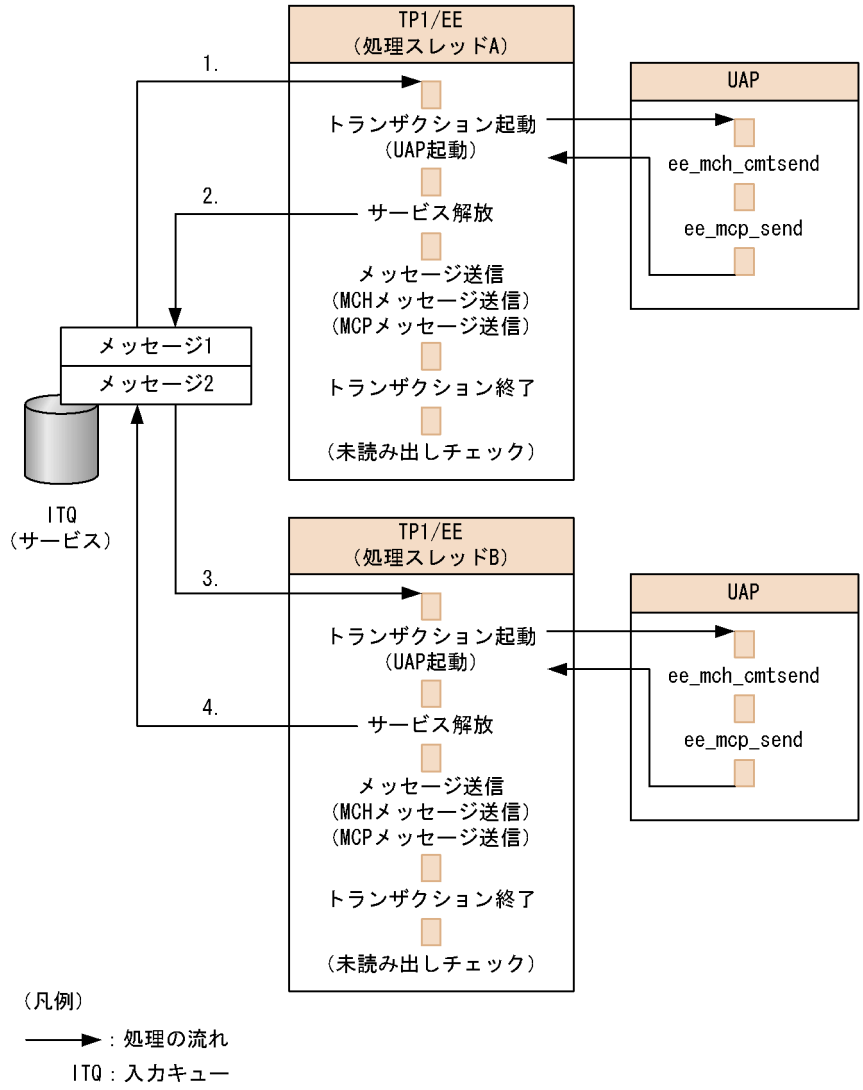
ランザクション（未読み出しメッセージ）の処理を先行して行えるようにすることです。UAP の処理（受信メッセージ処理）を早めるために，サービス先行解放を実行します。

サービス先行解放を実行するための条件を次に示します。

- TP1/EE プロセスが CL サーバである（クラスタ連携関連定義の cluster_mode オペランドに Y を指定）
- トランザクションがサービストランザクション（MN）またはタイマトランザクション（TM）である

サービス先行解放のタイミングを次の図に示します。

図 2-42　サービス先行解放のタイミング



説明

1. メッセージ 1 を読み出して，トランザクションを起動します。
2. 同期点でのメッセージの送信前に，サービスを解放します。
3. メッセージ 2 を読み出して，トランザクションを起動します。
4. 同期点でのメッセージの送信前に，サービスを解放します。

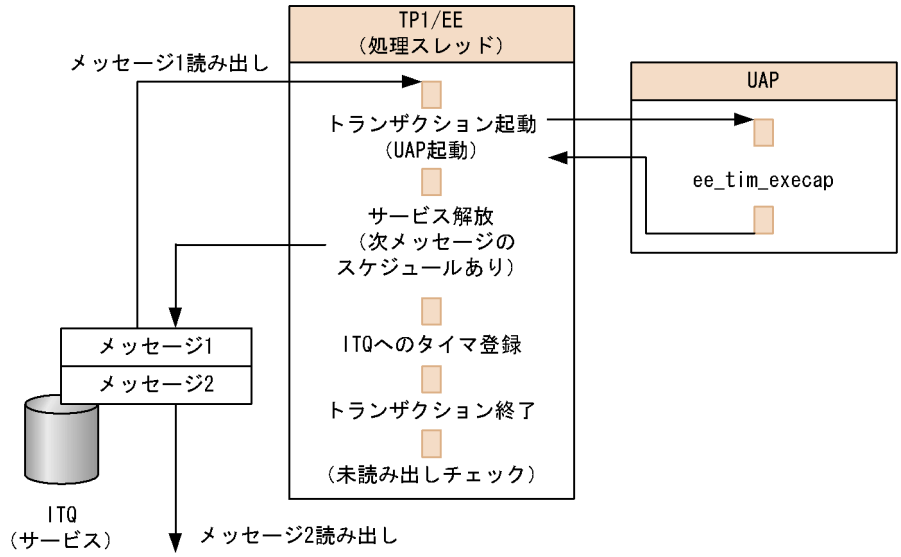
(1) タイマトランザクションのスケジュール

ee_tim_execap 関数でタイマトランザクションを即時起動する場合（action に EETIM_JUST を指定）などに，タイマ登録を行ったトランザクションが終了してからトランザクションを起動したいときは，flags オプションに EETIM_SCDL_HOLD を指定します。

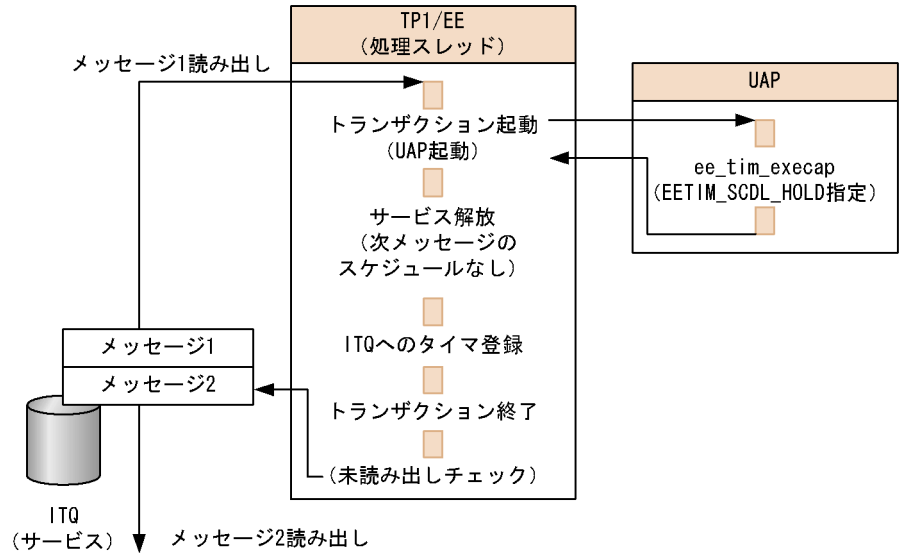
EETIM_SCDL_HOLD の指定によるメッセージ読み出しタイミングの違いを次の図に示します。

図 2-43　EETIM_SCDL_HOLD の指定によるメッセージ読み出しタイミングの違い

■次トランザクションのスケジュールを停止するオプション（EETIM_SCDL_HOLD）を指定しない場合



■次トランザクションのスケジュールを停止するオプション（EETIM_SCDL_HOLD）を指定した場合



（凡例）

：処理の流れ

ITQ：入力キュー

説明

- EETIM_SCDL_HOLD を指定しない場合は，サービス解放のタイミングでメッセージ 2 を読み出します。
- EETIM_SCDL_HOLD を指定した場合は，トランザクションが終了してからメッセージ 2 を読み出します。

EETIM_SCDL_HOLD を指定したタイマをキャンセルすると，EETIM_SCDL_HOLD を指定しなかったものとして，サービス解放時に次のメッセージをスケジュールします。

また，EETIM_SCDL_HOLD を指定した複数のタイマを登録した場合に，EETIM_SCDL_HOLD を指定したタイマをキャンセルすると，EETIM_SCDL_HOLD を指定したタイマが一つでもあれば，トランザクション終了時に次のメッセージをスケジュールします。

なお，EETIM_SCDL_HOLD を指定したタイマを登録したトランザクション（サービス）の同時実行数が複数かどうかに関係なく処理を行います。

2.6.8　連鎖モード連携機能

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，ロールバック

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，ロールバックがあります。

連鎖モード連携機能とは，読み出したメッセージを連鎖モードのコミット，ロールバックと連携して決着するか，または連携しないで UAP 終了で決着するかを指定する機能です。サービス属性定義の service_attr オペランドの -g オプションによって指定できます。

(1) 連鎖モード連携する場合

連鎖モード連携機能を使用すると，連鎖モードのコミット，ロールバックと連携して読み出したメッセージを決着させます。

■連鎖モードのコミットで決着した場合

トランザクションが連鎖モードのコミットによって決着すると，読み出したメッセージは読み出し済みになります。

■連鎖モードのロールバックで決着した場合

トランザクションが連鎖モードのロールバックによって決着すると，メッセージはリトライまたは破棄となります。

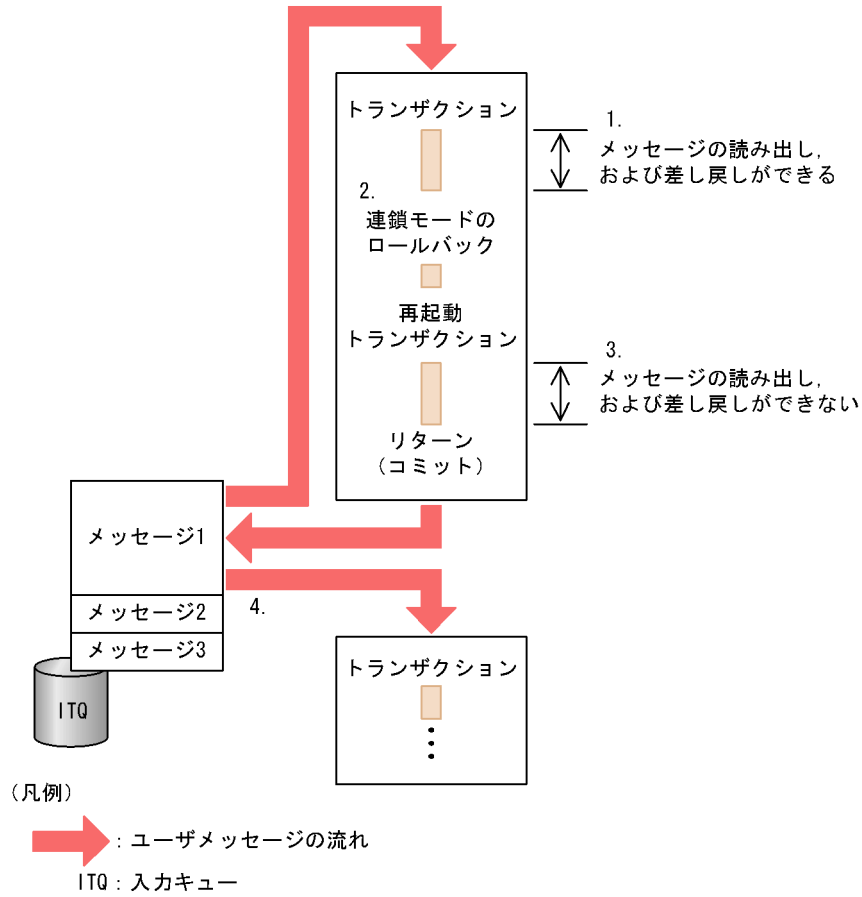
トランザクションがロールバックしたときの再起動トランザクションは，読み出したメッセージとは関係のないトランザクションとなります。再起動トランザクションでは，メッセージの読み出し，およびメッセージの差し戻しはできません。また，再起動トランザクションで連鎖モードのコミットまたはロールバックによってトランザクションが決着しても，メッセージには影響しません。

■連鎖モードのコミット，ロールバックを要求しなかった場合

連鎖モードのコミット，ロールバックを要求しなかった場合は，UAP がリターンした時点のトランザクションの決着によって，メッセージは読み出し済み，リトライ，または破棄となります。

連鎖モード連携する場合の処理の流れを次の図に示します。

図 2-44 連鎖モード連携する場合の処理の流れ



説明

1. メッセージ 1 のトランザクションが開始されたあとも，トランザクションが決着するまでの間，メッセージの読み出しおよび差し戻しができます。
2. メッセージ 1 のトランザクションが，連鎖モードのロールバックによって決着し，リトライが決定したため，残りの処理をメッセージ 1 とは関係のない再起動トランザクションとします。
3. 連鎖モードのロールバックによる再起動トランザクションでは，メッセージの読み出しおよび差し戻しはできません。また，再起動トランザクションが決着した時点では，メッセージ 1 は読み出し済みになりません。
4. 2. でリトライが決定したため，メッセージ 1 が再度読み出されます。トランザクションがコミットで決着すると，メッセージ 1 は読み出し済みになります。

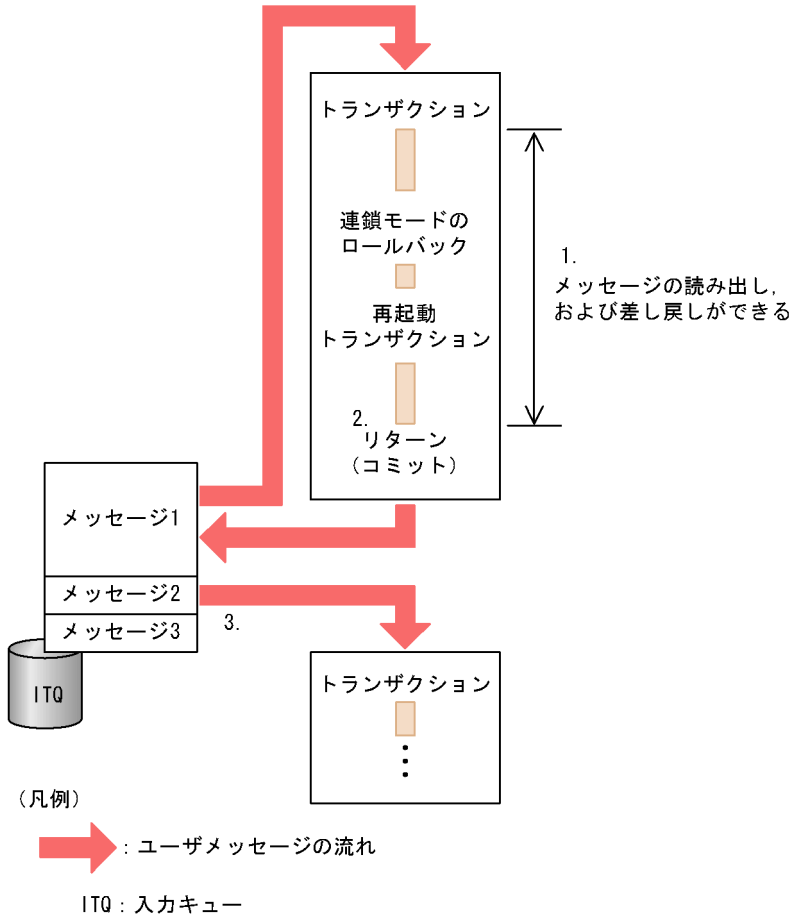
(2) 連鎖モード連携しない場合

連鎖モード連携しない場合，つまり，連鎖モードのコミットまたはロールバックによるトランザクションの決着でメッセージを決着しない場合は，UAP がリターンした時点の

トランザクションの決着で，メッセージを決着します（読み出し済み，リトライ，または破棄とします）。

連鎖モード連携しない場合は，再起動トランザクションでメッセージの読み出し，およびメッセージの差し戻しができます。連鎖モード連携しない場合の処理の流れを次の図に示します。

図 2-45　連鎖モード連携をしない場合の処理の流れ



説明

1. 連鎖モード連携しないため，トランザクションがロールバックで決着したあとの再起動トランザクションでもメッセージの読み出し要求，および差し戻し要求ができます。
2. メッセージ 1 は，再起動トランザクションの決着によって読み出し済みになります。
3. メッセージ 1 が読み出し済みになったため，メッセージ 2 が読み出され，トランザクションが開始されます。

2.7 ステータスファイルレス機能

TP1/EE で取得するステータスファイルには，TP1 キャッシュ機能使用時は必要のないシステム制御情報があります。そのため，**ステータスファイルレス機能**によって，ステータスファイルに取得するシステム制御情報を限定します。

ステータスファイルレス機能には，全体レスと部分レスのオプションがあります。オプションによる違いを次の表に示します。

表 2-21 ステータスファイルレス機能での全体レスと部分レスの違い

項番	項目	全体レス	部分レス
1	指定するシステム構成	CL サーバ（必須）	HA サーバ（任意）
2	取得するシステム制御情報	すべてのシステム制御情報を取得しません。	トランザクションに関連するシステム制御情報を取得しません。
3	ステータスファイルの使用	使用しません。	使用します。

取得するシステム制御情報の詳細を次の表に示します。

表 2-22 取得するシステム制御情報の詳細

項番	システム制御情報	全体レス	部分レス
1	サービスの閉塞情報※	×	○
2	システムの終了状態	×	○
3	ラン ID，システムステータス	×	○
4	処理キューの最大同時処理限界数の変更情報※	×	○
5	システム定義情報	×	○
6	オンライン EE のプロセス定義情報	×	×
7	トランザクション情報	×	×
8	ロールバックアボート時の RPC 情報	×	×
9	RPC 応答メッセージ送信抑止機能使用時の RPC 情報	×	×
10	UAP 共用ライブラリ入れ替え情報	×	○
11	ERRTRNR による RPC 応答メッセージ送信情報	×	×

（凡例）

○：システム制御情報を取得します。

×：システム制御情報を取得しません。

注※

CL サーバの場合，実行系から待機系へシステム制御情報を転送することによって永続化されます。そのため，実行系がプロセスダウンしても，待機系でシステム制御情報を引き継ぐことができます。

ステータスファイルレス機能の使用は，ステータスファイル関連定義の sts_fileless_use オペランドで指定します。また，全体レスとするか部分レスとするかは，sts_fileless_level オペランドで指定します。

2.8 メモリの管理

TP1 キャッシュ機能使用時のメモリの管理方法は，TP1/EE だけを使用する場合と同様です。

ここでは，TP1 キャッシュ機能使用時にだけ使用するバッファおよびワーク領域，ならびにバッファ不足時の面数拡張について説明します。また，大量処理用メモリ管理機能についても説明します。

2.8.1 バッファの管理

ここでのバッファとは，特定の用途に使用することを目的とした，固定サイズの領域です。

TP1 キャッシュ機能使用時は，TP1/EE のバッファに加えて次の表に示すバッファを使用します。

表 2-23 TP1 キャッシュ機能使用時に使用するバッファ

項番	名称	用途	定義
1	UDP 用受信バッファ（UIBF）	UDP プロトコルを使用した通信でのメッセージ受信時に，受信データを格納するバッファです。	メモリ関連定義 • udp_recv_message_buf_cnt オペランド • udp_recv_message_buf_size オペランド
2	UDP 用送信バッファ（UOBF）	UDP プロトコルを使用した通信でのメッセージ送信時に，送信データを格納するバッファです。	メモリ関連定義 • udp_send_message_buf_cnt オペランド • udp_send_message_buf_size オペランド

各バッファは，プロセス初期化時に TP1/EE が一括して確保したメモリ領域内に作成されます。

なお，プロセス初期化時に作成されたバッファ面数が不足した場合，XTC 用のワーク領域として確保したメモリ領域を不足分のバッファに割り当てます。詳細については，「2.8.2 ワーク領域の管理」および「2.8.3 バッファ不足時の面数拡張」を参照してください。

2.8.2 ワーク領域の管理

ワーク領域とは，TP1/EE プロセスで使用するメモリ領域のうち，システムが一時的に使用するセグメントです。一定の領域をワーク領域として管理し，オンライン中に一時的なメモリが必要になった時点で，ワーク領域から必要なサイズのセグメントを切り出して使用します。なお，使用済みとなったセグメントは，次回利用時に備えてワーク領

域に返還します。

TP1 キャッシュ機能使用時は，TP1/EE が使用するワーク領域に加えて次の表に示すワーク領域を使用します。

表 2-24 TP1 キャッシュ機能使用時のワーク領域

項番	名称	用途	定義
1	XTC 用ワーク領域（XTCPOOL）	XTC が利用するワーク領域です。	メモリ関連定義 • memory_xtc_area_size オペランド • memory_xtc_limit_size オペランド
2	XDB 用ワーク領域（XDBPOOL）	XDB が利用するワーク領域です。	メモリ関連定義 • memory_xdb_area_size オペランド • memory_xdb_limit_size オペランド

ワーク領域は，プロセス初期化時に TP1/EE が一括して確保したメモリ領域内に作成されます。

なお，TP1/EE のワーク領域と異なり，TP1 キャッシュ機能使用時のワーク領域は，拡張できるサイズの上限を設定できます。この定義は，メモリ関連定義の memory_xdb_limit_size オペランドおよび memory_xtc_limit_size オペランドで行います。

オンライン中にワーク領域が不足した場合，設定された拡張サイズでワーク領域を拡張（malloc 関数の発行）します。一度拡張したワーク領域はオンライン終了まで解放（free 関数の発行）しません。

2.8.3 バッファ不足時の面数拡張

TP1 キャッシュ機能使用時にバッファ面数が不足すると，XTC 用ワーク領域（XTCPOOL）からバッファを割り当てて，バッファ面数を拡張します。

ただし，各バッファの面数は不足しないように，余裕を持って確保してください。

面数拡張の対象となる TP1/EE および XTC のバッファと，拡張する面数を次の表に示します。

表 2-25 面数拡張の対象となる TP1/EE および XTC のバッファと拡張する面数

項番	製品	バッファ種別	拡張する面数
1	TP1/EE	処理キュー制御用バッファ（PCE）	pce_no オペランドの指定値
2		タイマサービス制御用バッファ（ICB）	icb_no オペランドの指定値

項番	製品	バッファ種別	拡張する面数
3		タイマトランザクション用バッファ（TTBF）	time_message_no オペランドの指定値
4		受信バッファ（IBF）	recv_message_buf_cnt オペランドの指定値
5		送信バッファ（OBF）	send_message_buf_cnt オペランドの指定値
6		DB キュー用受信バッファ（QIBF）	dbq_recv_message_buf_cnt オペランドの指定値
7		DB キュー用送信バッファ（QOBF）	dbq_send_message_buf_cnt オペランドの指定値
8		DB キュー用最大バッファ（QWBF）	dbq_use_buf_cnt オペランドの指定値
9		システムバッファ	システム固定
10	XTC	UDP 用受信バッファ（UIBF）	udp_recv_message_buf_cnt オペランドの指定値
11		UDP 用送信バッファ（UOBF）	udp_send_message_buf_cnt オペランドの指定値

なお，面数の拡張で割り当てたバッファが使用済みになると，XTC 用ワーク領域へ返還します。割り当て先のバッファに，面数の拡張で割り当てたバッファがストックされることはありません。

2.8.4 大量処理用メモリ管理機能

一時的に大量のメモリを消費するトランザクションが発生すると，通常のオンライン中には使用しないメモリを TP1/EE が大量に確保し続けるおそれがあります。そのため，このようなトランザクションが発生する場合は，大量処理用メモリ管理機能を使用します。

大量処理用メモリ管理機能を使用すると，該当するトランザクションの処理中に要求されたメモリを，初期確保する領域からではなく別のメモリ領域から割り当てます。大量処理用メモリ管理機能で割り当てられた領域は，該当するトランザクションが終了したあとで，OS に解放されます。

（1） 使用する領域の種類と定義

大量処理用メモリ管理機能で使用する領域を次の表に示します。

表 2-26 大量処理用メモリ管理機能で使用する領域

項番	名称	TP1/EE が管理するメモリリソースとの対応	定義
1	大量処理用システム領域（MPSPOOL）	• すべてのバッファ • システム用ワーク領域 • XTC 用ワーク領域（XTCPOOL） • XDB 用ワーク領域（XDBPOOL）	メモリ関連定義の memory_mdpsys_area_size オペランド
2	大量処理用ユーザ領域（MPUPOOL）	ユーザ用ワーク領域	メモリ関連定義の memory_mdpusr_area_size オペランド

なお，大量処理用システム領域と大量処理用ユーザ領域は，どちらか一方だけを使用することもできます。

(2) 機能の使用開始と終了

UAP で ee_mem_mdpstart 関数を発行すると，大量処理用メモリ管理機能の使用を開始します。また，該当する UAP がリターンしてトランザクションが終了するときに，大量処理用メモリ管理機能も終了し，使用中のセグメントが存在しない領域を OS に解放します。なお，使用中のセグメントが存在する場合は，使用中のセグメントがすべて返還されてから，領域ごとに OS に解放します。

複数の UAP で並行して大量処理用メモリ管理機能を使用する場合，大量処理用システム領域および大量処理用ユーザ領域はそれぞれの UAP で共有します。この場合，領域の解放は，並行して動作する UAP のうち，最後の UAP がリターンしてトランザクションが終了してから行われます。

(3) 留意事項

大量処理用メモリ管理機能を使用する際の留意事項を次に示します。

■スレッドダウン時の動作

大量処理用メモリ管理機能を使用中のスレッドでスレッドダウンが発生した場合，該当するスレッドで使用された未返還の TASK 属性のセグメントを強制解放するときに，大量処理用メモリ管理機能を終了します。

■ XTC 用ワーク領域および XDB 用ワーク領域の動作

CL サーバ（クラスタ連携関連定義の cluster_mode オペランドに Y を指定）の場合，XTC ワーク領域および XDB ワーク領域の動作が次のとおりになります。

- 追加確保した領域については，使用中のセグメントが存在しなくなった時点で，オンライン中に該当する領域を OS に解放（free 関数を発行）します。
- 待機系では，メモリ関連定義の memory_xtc_limit_size オペランドおよび memory_xdb_limit_size オペランドの設定値が無効になります。なお，設定値を

超えるセグメント要求があった場合，設定値を無視して領域を追加確保（malloc 関数を発行）します。

なお，ee_mem_mdpstart 関数を発行したかどうかに関係なく，オンライン中は常にこの動作となります。

2.9　キューダンプ機能

ここでは，キューダンプ機能について説明します。

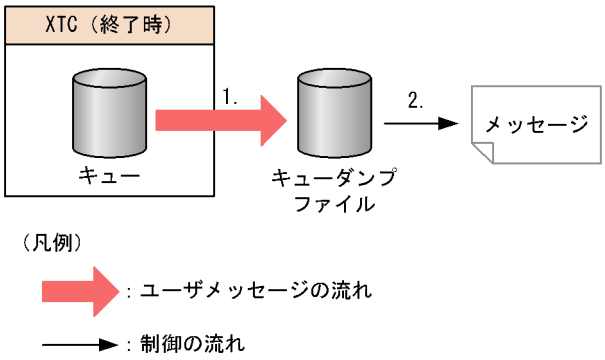
2.9.1　キューダンプ機能の概要

TP1 キャッシュ機能を使用すると，オンライン終了時にキューにメッセージが滞留する場合があります（このメッセージを**滞留メッセージ**といいます）。

この場合，**キューダンプ機能**によって，滞留メッセージのキューダンプファイルへの出力，編集が行えます。

キューダンプ機能の概要を次の図に示します。

図 2-46　キューダンプ機能の概要



説明

1. 滞留メッセージをキューダンプファイルに出力します。
   詳細については，「2.9.2　滞留メッセージのキューダンプファイルへの出力」を参照してください。
2. キューダンプファイルを編集して，参照します。
   詳細については，「2.9.3　キューダンプファイルに出力されたメッセージの参照」を参照してください。

2.9.2　滞留メッセージのキューダンプファイルへの出力

(1) キューダンプファイルへの出力タイミング

滞留メッセージは，次のタイミングでキューダンプファイルに出力されます。

- XTC 終了時
  正常終了，計画停止 A，または計画停止 B の場合に出力されます。また，CL サーバの待機系の場合は，待機系が単独で終了するときにだけ出力されます。
- 実行系孤立状態になったとき

参考

次の場合，滞留メッセージはキューダンプファイルに出力されません。
- 出力対象となるメッセージがない場合
- XTC の終了モードが強制停止またはプロセスダウンの場合
- 待機系が実行系と連動して終了する場合

(2) キューダンプファイルの作成と削除

キューダンプファイルは，TP1/EE によって TP1/EE プロセス（サービスグループ）ごとに専用のファイルが作成されます。キューダンプファイルの容量見積もりは，「3.2.1 キューダンプファイルの容量見積もり」を参照してください。

(a) XTC 終了時

XTC 終了時に作成されるキューダンプファイルは，次のディレクトリに格納されます。

```
$DCDIR/spool/dceeinf/quedump/
```

ファイル名は次のとおりです。

```
gg....ggquedump
```

変数の説明

gg....gg：サービスグループ名（最大 31 バイト）
quedump：キューダンプファイルを示す文字列（7 バイト固定）

なお，古いキューダンプファイルがある場合は新しいファイルに上書きされるため，必要に応じて古いファイルのバックアップを取得してください。

(b) 実行系孤立状態になったとき

実行系孤立状態になったときに作成されるキューダンプファイルは，次のディレクトリに格納されます。

```
$DCDIR/spool/dceeinf/hamdump/
```

ファイル名は次のとおりです。

```
gg....gg_nnnn_rrrrrrrr_cc....cc_xtc
```

変数の説明

gg....gg：サービスグループ名（最大 31 バイト）
nnnn：ノード識別子（4 バイト）
rrrrrrrr：ラン ID（8 バイト）※ 1
cc....cc：CL 通番（16 バイト）※ 2
xtc：キューダンプファイルを示す文字列（3 バイト固定）

注※ 1
ラン ID は，16 進数の通算秒です。

注※ 2
CL 通番は，実行系でトランザクションを起動するたびに加算される 16 進数の通番です。待機系から実行系に切り替わった場合も，プロセスダウン前の実行系の CL 通番を引き継ぐため，CL サーバでのトランザクションを一意に決定できます。

実行系孤立状態になったときに作成されるキューダンプファイルは，自動的には削除されません。必要に応じてバックアップを取得するか，または削除してください。

複数のキューダンプファイルから最新のファイルを探すには，サービスグループ名とノード識別子からファイルを絞り込み，その中で「ラン ID」＋「CL 通番」が最も大きいファイルを特定します。

(3) 出力対象となるメッセージ

キューダンプファイルに出力される対象となるメッセージを次の表に示します。

表 2-27 キューダンプファイルへの出力対象メッセージ

項番	システム構成（系）	キュー種別	メッセージ種別	出力対象
1	CL サーバ（実行系）	ITQ	RPC メッセージ（非永続）	×
2			RPC メッセージ（永続）	○
3			MCP メッセージ（非永続）	×
4			MCP メッセージ（永続）	○
5			タイマメッセージ（非永続）	×
6			タイマメッセージ（永続）※ 1	×

項番	システム構成（系）	キュー種別	メッセージ種別	出力対象
7			MCH メッセージ（非永続）	×
8			MCH メッセージ（永続）	○
9		OTQ	MCH メッセージ（非永続）	×
10			MCH メッセージ（永続）	○
11	CL サーバ（待機系）	ITQ（仮登録）※2	RPC メッセージ（永続）	○
12			MCP メッセージ（永続）	○
13			タイマメッセージ（永続）	×
14			MCH メッセージ（永続）	○
15		OTQ	MCH メッセージ（永続）	○
16	HA サーバ	ITQ	RPC メッセージ	×
17			MCP メッセージ	×
18			タイマメッセージ	×
19			MCH メッセージ	×
20		OTQ	MCH メッセージ（非永続）	×
21			MCH メッセージ（永続）	×

（凡例）

ITQ：入力キューを示します。

OTQ：出力キューを示します。

○：出力対象になります。

×：出力対象になりません。

注※ 1

タイマトランザクションの起動時刻前で，入力キュー（ITQ）に登録待ちとなっているメッセージを含みます。

注※ 2

処理キュー仮登録のメッセージに対応する XDB の更新ログ，および ee_mch_cmtsend 関数または ee_mch_cmtsend_sync 関数による一方送信メッセージを含みます。

（4） キューダンプファイル作成時の障害

キューダンプファイル作成時にファイルの I/O で障害が発生した場合，エラーメッセー

ジが出力されて，キューダンプファイルの作成は中止されます。また，オンラインは終了処理を中断してコマンド入力待ちとなります。TP1/EE 管理者は，障害要因を取り除いたあとに eetrbwtor コマンドで応答してください。eetrbwtor コマンドのオプションによって，キューダンプファイルの作成をリトライするか，キューダンプファイルを作成しないでオンラインを終了するか選択できます。

2.9.3 キューダンプファイルに出力されたメッセージの参照

キューダンプファイルに出力されたメッセージを参照するには，キューダンプファイルを編集します。キューダンプファイルの編集には eetrbqueed コマンドを使用します。

2.10　UAP のテスト支援機能

UAP の動作確認テストを支援する機能として，CL サーバのシミュレート機能があります。

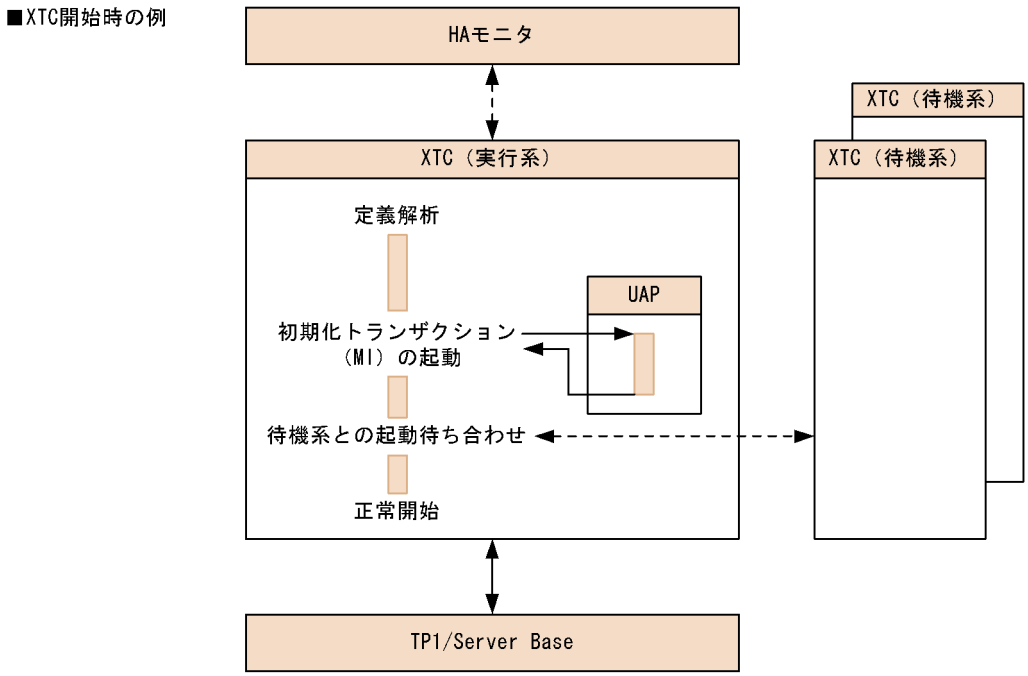
2.10.1　CL サーバのシミュレート機能

通常，CL サーバを実現するには，HA モニタ環境を構築し，3 台構成とする必要があります。しかし，CL **サーバのシミュレート機能**を使用することで，HA モニタ環境，および待機系 2 台の環境を構築しなくても，実行系 1 台だけで UAP の動作を確認できます。ただし，実行系で使用する TP1/Server Base の環境構築は必要です。

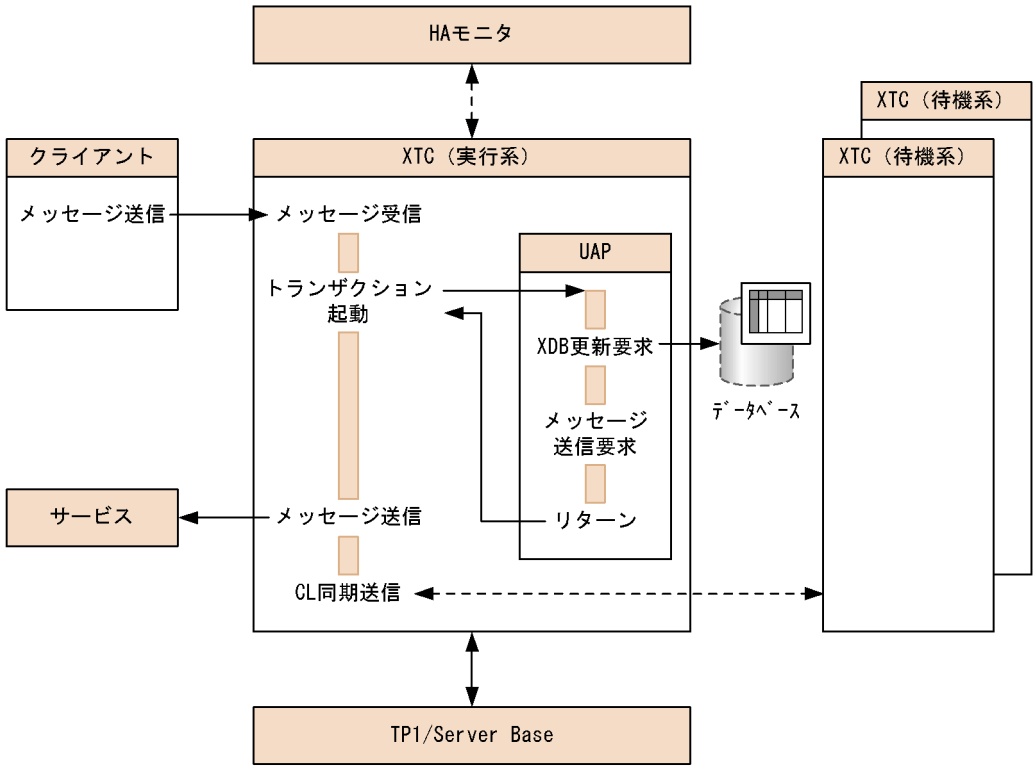
CL サーバのシミュレート機能を使用するには，クラスタ連携関連定義の cluster_test_mode オペランドに Y を指定します。また，クラスタ構成に関する定義を記述します。

CL サーバのシミュレート機能を使用すると，次の図に示す項目をシミュレートします。

図 2-47　CL サーバのシミュレート機能



■メッセージ送受信時の例



（凡例）

：制御の流れ

：シミュレートする項目

それぞれの項目について，CL サーバのシミュレート機能使用時の処理を次の表に示します。

表 2-28　CL サーバのシミュレート機能使用時の処理

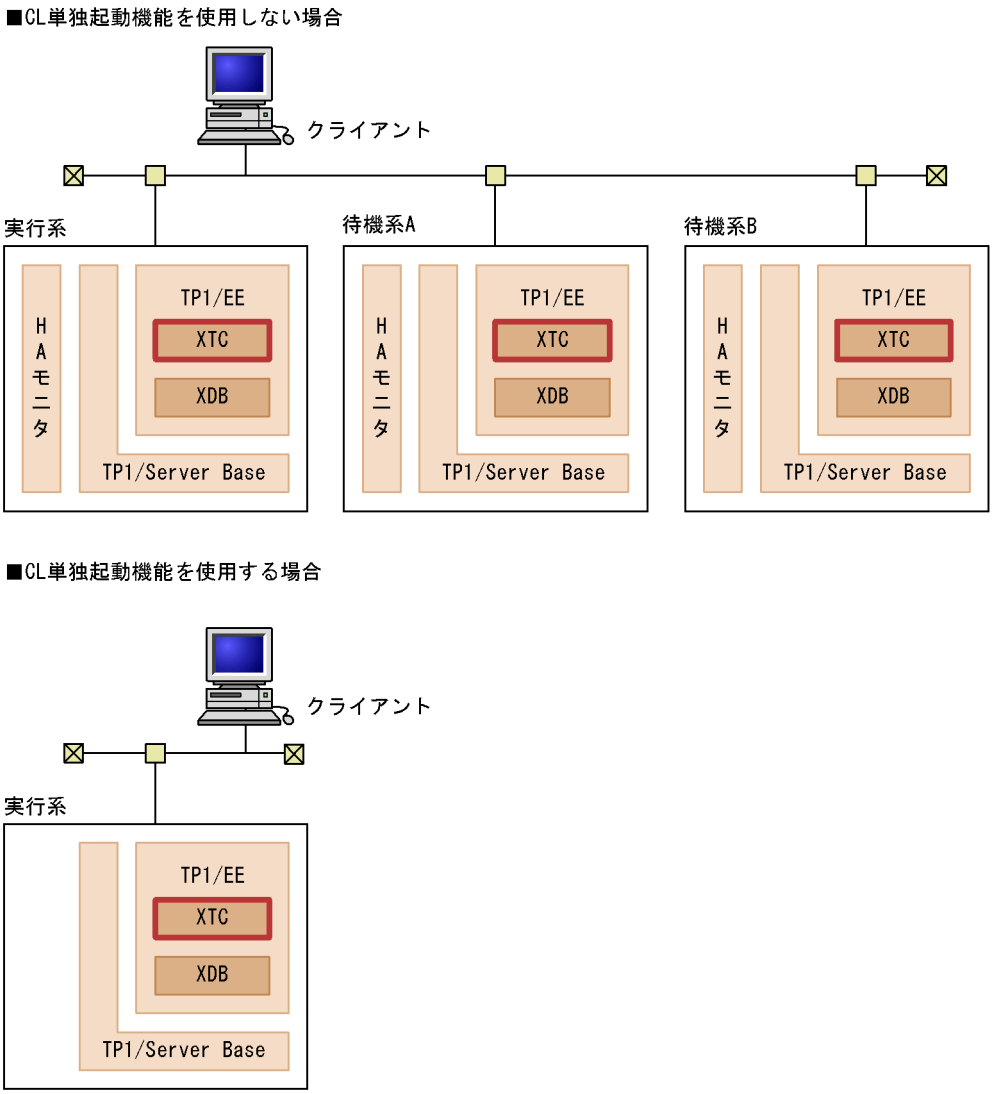
項番	項目	処理内容
1	HA モニタとの連携	HA モニタと連携する機能は，すべて実行しません。HA モニタから正常にリターンしたものとして処理します。 なお，eehamls コマンドでは，実行系だけとして表示します。
2	実行系と待機系の起動待ち合わせ	実行系と待機系の起動待ち合わせをしません。実行系の初期化処理が完了した際，待機系が起動したものとして処理します。
3	CL 同期通信などの待機系との通信	待機系に対する通信が成功したものとして処理します。

2.11　CL 単独起動機能

通常，CL サーバを実現するには，HA モニタ環境を構築し，XTC を配置した実行系 1 台，待機系 2 台が必要です。しかし，CL **単独起動機能**を使用することで，XTC を配置した実行系 1 台で，TP1 キャッシュ機能を実現できます。実行系の HA モニタ環境，および XTC の待機系は不要になります。

CL 単独起動機能を使用しない場合と使用する場合の構成を次の図に示します。

図 2-48　CL 単独起動機能を使用しない場合と使用する場合の構成



CL 単独起動機能には，次の特徴があります。

- 実行系だけの構成で，XDB のインメモリ DB を使用できます。
- CL サーバで待機系との通信が行われないため，CL 単独起動機能を使用しない場合と比べて性能が向上します。
  ただし，プロセスダウンした場合，永続メッセージ，およびインメモリ DB を保証しません。
- 正常終了，計画停止 A，および計画停止 B で，キューダンプを出力します。

(1) CL 単独起動機能を使用する場合と使用しない場合の相違点

CL 単独起動機能を使用する場合，使用しない場合と比べて次の点が異なります。

- HA モニタと連携しません。このため，HA モニタが前提プログラムではありません。
- クラスタ構成による系切り替えをしません。
- プロセスダウンしても待機系でキューダンプを出力しません。
- 待機系でのメッセージの永続化ができません。
- XTC の開始・終了時に，待機系に関連した処理を行いません。XTC の開始・終了時の待機系の処理については，「6.1.2　CL サーバでの XTC の開始」，および「6.2.2 CL サーバでの XTC の終了」を参照してください。

(2) CL 単独起動機能に関する定義

CL 単独起動機能を使用する場合，次のように定義してください。

- クラスタ連携関連定義の cluster_mode オペランドに Y を指定
- UDP グループ情報関連定義の clgrpdef 定義コマンドの指定を省略

(3) CL 単独起動機能使用時の注意事項

CL 単独起動機能を使用する場合，次のことに注意してください。

- eehamls コマンドを実行した場合，実行系だけが表示されます。
- CL 単独起動機能では，HA モニタ構成と CL 同期をシミュレートする必要がありません。このため，CL サーバのシミュレート機能は使用できません。

3 XTC が扱うファイルの種類と容量見積もり

この章では，XTC が扱うファイルの種類と容量見積もりについて説明します。

3.1 XTC が扱うファイルの種類

XTC では，TP1/EE が扱うファイル以外に，次の表に示すファイルを扱います。

表 3-1 XTC が扱うファイル

項番	ファイル名称	内容	ファイル形式
1	キューダンプファイル	キューダンプを取得するファイルです。	UNIX ファイル

TP1/EE が扱うファイルについては，マニュアル「TP1/Server Base Enterprise Option 使用の手引」を参照してください。

3.2　ファイルの容量見積もり

ファイルの容量見積もりについては，マニュアル「TP1/Server Base Enterprise Option 使用の手引」を参照してください。

ここでは，XTC が独自に扱うファイル，および見積もり方法に変更のあるファイルについて説明します。

3.2.1　キューダンプファイルの容量見積もり

キューダンプファイルの容量は，次の計算式で求めます。

```
全体のキューダンプファイルサイズ（T）＝
    ↑（416＋（516＋（1044＋M）×↑（L÷M）↑）×I
    ＋（（1044＋N）×↑（L÷N）↑）×Q）×1.2（安全率）↑
```

（凡例）

L：平均メッセージ長

M：udp_recv_message_buf_size オペランドの指定値

N：udp_send_message_buf_size オペランドの指定値

I：入力キュー（ITQ）の滞留メッセージ数

Q：出力キュー（OTQ）の滞留メッセージ数

3.2.2　TASKTM ファイルの容量見積もり

TASKTM ファイルのサイズの見積もりに必要な情報を次に示します。ここで求める値を，TP1/EE の見積もりサイズに加算してください。

- トランザクション単位のバイト数：512
- 1 ブロック単位に付加する情報のバイト数：96
- 1 ファイル単位に付加する情報のバイト数：128 ＋ 34 ×（サービス数）＋ 256

全体の TASKTM ファイルのサイズは次の計算式で求めます。

```
全体のTASKTMファイルのサイズ（T）＝（512×x）
                                  ＋（↑（（512×x）÷b）↑×96）
                                  ＋（128＋34×s＋256）
```

（凡例）

b：trb_tasktm_buffer_size オペランドで指定する値

s：サービス数

x：トランザクション数

また，トランザクション数を見積もる場合の参考値は，システム構成によって異なります。なお，次に示す計算式は参考値です。実際には余裕を持って見積もってください。

■ HA サーバの場合

トランザクション数＝受信メッセージ数（MCH）×2
＋受信メッセージ数（RPC，MCP）
＋ee_tim_execap関数のAPI発行回数
＋ee_mch_cmtsend関数のAPI発行回数

■ CL サーバの場合

トランザクション数＝受信メッセージ数（MCH）×2※1
＋受信メッセージ数（RPC）※2
＋受信メッセージ数（MCP）※3
＋ee_mch_cmtsend関数のAPI発行回数×2
＋ee_tim_execap関数のAPI発行回数×3
＋転送対象コマンド発行回数※4
＋転送対象API発行回数※4

注※ 1

受信メッセージを永続化している場合は，さらに 2 倍します。

注※ 2

RPC 関連定義で受信メッセージを永続化する指定（rpc_recv_permanence オペランドに Y を指定）の場合は，4 倍します。

注※ 3

MCP 構成定義で受信メッセージを永続化する指定（eemcpfunc 定義コマンドの -m オプションの recvpermanence オペランドに yes を指定）の場合は，4 倍します。

注※ 4

転送対象コマンドおよび転送対象 API とは，次のコマンドおよび API です。

- スレッドダウンによるサービス自動閉塞
- eedctsv コマンド，eeactsv コマンド，または ee_thd_abdctl 関数によるサービス閉塞状態の変更
- 一方送信メッセージ障害による出力キュー（OTQ）の自動閉塞
- eemchotqact コマンド，eemchotqdct コマンド，または ee_mch_otqbkctl 関数による出力キュー閉塞状態の変更
- eelspce コマンドによるサービスの最大同時処理限界数の変更
- eemchotqskip コマンドまたは ee_mch_otqskip 関数による出力キューメッセージのスキップ
- eepceskip コマンドまたは ee_scd_msg_skip 関数による滞留メッセージのス

キップ

3.2.3　回線トレースファイルの容量見積もり

(1) HA サーバの場合

HA サーバの場合に，回線トレースファイルのサイズの見積もりに必要な情報を次に示します。ここで求める値を，TP1/EE の見積もりサイズに加算してください。

- トランザクション単位のバイト数：328
- メッセージ受信単位のバイト数：600
- 1 ブロック単位に付加する情報のバイト数：96
- 1 ファイル単位に付加する情報のバイト数：128 ＋ 34 ×（サービス数）＋ 256
- パケット数：（↑メッセージ長÷（パケットサイズ※ 1）↑＋ 1）※ 2

注※ 1

パケットサイズは次の計算式で求めます。

rpc_udp_packet_size**オペランドの指定値**－96

注※ 2

W-send 機能使用時は 2 倍にしてください。

全体の回線トレースファイルのサイズは次の計算式で求めます。

全体の回線トレースファイルのサイズ（T）＝ （328×x）＋（600×y1）＋（600×y2）＋（600×p×z） ＋（↑（（（328×x）＋（600×y1）＋（600×p×y2） ＋（600×p×z））÷b）↑×96） ＋（128＋34×（s）＋256）

（凡例）

b：バッファサイズ※

p：パケット数

s：サービス数

x：トランザクション数

y1：受信メッセージ数（RPC メッセージ）

y2：受信メッセージ数（MCH メッセージ）

z：送信メッセージ数

注※

バッファサイズには次に示す値を指定します。

- トラブルシュート関連定義の trb_extend_function オペランドに 00000001 を論理和で指定した場合
  trb_line_trace_buffer_size オペランドでの指定値と
  trb_line_cmtrace_buffer_size オペランドでの指定値の小さい方
- トラブルシュート関連定義の trb_extend_function オペランドを省略するか，または 00000000 を論理和で指定した場合
  trb_line_trace_buffer_size オペランドでの指定値

（2）CL サーバの場合

CL サーバの場合に，回線トレースファイルのサイズの見積もりに必要な情報を次に示します。ここで求める値を，TP1/EE の見積もりサイズに加算してください。

- トランザクション単位のバイト数：328
- メッセージ受信単位のバイト数：600
- メッセージ送信単位のバイト数：600
- 送達確認メッセージ送信単位のバイト数：1800
- 1 ブロック単位に付加する情報のバイト数：96
- 1 ファイル単位に付加する情報のバイト数：128 ＋ 34 ×（サービス数）
- パケット数：（↑メッセージ長÷（パケットサイズ※ 1）↑＋ 1）※ 2

注※ 1

パケットサイズは次の計算式で求めます。

rpc_udp_packet_size**オペランドの指定値**－96

注※ 2

W-send 機能使用時は 2 倍にしてください。

- XDB の更新ログのバイト数：D

$$\text{XDBの更新ログのバイト数（D）} = \sum_{k=1}^{Ni} Lk + \sum_{k=1}^{Nu} Mk + 48 \times (Ni + Nu + Nd) + 40$$

（凡例）

Ni：挿入（INSERT）した行の数
Nu：更新（UPDATE）した行の数
Nd：削除（DELETE）した行の数
Lk：k 番目に挿入（INSERT）した行の行長
Mk：k 番目に更新（UPDATE）した行の行長

全体の回線トレースファイルのサイズは次の計算式で求めます。

全体の回線トレースファイルのサイズ（T）＝
（328×x1）＋（600×p×y）＋（600×p×z）
＋（600×p×t）＋（600×↑D÷8388608↑×x2）
＋1644×（x1＋x2＋y＋z＋t）
＋（↑（（（328×x1）＋（600×p×y）＋（600×p×z）
＋（600×p×t）＋（600×↑D÷8388608↑×x2）
＋1644×（x1＋x2＋y＋z＋t））÷b）↑×96）
＋（128＋34×（s））

（凡例）

D：XDB の更新ログのバイト数

b：バッファサイズ※

p：パケット数

s：サービス数

t：ee_tim_execap 関数（永続指定）を発行した回数

x1：トランザクション数（SQL 発行なし）

x2：トランザクション数（SQL 発行あり）

y：受信メッセージ数

z：送信メッセージ数

注※

バッファサイズには次に示す値を指定します。

- トラブルシュート関連定義の trb_extend_function オペランドに 00000001 を論理和で指定した場合
  trb_line_trace_buffer_size オペランドでの指定値と
  trb_line_cmtrace_buffer_size オペランドでの指定値の小さい方
- トラブルシュート関連定義の trb_extend_function オペランドを省略するか，または 00000000 を論理和で指定した場合
  trb_line_trace_buffer_size オペランドでの指定値

3.2.4 メモリダンプファイルの容量見積もり

メモリダンプファイルの見積もり方法は，TP1/EE サービス定義のトラブルシュート関連定義の trb_dump_area_kind オペランド（メモリダンプファイル出力領域種別）の指定によって異なります。

■ trb_dump_area_kind=13 の場合

次の計算式で求めます。

（max_mem_sizeオペランドまたはmax_mem_size_mbオペランドの指定値
　－user_work_sizeオペランドの指定値
　－368×pce_noオペランドの指定値
　－144×（10＋icb_noオペランドの指定値）
　－time_message_sizeオペランドの指定値×time_message_noオペランドの指定値
　－（recv_message_buf_sizeオペランドの指定値＋1040）
　　×recv_message_buf_cntオペランドの指定値
　－（send_message_buf_sizeオペランドの指定値＋1040）
　　×send_message_buf_cntオペランドの指定値
　－（udp_recv_message_buf_sizeオペランドの指定値＋1040）
　　×udp_recv_message_buf_cntオペランドの指定値
　－（udp_send_message_buf_sizeオペランドの指定値＋1040）
　　×udp_send_message_buf_cntオペランドの指定値
　－（max_message_sizeオペランドの指定値＋2264）
　　×（thread_noオペランドの指定値＋reserve_thread_noオペランドの指定値＋1）
）

■ trb_dump_area_kind=9 の場合

次の計算式で求めます。

（max_mem_sizeオペランドまたはmax_mem_size_mbオペランドの指定値
　－user_work_sizeオペランドの指定値
　－system_work_sizeオペランドの指定値
　－memory_cobol_area_sizeオペランドの指定値
　－memory_xtc_area_sizeオペランドの指定値
　－memory_xdb_area_sizeオペランドの指定値
　－368×pce_noオペランドの指定値
　－144×（10＋icb_noオペランドの指定値）
　－time_message_sizeオペランドの指定値×time_message_noオペランドの指定値
　－（recv_message_buf_sizeオペランドの指定値＋1040）
　　×recv_message_buf_cntオペランドの指定値
　－（send_message_buf_sizeオペランドの指定値＋1040）
　　×send_message_buf_cntオペランドの指定値
　－（udp_recv_message_buf_sizeオペランドの指定値＋1040）
　　×udp_recv_message_buf_cntオペランドの指定値
　－（udp_send_message_buf_sizeオペランドの指定値＋1040）
　　×udp_send_message_buf_cntオペランドの指定値
　－（max_message_sizeオペランドの指定値＋2264）
　　×（thread_noオペランドの指定値＋reserve_thread_noオペランドの指定値＋1）
）

■注意事項

大量処理用メモリ管理機能を使用している場合，計算式で求めたメモリダンプファイルのサイズに，大量処理用メモリ管理機能で確保した大量処理用メモリ領域のサイズを加算してください。

大量処理用メモリ領域のサイズは，メモリダンプ出力時に eememls コマンドで表示される「大量処理用システム領域（MDP system area）」と「大量処理用ユーザ領域（MDP user area）」の領域確保サイズを合算した値です。

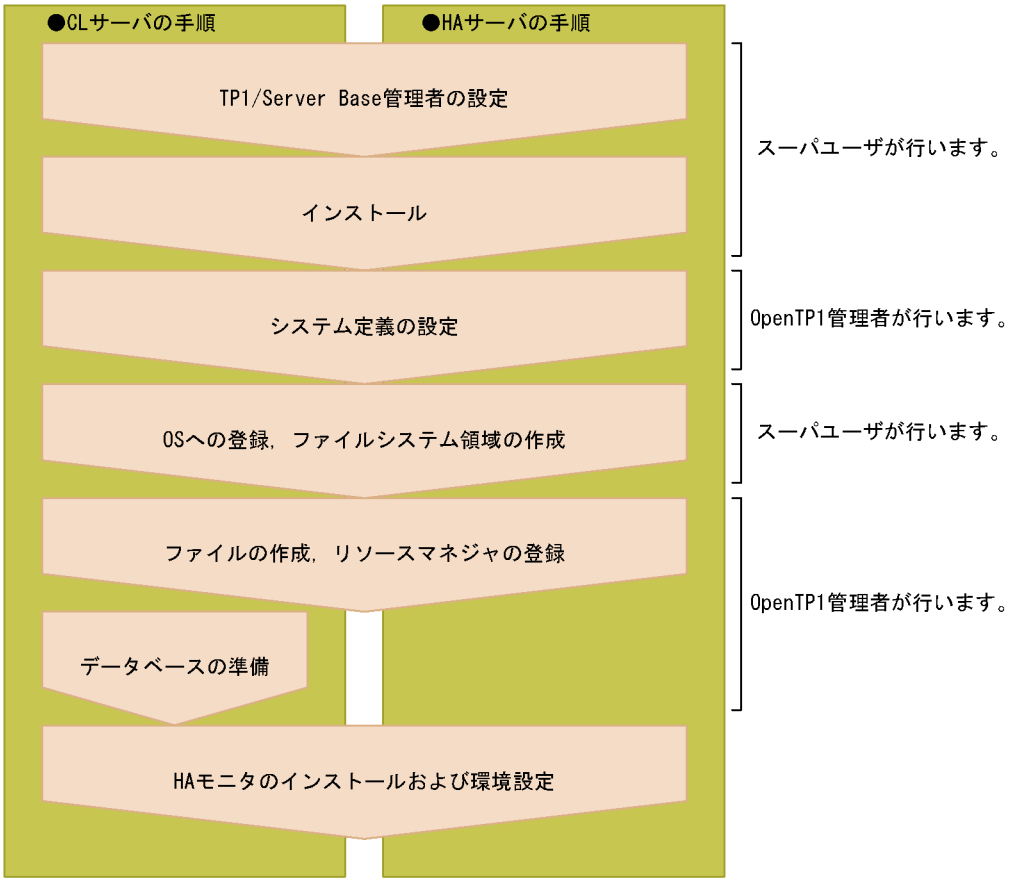
4 環境設定

この章では，TP1 キャッシュ機能使用時の環境設定とその手順について説明します。

4.1　環境設定の概要

TP1 キャッシュ機能使用時の環境設定は，CL サーバの場合と HA サーバの場合とで異なります。手順を次の図に示します。

図 4-1　TP1 キャッシュ機能使用時の環境設定の手順



各項目の作業については，次節以降で説明します。

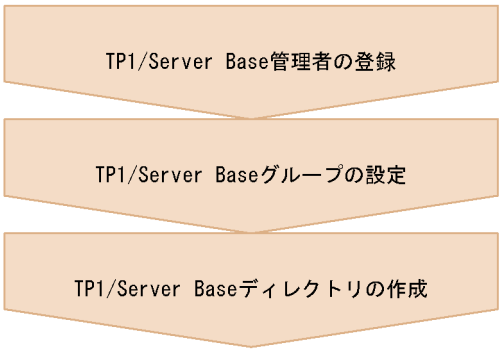
CL サーバの場合は，CL サーバを構成する全マシンに対して同じ手順で環境設定を行ってください。

HA サーバの場合は，HA サーバを構成する全マシンに対して同じ手順で環境設定を行ってください。

4.2　TP1/Server Base 管理者の設定

TP1/Server Base 管理者の設定手順を次の図に示します。CL サーバの場合も HA サーバの場合も，設定手順は同じです。

図 4-2　TP1/Server Base 管理者の設定手順

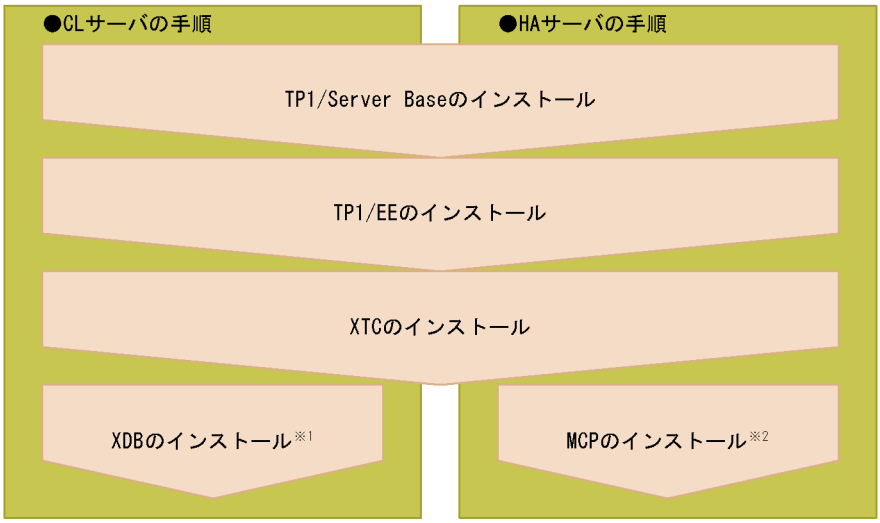


詳細については，マニュアル「OpenTP1 運用と操作」を参照してください。

4.3　インストール

インストール手順を次の図に示します。

図 4-3　インストール手順



注※1　XDBを使用する場合に必要な手順です。

注※2　MCPを使用する場合に必要な手順です。

TP1/Server Base のインストールについては，マニュアル「OpenTP1 運用と操作」を参照してください。

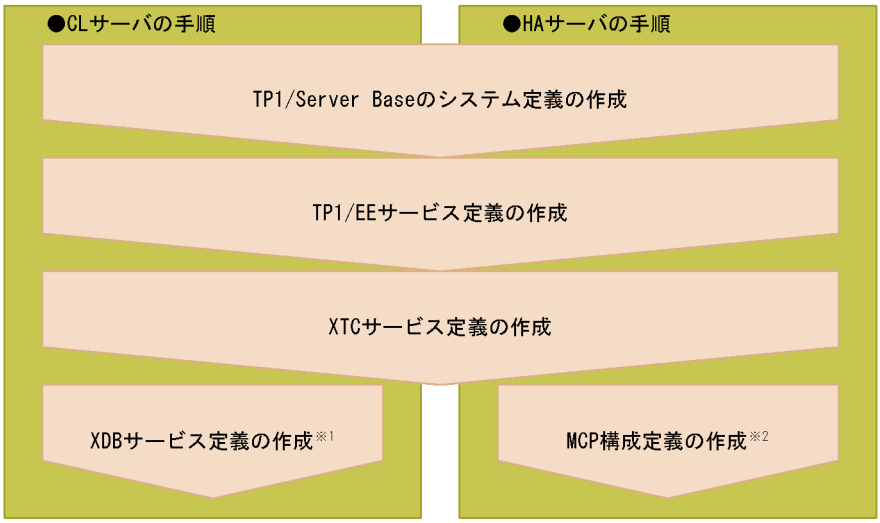
TP1/EE のインストールについては，マニュアル「TP1/Server Base Enterprise Option 使用の手引」を参照してください。

それ以外については，各製品のリリースノートを参照してください。

4.4　システム定義の設定

システム定義の設定手順を次の図に示します。

図 4-4　システム定義の設定手順



注※1　XDBを使用する場合に必要な手順です。
注※2　MCPを使用する場合に必要な手順です。

TP1/Server Base のシステム定義の作成については，マニュアル「OpenTP1 システム定義」およびマニュアル「TP1/Server Base Enterprise Option 使用の手引」を参照してください。

TP1/EE サービス定義の作成については，マニュアル「TP1/Server Base Enterprise Option 使用の手引」を参照してください。

XTC サービス定義の作成については，「5.　システム定義」を参照してください。

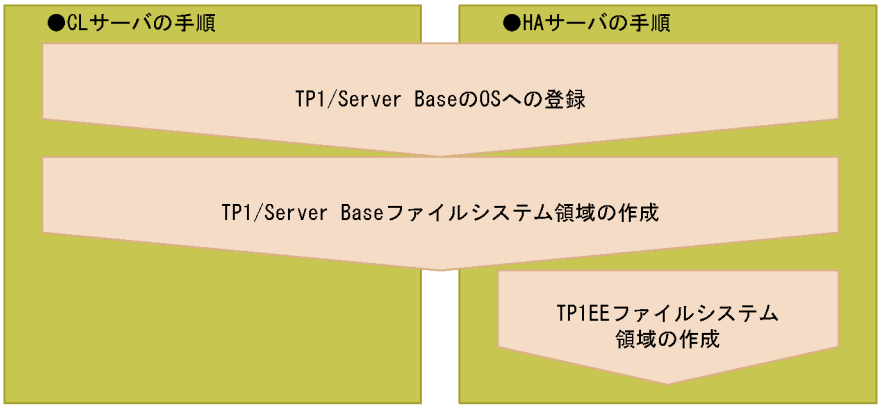
MCP 構成定義の作成については，マニュアル「TP1/EE/Message Control Extension 使用の手引」を参照してください。

XDB サービス定義の作成については，マニュアル「TP1/EE/Extended Data Cache 使用の手引」を参照してください。

4.5　OS への登録，ファイルシステム領域の作成

OS への登録，およびファイルシステム領域の作成手順を次の図に示します。

図 4-5　OS への登録，およびファイルシステム領域の作成手順

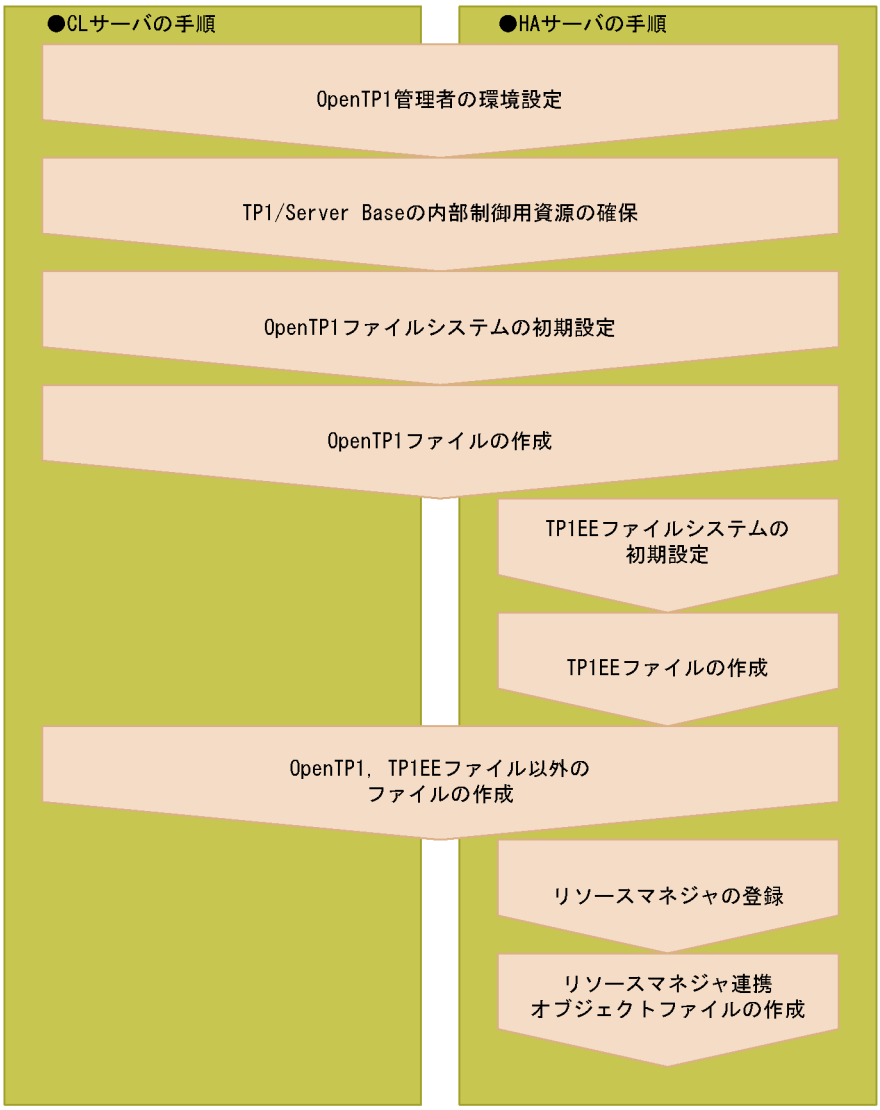


詳細については，マニュアル「OpenTP1 運用と操作」およびマニュアル「TP1/Server Base Enterprise Option 使用の手引」を参照してください。

4.6　ファイルの作成，リソースマネジャの登録

ファイルの作成，およびリソースマネジャの登録手順を次の図に示します。

図 4-6　ファイルの作成，およびリソースマネジャの登録手順



詳細については，マニュアル「OpenTP1 運用と操作」およびマニュアル「TP1/Server Base Enterprise Option 使用の手引」を参照してください。

4.7 データベースの準備

CL サーバではデータベースの準備が必要になります。データベースの準備については，マニュアル「TP1/EE/Extended Data Cache 使用の手引」の「環境設定」を参照してください。

4.8 HA モニタの環境設定

CL サーバと HA サーバでは，HA モニタの環境設定方法が異なります。ここでは，サーバごとの HA モニタの環境設定方法について説明します。

4.8.1 CL サーバの場合

CL サーバでは，HA モニタのマルチスタンバイ機能を使用して，クラスタ型系切り替え構成を構築します。マルチスタンバイ機能を使用したときの HA モニタの環境設定については，マニュアル「高信頼化システム監視機能 HA モニタ Linux(R) 編」を参照してください。

ここでは，環境設定時の注意点を説明します。

(1) 環境設定時の注意点

実行系と待機系の実行環境を同じにしてください。例えば，次に示すことを一致させてください。一致していないと，系が正常に切り替わりません。

- システム定義の内容
- UAP の実行形式プログラム
- トランザクションサービス制御用実行形式プログラム（trnlnkrm コマンドで再作成した場合）
- TP1/EE および関連製品のバージョン
- TP1/EE 管理者の環境（ユーザ ID，グループ ID，環境変数など）
- TP1/EE ディレクトリの絶対パス名

(2) TP1/Server Base のシステムサービス定義を作成するときの注意点

TP1/Server Base のシステムサービス定義を作成するときの注意点を次に示します。

- システムサービス構成定義の ha_conf オペランドに N を指定し，TP1/Server Base に対して系切り替え機能を適用しないようにしてください。
  CL サーバでは，TP1/EE プロセスを HA モニタが直接監視します。そのため，TP1/Server Base を HA モニタの監視対象から外す必要があります。
- ユーザサービス定義の hold オペランドに Y を指定し，TP1/EE プロセスが異常終了した場合，サービスグループまたはサービスを閉塞するようにしてください。

(3) HA モニタの定義ファイルを作成するときの注意点

HA モニタの定義ファイルを作成するときの注意点を次に示します。

- 次に示すオペランドに指定するホスト名を一致させてください。
  - UDP グループ情報関連定義の clgrpdef 定義コマンドの -n オプションに指定するホスト名

- HA モニタの environment 定義文の name オペランドに指定するホスト名

- 次に示すオペランドに指定するクラスタ識別子を一致させてください。
  - UDP グループ情報関連定義の clgrpdef 定義コマンドの -c オプションに指定するクラスタ識別子
  - HA モニタの server 定義文の name オペランドに指定するクラスタ識別子
- TP1/EE プロセスを直接監視するため，HA モニタの server 定義文の server_type オペランドに B を指定してください。
- HA モニタのマルチスタンバイ機能を使用するため，HA モニタの function 定義文の multistandby オペランドに use を指定してください。

HA モニタの定義ファイルの指定例を次に示します。

■実行系の定義例

- OpenTP1 および XTC の定義例

```
set node_id = ap01
   :
clgrpdef -p 10001 -m path01 -n ap01:host01,ap02:host02,ap03:host03 -c cl01
                                    [1]                              [2]
```

- HA モニタの定義例

```
environment    name host01,  ...[1]
   :
function
               multistandby    use,  ...[3]
   :
server    name cl01,          ...[2]
          alias server01,
          acttype server,
          server_type B,      ...[4]
          initial online,     ...[5]
   :
```

説明

1. 同じホスト名を指定します。
2. 同じクラスタ識別子を指定します。
3. マルチスタンバイ機能を使用するため，use を指定します。
4. TP1/EE プロセスを直接監視するため，B を指定します。
5. CL サーバの開始時にこのサーバを実行系にするため，online を指定します。

■待機系 1 の定義例

- OpenTP1 および XTC の定義例

```
set node_id = ap02
   :
clgrpdef -p 10001 -m path01 -n ap01:host01,ap02:host02,ap03:host03 -c cl01
                                          [1]                  [2]
```

- HA モニタの定義例

```
environment    name host02,  ...[1]
   :
function
               multistandby    use,  ...[3]
   :
server    name cl01,          ...[2]
          alias server01,
          acttype server,
          server_type B,      ...[4]
          standbypri 1,       ...[5]
          initial standby,    ...[6]
   :
```

説明

1. 同じホスト名を指定します。
2. 同じクラスタ識別子を指定します。
3. マルチスタンバイ機能を使用するため，use を指定します。
4. TP1/EE プロセスを直接監視するため，B を指定します。
5. 待機系の優先度を指定します。1 を指定しているため，最初の系切り替えが発生した場合，このサーバが実行系になります。
6. CL サーバの開始時にこのサーバを待機系にするため，standby を指定します。

■待機系 2 の定義例

- OpenTP1 および XTC の定義例

```
set node_id = ap03
   :
clgrpdef -p 10001 -m path01 -n ap01:host01,ap02:host02,ap03:host03 -c cl01
                                                              [1]       [2]
```

- HA モニタの定義例

```
environment    name host03,  ...[1]
   :
function
               multistandby    use,  ...[3]
   :
server    name cl01,          ...[2]
          alias server01,
          acttype server,
          server_type B,      ...[4]
          standbypri 2,       ...[5]
          initial standby,    ...[6]
   :
```

説明

1. 同じホスト名を指定します。
2. 同じクラスタ識別子を指定します。

3. マルチスタンバイ機能を使用するため，use を指定します。
4. TP1/EE プロセスを直接監視するため，B を指定します。
5. 待機系の優先度を指定します。2 を指定しているため，最初の系切り替えが発生した場合，このサーバは待機系のままです。
6. CL サーバの開始時にこのサーバを待機系にするため，standby を指定します。

4.8.2 HA サーバの場合

HA サーバの場合，TP1/EE が直接 HA モニタと連携することはありません。系切り替え時の TP1/EE の動作は，TP1/Server Base の設定に依存します。

HA モニタの環境設定については，マニュアル「高信頼化システム監視機能 HA モニタ Linux(R) 編」およびマニュアル「TP1/Server Base Enterprise Option 使用の手引」を参照してください。

5 システム定義

この章では，TP1 キャッシュ機能使用時のシステム定義について説明します。

5.1 システム定義の概要

ここでは，TP1 キャッシュ機能使用時のシステム定義の体系，定義の構成，定義の規則，および注意事項について説明します。

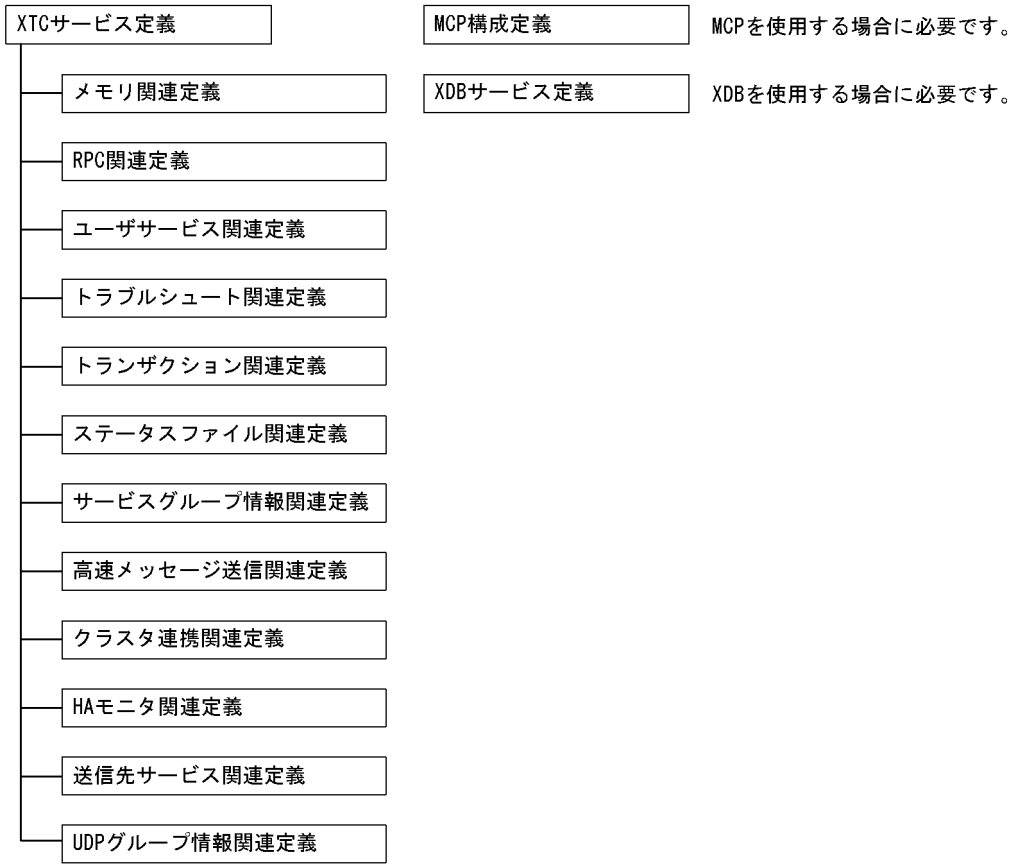
5.1.1 定義の体系

TP1 キャッシュ機能使用時の定義は，次に示す定義で構成されます。

- TP1/Server Base のシステムサービス定義
- TP1/EE サービス定義
- XTC サービス定義（XTC の定義）
- MCP 構成定義（MCP の定義）
- XDB サービス定義（XDB の定義）

TP1 キャッシュ機能使用時の定義体系を次の図に示します。TP1/Server Base のシステムサービス定義，および TP1/EE サービス定義の体系の詳細については，マニュアル「OpenTP1 システム定義」およびマニュアル「TP1/Server Base Enterprise Option 使用の手引」を参照してください。

図 5-1　TP1 キャッシュ機能使用時の定義体系



MCP 構成定義については，マニュアル「TP1/EE/Message Control Extension 使用の手引」を参照してください。

XDB サービス定義については，マニュアル「TP1/EE/Extended Data Cache 使用の手引」を参照してください。

5.1.2　定義の構成

XTC サービス定義は，一つのファイルに定義します。TP1/EE システムの動作する環境を TP1/EE（SPP）ごとに定義します。

XTC サービス定義の概要を次の表に示します。

表 5-1　XTC サービス定義の概要

項番	定義名	内容
1	メモリ関連定義	使用するメモリの関連情報を定義します。
2	RPC 関連定義	RPC 関連情報を定義します。

項番	定義名	内容
3	ユーザサービス関連定義	UAP 関連情報を定義します。
4	トラブルシュート関連定義	トラブルシュートおよびトレース関連情報を定義します。
5	トランザクション関連定義	トランザクション関連情報を定義します。
6	ステータスファイル関連定義	ステータスファイル関連情報を定義します。
7	サービスグループ情報関連定義	サービスグループ関連情報を定義します。
8	高速メッセージ送信関連定義	高速メッセージ送信制御の関連情報を定義します。
9	クラスタ連携関連定義	CL サーバでの，クラスタ連携の関連情報を定義します。
10	HA モニタ関連定義	CL サーバでの，HA モニタの関連情報を定義します。
11	送信先サービス関連定義	送信先サービスの関連情報を定義します。
12	UDP グループ情報関連定義	UDP グループの関連情報を定義します。

5.1.3 定義の規則

XTC サービス定義を作成する場合は，テキストエディタを使用して定義ファイルを作成します。TP1/EE サービス定義と同じ定義ファイルの中に定義します。

また，XTC サービス定義の規則は TP1/EE サービス定義の規則に準じます。詳細については，マニュアル「TP1/Server Base Enterprise Option 使用の手引」を参照してください。

5.1.4 CL サーバでの注意事項

CL サーバを構成する各 TP1/EE の定義は，記述順序も含めて，基本的には同じにしてください。ただし，サーバごとに異なる値の設定が必要な定義（自ホスト名など，ネットワークに関連する定義）は除きます。

また，CL サーバでは，構成するすべての TP1/EE で異なる値を指定しなければならないオペランドがあります。詳細を次の表に示します。

表 5-2 CL サーバを構成するすべての TP1/EE で異なる値を指定しなければならないオペランド

項番	定義		異なる値を指定しなければならないオペランド
1	TP1/EE サービス定義	RPC 関連定義	node_id オペランド

また，CL サーバでは指定できないオペランドを次の表に示します。これらを指定すると，定義エラーとなります。

表 5-3 CL サーバでは指定できないオペランド

項番	定義		指定できないオペランド
1	TP1/EE サービス定義	トランザクション関連定義	trnstring オペランド
2	XTC サービス定義	トランザクション関連定義	• trn_transactional_rpcless_use オペランドに N を指定 • trn_transactional_rpcless_use オペランドを省略
3		ステータスファイル関連定義	• sts_fileless_use オペランドに N を指定 • sts_fileless_use オペランドを省略
4			sts_fileless_level オペランドに 2 を指定

5.2 TP1/EE サービス定義

TP1 キャッシュ機能を使用する場合，TP1/EE サービス定義の中で指定できないオペランドや定義コマンドがあります。ここでは，TP1 キャッシュ機能使用時に指定できないオペランドおよび定義コマンドについて説明します。

TP1/EE サービス定義の各オペランドおよび定義コマンドの内容については，マニュアル「TP1/Server Base Enterprise Option 使用の手引」を参照してください。

5.2.1 TP1 キャッシュ機能使用時に指定できない TP1/EE サービス定義

TP1 キャッシュ機能使用時に指定できない TP1/EE サービス定義の一覧を次の表に示します。

表 5-4 TP1 キャッシュ機能使用時に指定できない TP1/EE サービス定義

項番	定義	指定できないオペランドおよび定義コマンド
1	プロセス関連定義	• dbq_use • dbq_obs_use • process_kind • online_process_cnt • master_svgname
2	メモリ関連定義	• dbq_recv_message_buf_cnt • dbq_send_message_buf_cnt • dbq_max_message_size • dbq_use_buf_cnt
3	RPC 関連定義	• ipc_sendbuf_size_dbq • ipc_recvbuf_size_dbq • ipc_backlog_count_dbq • ipc_tcpnodelay_dbq • rpc_first_connect_errmsg_dbq • rpc_firstmsg_recv_timer_dbq • rpc_nam_server_list • rpc_tcpsend_con_cnt • rpc_tcpsend_con_max_cnt • rpc_tcpsend_proc_max_cnt
4	ユーザサービス関連定義	• dbq_service • dbq_obs_service

項番	定義	指定できないオペランドおよび定義コマンド
5	DB キュー機能関連定義	• dbq_reqbuf_retry_timer • dbq_readcheckmsg_interval • dbq_endcheckmsg_interval • dbq_rollback_retry_count • dbqdef 定義コマンド • dbqgrpdef 定義コマンド • dbqsrvdef 定義コマンド • dbqsvgdef 定義コマンド • dbqprcdef 定義コマンド
6	オンラインバッチ機能関連定義	• dbq_obs_lot_no • dbq_obs_endcheckmsg_interval • dbq_obs_trn_end_api • dbqobssrvdef 定義コマンド • dbqobsdef 定義コマンド • dbqobslotdef 定義コマンド

5.2.2 CL サーバで指定できない TP1/EE サービス定義

CL サーバで指定できない TP1/EE サービス定義の一覧を次の表に示します。

表 5-5 CL サーバで指定できない TP1/EE サービス定義の一覧

項番	定義	指定できないオペランドおよび定義コマンド
1	プロセス関連定義	• service_hold_watch_procdown • fil_filesystem_no • sys_dba_waittime
2	RPC 関連定義	• rpc_reply_suspend_recover • rpc_reply_suspend_autosend
3	トランザクション関連定義	• trn_max_subordinate_count • trn_optimum_item • trn_watch_time • trn_wait_rm_open • trn_retry_interval_rm_open • trn_retry_count_rm_open • trn_retry_interval_rm_open_mime • trn_retry_count_rm_open_mime • trn_delayed_allrecover • trn_endrerun_msg_interval • trn_thd_down_count • trn_thd_down_interval • trn_unknown_rm_tran_down • trn_max_commit_count • trn_max_commit_downmode • trn_communication_tp1ee0101 • trn_undelayed_recover_thread • trn_rm_open_close_scope • trnstring 定義コマンド

項番	定義	指定できないオペランドおよび定義コマンド
4	ステータスファイル関連定義	• sts_initial_error_switch • sts_single_operation_switch • sts_buffer_count • sts_control_buffer_length • sts_signal_buffer_length • stsflgrp 定義コマンド • stsflnam 定義コマンド
5	ファイルサービス関連定義	• fil_watch_time • fil_watch_timeout

5.2.3 HA サーバで指定できない TP1/EE サービス定義

HA サーバで指定できない TP1/EE サービス定義の一覧を次の表に示します。

表 5-6 HA サーバで指定できない TP1/EE サービス定義の一覧

項番	定義	指定できないオペランド
1	プロセス関連定義	• xdb_use

5.3 XTC サービス定義

ここでは，XTC サービス定義を説明します。

メモリ関連定義

ここでは，メモリ関連定義の各オペランドについて説明します。

なお，メモリ関連定義のオペランドには，HA サーバで指定できないオペランドがあります。HA サーバで指定できないオペランドについては，「HA サーバで指定できない XTC サービス定義」を参照してください。

形式

〔set udp_recv_message_buf_size＝UDP用受信バッファサイズ〕
〔set udp_recv_message_buf_cnt＝UDP用受信バッファ面数〕
〔set udp_send_message_buf_size＝UDP用送信バッファサイズ〕
〔set udp_send_message_buf_cnt＝UDP用送信バッファ面数〕
 set memory_xdb_area_size＝初期化確保時のXDB用ワーク領域サイズ,追加確保時のXDB用ワーク領域サイズ
 set memory_xdb_limit_size＝確保するXDB用ワーク領域の最大サイズ
 set memory_xtc_area_size＝初期化確保時のXTC用ワーク領域サイズ,追加確保時のXTC用ワーク領域サイズ
 set memory_xtc_limit_size＝確保するXTC用ワーク領域の最大サイズ
〔set memory_mdpsys_area_size＝初期化確保時の大量処理用システム領域サイズ,追加確保時の大量処理用システム領域サイズ〕
〔set memory_mdpusr_area_size＝初期化確保時の大量処理用ユーザ領域サイズ,追加確保時の大量処理用ユーザ領域サイズ〕
〔set mem_ibf_extend_num＝受信バッファの最大拡張数〕
〔set mem_uibf_extend_num＝UDP用受信バッファの最大拡張数〕
〔set mem_obf_extend_num＝送信バッファの最大拡張数〕
〔set mem_uobf_extend_num＝UDP用送信バッファの最大拡張数〕

オペランドの説明

● udp_recv_message_buf_size ＝ UDP 用受信バッファサイズ

～〈符号なし整数〉((2000 ～ 66000))《33000》（単位：バイト）

UDP 用受信バッファ（UIBF）のサイズをバイト単位で指定します。

UDP 用受信バッファサイズには，クライアントの UDP 用送信バッファサイズ以上の値を指定することをお勧めします。

参考

UDP 用受信バッファは，UDP プロトコルを使用する通信で使用します。UDP プロトコルを使用する通信を次に示します。

- トランザクション同期の一方送信メッセージの受信
- トランザクション非同期の一方送信メッセージの受信
- CL サーバ間の通信
  - ・定義情報の受信
  - ・CL 同期メッセージの受信（XDB の更新ログ情報を含む）
  - ・CL 同期済み通知メッセージの受信
  - ・実行系ダウン時のリソース回復用メッセージの受信
- RPC サービス要求の受信

● udp_recv_message_buf_cnt ＝ UDP **用受信バッファ面数**

～〈符号なし整数〉((1 ～ 2000000))《128》

UDP 用受信バッファの面数を指定します。

UDP 用受信バッファは，クライアントからメッセージを受信してから UAP 実行前までメッセージ単位（フラグメント時は分割単位）に使用します。処理キュー登録数（メモリ関連定義の pce_no オペランドの指定値）やフラグメント発生有無を考慮して設定してください。

UDP 用受信バッファ面数に最小の値を設定する場合は，次に示す計算式の値を指定してください。

処理スレッド分＋（UDP受信スレッド分×6）

（凡例）

処理スレッド分：thread_no オペランドの指定値＋ reserve_thread_no オペランドの指定値＋ 1

UDP 受信スレッド分：myudprcvdef 定義コマンドと clgrpdef 定義コマンドの -p オプションに設定したポート番号数

● udp_send_message_buf_size ＝ UDP **用送信バッファサイズ**

～〈符号なし整数〉((2000 ～ 66000))《33000》**（単位：バイト）**

UDP 用送信バッファ（UOBF）のサイズをバイト単位で指定します。

送信メッセージサイズが送信バッファサイズより大きい場合は，分割（フラグメント）してメッセージを送信します。

参考

UDP 用送信バッファは，UDP プロトコルを使用する通信で使用します。UDP プロトコルを使用する通信を次に示します。

- トランザクション同期の一方送信メッセージの送信
- トランザクション非同期の一方送信メッセージの送信
- タイマトランザクション起動要求のメッセージ（CL サーバで，永続指定の場合だけ）
- CL サーバ間の通信
  - ・定義情報の送信
  - ・CL 同期メッセージの送信（XDB 更新ログ情報を含む）
  - ・CL 同期済み通知メッセージの送信
  - ・実行系ダウン時のリソース回復用メッセージの送信

● udp_send_message_buf_cnt ＝ UDP **用送信バッファ面数**

～〈符号なし整数〉((1 ～ 2000000))《128》

UDP 用送信バッファの面数を指定します。

分割（フラグメント）が発生した場合は，分割した数の送信バッファを使用します。そのため，分割が発生する数を考慮して面数を設定してください。

分割が発生しない場合は，次の計算式で見積もってください。

（システム内で同時に発行する，UDP**プロトコルを使用する通信の要求数**
＋システム内で同時に起動するトランザクションブランチ数）
×1.5**（安全率）**

システム内で同時に起動するトランザクションブランチ数は，RPC 関連定義の rpc_replymsg_save オペランドに Y を指定する場合に必要です。

● memory_xdb_area_size **＝初期化確保時の** XDB **用ワーク領域サイズ**，**追加確保時の** XDB **用ワーク領域サイズ**

～〈符号なし整数〉((1 ～ 1048576))**（単位：キロバイト）**

XDB 用ワーク領域（XDBPOOL）の初期化確保時および追加確保時のサイズをキロバイト単位で指定します。

オペランドの第 1 指定値には，初期化時に確保する XDB 用ワーク領域のサイズを指定します。第 2 指定値には，追加確保時に確保する XDB 用ワーク領域のサイズを指定します。

注意事項

CL サーバ（クラスタ連携関連定義の cluster_mode オペランドに Y を指定）の場合，次に示すオペランドのどちらか一方でも指定していると，このオペランドで指定して追加確保した XDB 用ワーク領域のうち未使用になった領域を解放します。

- memory_mdpsys_area_size オペランド
- memory_mdpusr_area_size オペランド

なお，プールサイズが小さいと，XDB 用ワーク領域が解放されないおそれがあります（OS の仕様）。そのため，memory_xdb_area_size オペランドの第 2 指定値には，MMAP_THRESHOLD の値以上を設定してください。

実際のメモリ要求などで追加確保されるワーク領域のサイズは，指定値と一致しないことがあります。
xdb_use オペランドに Y を指定した場合は，このオペランドは省略できません。必ず，オペランドの第 1 指定値と第 2 指定値の両方を指定してください。
XDB 用ワーク領域を求める計算式については，マニュアル「TP1/EE/Extended Data Cache 使用の手引」を参照してください。

● memory_xdb_limit_size **＝確保する** XDB **用ワーク領域の最大サイズ**

～〈符号なし整数〉((1 ～ 104857600))**（単位：キロバイト）**
memory_xdb_area_size オペランドの指定によって確保する XDB 用ワーク領域の最大サイズを指定します。
memory_xdb_area_size オペランドの第 1 指定値（初期化確保時の XDB 用ワーク領域サイズ）より小さな値を指定した場合，追加確保は行いません。
xdb_use オペランドに Y を指定した場合は，このオペランドは省略できません。必ず指定してください。

● memory_xtc_area_size **＝初期化確保時の** XTC **用ワーク領域サイズ** , **追加確保時の** XTC **用ワーク領域サイズ**

～〈符号なし整数〉((1 ～ 1048576))**（単位：キロバイト）**
XTC 用ワーク領域（XTCPOOL）の初期化確保時および追加確保時のサイズをキロバイト単位で指定します。
オペランドの第 1 指定値には，初期化時に確保する XTC 用ワーク領域のサイズを指定します。第 2 指定値には，追加確保時に確保する XTC 用ワーク領域のサイズを指定します。

注意事項

CL サーバ（クラスタ連携関連定義の cluster_mode オペランドに Y を指定）の場合，次に示すオペランドのどちらか一方でも指定していると，このオペランドで指定して追加確保した XTC 用ワーク領域のうち未使用になった領域を解放します。

- memory_mdpsys_area_size オペランド
- memory_mdpusr_area_size オペランド

なお，プールサイズが小さいと，XTC 用ワーク領域が解放されないおそれがあります（OS の仕様）。そのため，memory_xtc_area_size オペランドの第 2 指定値には，MMAP_THRESHOLD の値以上を設定してください。

実際のメモリ要求などで追加確保されるワーク領域のサイズは，指定値と一致しないことがあります。
XTC 用ワーク領域は，次のオペランドで指定するバッファおよびシステム用バッファの予備領域としても使用します。

- pce_no オペランド
- icb_no オペランド
- time_message_no オペランド
- recv_message_buf_cnt オペランド
- send_message_buf_cnt オペランド
- dbq_recv_message_buf_cnt オペランド
- dbq_send_message_buf_cnt オペランド
- dbq_use_buf_cnt オペランド
- udp_recv_message_buf_cnt オペランド
- udp_send_message_buf_cnt オペランド

このオペランドは省略できません。必ず，オペランドの第 1 指定値と第 2 指定値の両方を指定してください。
XTC 用ワーク領域を求める計算式については，リリースノートを参照してください。

● memory_xtc_limit_size **＝確保する** XTC **用ワーク領域の最大サイズ**

～〈符号なし整数〉((1 ～ 104857600))**（単位：キロバイト）**
memory_xtc_area_size オペランドの指定によって確保する XTC 用ワーク領域の最大サイズを指定します。
memory_xtc_area_size オペランドの第 1 指定値（初期化確保時の XTC 用ワーク領域サイズ）より小さな値を指定した場合，追加確保は行いません。
このオペランドは省略できません。必ず指定してください。

● memory_mdpsys_area_size **＝初期化確保時の大量処理用システム領域サイズ**, **追加確保時の大量処理用システム領域サイズ**

～〈符号なし整数〉((1 ～ 1048576))**（単位：キロバイト）**
大量処理用システム領域（MPSPOOL）を使用する場合，初期化確保時および追加

確保時のサイズをキロバイト単位で指定します。
オペランドの第 1 指定値には，初期化時に確保する大量処理用システム領域のサイズを指定します。第 2 指定値には，追加確保時に確保する大量処理用システム領域のサイズを指定します。このオペランドを指定する場合は，第 1 指定値と第 2 指定値の両方を指定してください。
このオペランドを指定すると，XTC ワーク領域および XDB ワーク領域の動作が次のとおり変更になります。

- 追加確保した領域については，使用中のセグメントが存在しなくなった時点で，オンライン中に該当する領域を OS に解放（free 関数を発行）します。
- CL サーバ（クラスタ連携関連定義の cluster_mode オペランドに Y を指定）の場合，待機系では memory_xtc_limit_size オペランドおよび memory_xdb_limit_size オペランドの設定値が無効になります。なお，設定値を超えるセグメント要求があった場合，設定値を無視して領域を追加確保（malloc 関数を発行）します。

! 注意事項

プールサイズが小さいと，大量処理用システム領域が解放されないおそれがあります（OS の仕様）。第 1 指定値と第 2 指定値には MMAP_THRESHOLD の値以上を設定してください。

実際のメモリ要求などで追加確保される大量処理用システム領域のサイズは，指定値と一致しないことがあります。
このオペランドを省略すると，大量処理用システム領域を使用しません。

● memory_mdpusr_area_size ＝初期化確保時の大量処理用ユーザ領域サイズ , 追加確保時の大量処理用ユーザ領域サイズ

～〈符号なし整数〉((1 ～ 1048576))（単位：キロバイト）
大量処理用ユーザ領域（MPUPOOL）を使用する場合，初期化確保時および追加確保時のサイズをキロバイト単位で指定します。
オペランドの第 1 指定値には，初期化時に確保する大量処理用ユーザ領域のサイズを指定します。第 2 指定値には，追加確保時に確保する大量処理用ユーザ領域のサイズを指定します。このオペランドを指定する場合は，第 1 指定値と第 2 指定値の両方を指定してください。
このオペランドを指定すると，XTC ワーク領域および XDB ワーク領域の動作が次のとおり変更になります。

- 追加確保した領域については，使用中のセグメントが存在しなくなった時点で，オンライン中に該当する領域を OS に解放（free 関数を発行）します。
- CL サーバ（クラスタ連携関連定義の cluster_mode オペランドに Y を指定）の場合，待機系では memory_xtc_limit_size オペランドおよび memory_xdb_limit_size オペランドの設定値が無効になります。なお，設定値を超えるセグメント要求があった場合，設定値を無視して領域を追加確保（malloc

関数を発行）します。

注意事項

プールサイズが小さいと，大量処理用ユーザ領域が解放されないおそれがあります（OS の仕様）。第 1 指定値と第 2 指定値には MMAP_THRESHOLD の値以上を設定してください。

実際のメモリ要求などで追加確保される大量処理用ユーザ領域のサイズは，指定値と一致しないことがあります。
このオペランドを省略すると，大量処理用ユーザ領域を使用しません。

● mem_ibf_extend_num **＝受信バッファの最大拡張数**

～〈符号なし整数〉((0 ～ 2147483647))《2147483647》
受信バッファ（IBF）が不足した場合に拡張する面数の上限を指定します。
このオペランドを指定した場合に利用できるバッファの最大数を次に示します。

recv_message_buf_cnt**オペランドで指定した初期確保数**
＋mem_ibf_extend_num**の指定値**

ただし，この値が 2147483647 を超えても，バッファの最大数の上限は 2147483647 です。0 を指定した場合はバッファを拡張しません。
なお，このオペランドで指定した面数を超えなくても，XTC 用ワーク領域の最大サイズを超えたときは拡張できません。

● mem_uibf_extend_num ＝ UDP **用受信バッファの最大拡張数**

～〈符号なし整数〉((0 ～ 2147483647))《2147483647》
UDP 用受信バッファ（UIBF）が不足した場合に拡張する面数の上限を指定します。
このオペランドを指定した場合に利用できるバッファの最大数を次に示します。

udp_recv_message_buf_cnt**オペランドで指定した初期確保数**
＋mem_uibf_extend_num**の指定値**

ただし，この値が 2147483647 を超えても，バッファの最大数の上限は 2147483647 です。0 を指定した場合はバッファを拡張しません。
なお，このオペランドで指定した面数を超えなくても，XTC 用ワーク領域の最大サイズを超えたときは拡張できません。

● mem_obf_extend_num **＝送信バッファの最大拡張数**

～〈符号なし整数〉((0 ～ 2147483647))《2147483647》
送信バッファ（OBF）が不足した場合に拡張する面数の上限を指定します。
このオペランドを指定した場合に利用できるバッファの最大数を次に示します。

send_message_buf_cnt**オペランドで指定した初期確保数**
＋mem_obf_extend_num**の指定値**

ただし，この値が 2147483647 を超えても，バッファの最大数の上限は 2147483647 です。0 を指定した場合はバッファを拡張しません。
なお，このオペランドで指定した面数を超えなくても，XTC 用ワーク領域の最大サイズを超えたときは拡張できません。

● mem_uobf_extend_num ＝ UDP **用送信バッファの最大拡張数**

〜〈符号なし整数〉((0 〜 2147483647))《2147483647》
UDP 用送信バッファ（UOBF）が不足した場合に拡張する面数の上限を指定します。
このオペランドを指定した場合に利用できるバッファの最大数を次に示します。

udp_send_message_buf_cnt**オペランドで指定した初期確保数**
＋mem_uobf_extend_num**の指定値**

ただし，この値が 2147483647 を超えても，バッファの最大数の上限は 2147483647 です。0 を指定した場合はバッファを拡張しません。
なお，このオペランドで指定した面数を超えなくても，XTC 用ワーク領域の最大サイズを超えたときは拡張できません。

RPC 関連定義

ここでは，RPC 関連定義の各オペランドについて説明します。

形式

〔set rpc_udp_packet_size＝パケットサイズ〕
〔set rpc_udp_congestion_packet_count＝パケット数〕
〔set rpc_udp_departure_limit＝UDP通信を行うスレッド数の合計〕
〔set rpc_udp_linetrace＝回線トレース情報の取得量の変更〕
〔set rpc_tcp_communication_use＝｛Y｜N｝〕
〔set rpc_udp_lmsg_deliverychk_size＝複数メッセージの一括送受信機能を使用した場合の送達確認サイズ〕
〔set rpc_udp_msg_deliverychk_size＝送達確認サイズ〕
〔set rpc_recv_permanence＝｛Y｜N｝〕

オペランドの説明

● rpc_udp_packet_size **＝パケットサイズ**

～〈符号なし整数〉((1452 ～ 16000))《1472》**（単位：バイト）**

UDP 通信機能の通信単位であるパケットのサイズをバイト単位で指定します。このオペランドの指定値は，UDP 通信を行う XTC 間で同じ値を指定します。一致しない場合は，通信に失敗します。

このオペランドの指定値には，次に示す値を指定することお勧めします。次に示す値以外を指定した場合は，IP 層でのフラグメント化の多発によるパケット数増加によって，性能が低下するおそれがあります。

MTU**サイズ**－28

UDP 通信を行う XTC 間で MTU サイズが異なる場合は，最も小さい MTU サイズに合わせます。MTU サイズの取得方法は OS によって異なります。詳細については，OS のマニュアルを参照してください。

なお，ジャンボフレームを使用できる環境では，ジャンボフレームの使用をお勧めします。ジャンボフレームを使用した場合，パケットサイズの拡張によって通信量が減少し，通信性能が向上します。

ただし，hbonding ドライバによって W-send 機能を使用する場合は，hbonding ドライバの仕様上，推奨値を超える値は指定しないでください。指定した場合の動作は保証しません。

なお，UDP 受信バッファ（UIBF）のサイズを超える値を指定した場合，XTC の起動に失敗します。

● rpc_udp_congestion_packet_count **＝パケット数**

～〈符号なし整数〉((0，100 ～ 16000))《1000》

輻輳制御の開始，終了の判断に使用するための監視対象パケット数を指定します。このオペランドに 0 を指定した場合は監視しません。
パケット送信時には，監視対象パケット数分の送信履歴を残します。再送パケット比率は，監視対象パケット数と再送パケット数を基に算出します。この再送パケット比率と，パケットロス上限およびパケットロス下限を比較することで，輻輳制御の開始，終了を判断します。パケットロス上限およびパケットロス下限については，UDP グループ情報関連定義の myudpsnddef 定義コマンドで指定します。
監視対象パケット数が少ない場合は，輻輳の検出が速くなります。しかし，瞬間的なパケットロスを輻輳として判断する場合があります。また，監視対象パケット数が多い場合は輻輳の検出が遅くなります。
輻輳制御の詳細については，「2.4.2　輻輳制御」を参照してください。

● rpc_udp_departure_limit ＝ UDP 通信を行うスレッド数の合計

～〈符号なし整数〉((100 ～ 10000))《200》
自 XTC と UDP 通信を行う，すべての相手プロセスのスレッド数の合計値を指定します。
自プロセスに対して送信を行う場合は，自プロセスのスレッド数も加算する必要があります。ただし，ee_mch_cmtsend 関数，または ee_mch_cmtsend_sync 関数のメッセージ送信形態に EEMCH_MYPROC を指定して送信する場合は，自プロセスのスレッド数を加算する必要はありません。
すべての相手プロセスのスレッド数が，このオペランドの指定値を超えた場合は，通信が失敗します。

● rpc_udp_linetrace ＝回線トレース情報の取得量の変更

～〈16 進数〉((00000000 ～ 00000007 ｜ 00000020 ～ 00000023))《00000000》
UDP 通信機能の回線トレース情報の取得量を変更する場合，変更する項目を論理和で指定します。指定値の組み合わせについては，「6.3.1(1)(c)　rpc_udp_linetrace オペランドの指定値の組み合わせ」を参照してください。

00000000：
すべての回線トレース情報を取得します。

00000001：
W-send 機能や再送などによって重複して受信したパケットについて，回線トレース情報を取得しません。

00000002：
次のすべての条件を満たす場合に，回線トレース情報を取得しません。
- myudpsnddef 定義コマンドの -k オプションを指定している
- クラスタ内でのマルチキャストによって自 TP1/EE が送信したメッセージを，自 TP1/EE で受信した

00000004：
メッセージが複数のパケットに分割されている場合，先頭パケットは UDP 通

信制御ヘッダおよびデータ部分を最大 374 バイト取得します。残りのパケットは UDP 通信制御ヘッダだけを取得します。

00000020：

送受信が正常に行われた場合は，送受信データの先頭にある製品ヘッダを回線トレース情報として取得します。送受信データが複数のパケットに分割されている場合は，先頭パケットだけが製品ヘッダを取得します。残りのパケットは，製品ヘッダのうち UDP 通信制御ヘッダだけを取得します。

● rpc_tcp_communication_use ＝ {Y ｜ N}

～《Y》

RPC 通信で，TCP/IP プロトコルを使用するかどうかを指定します。

Y：

RPC 通信で，TCP/IP プロトコルを使用します。

N：

RPC 通信で，TCP/IP プロトコルを使用しません。

次の条件をすべて満たしている場合にだけ，このオペランドに N を指定できます。

- xtc_use ＝ Y
- name_use ＝ N
- trn_transactional_rpcless_use ＝ Y

このオペランドに N を指定した場合でも，ee_rpc_call 関数と ee_rpc_cmtsend 関数の自プロセス送信（通信レス）による RPC 要求はできます。ただし，RPC 要求先のサービスで，RPC 応答のトランザクションまたがり送信機能，および ERRTRNR による RPC 応答メッセージ送信機能を使用することはできません。

このオペランドに N を指定した場合は，次の表に示す定義コマンドおよびオペランドの省略時解釈値または指定値が変更されます。

サービスグループ情報関連定義の mysvgdef 定義コマンドおよび myreplydef 定義コマンドは省略できません。しかし，このオペランドに N を指定した場合は省略できます。

表 5-7　rpc_tcp_communication_use オペランドに N を指定した場合の省略時解釈値または指定値の変化

定義コマンド名，またはオペランド名	Y 指定時の省略時解釈値	N 指定時の省略時解釈値	指定値
mysvgdef	なし	なし	無視されます。
myreplydef	なし	なし	
rpc_reply_con_cnt	2	0	
rpc_reply_con_max_cnt	256	0	
rpc_reply_proc_max_cnt	128	0	

定義コマンド名，またはオペランド名	Y 指定時の省略時解釈値	N 指定時の省略時解釈値	指定値
rpc_reply_suspend_cnt	128	0	
rpc_reply_suspend_time	180	0	
rpc_reply_suspend_recover	Y	N	
rpc_reply_suspend_autosend	N	N	
rpc_reply_errtrnr	N	N	
rpc_reply_errtrnr_cnt	128	0	
rpc_reply_port_auto	N	N	

● rpc_udp_lmsg_deliverychk_size **＝複数メッセージの一括送受信機能を使用した場合の送達確認サイズ**

～〈符号なし整数〉((1452 ～ 9437184))《524288》**（単位：バイト）**

複数メッセージの一括送受信機能を使用する場合に，送達確認を行うサイズをバイト単位で指定します。

複数メッセージの一括送受信機能については，「2.4.3　複数メッセージの一括送受信機能」を参照してください。

このオペランドの指定値が小さい場合，送達確認が多発し，通信遅延の原因になります。このオペランドの指定値が大きい場合，パケットロスなどによる通信障害時に，再送による通信量が増加します。そのため，パケットロス頻度が高い（ネットワーク性能やマシン性能が低い）構成では，このオペランドの指定値を小さくし，パケットロス頻度の低い（ネットワーク性能やマシン性能が高い）構成では，このオペランドの指定値を大きくすることをお勧めします。

複数メッセージの一括送受信は，次の場合に行います。

- CL サーバで CL 同期メッセージの送受信をする場合
- 高速メッセージ送信関連定義の mch_send_lump_count オペランドに 2 以上を指定し，一方送信メッセージを送受信する場合

なお，CL 同期メッセージの送受信だけで使用するソケットの場合に，ユーザデータのサイズがこのオペランドの指定値以上になるときは，このオペランドの指定値はユーザデータのサイズになります。

● rpc_udp_msg_deliverychk_size **＝送達確認サイズ**

～〈符号なし整数〉((1452 ～ 9437184))《2097152》**（単位：バイト）**

メッセージ送受信時の，送達確認を行うサイズをバイト単位で指定します。

このオペランドの指定値が小さい場合，送達確認が多発し，通信遅延の原因になります。このオペランドの指定値が大きい場合，パケットロスなどによる通信障害時に，再送による通信量が増加します。そのため，パケットロス頻度が高い（ネットワーク性能やマシン性能が低い）構成では，このオペランドの指定値を小さくし，パケットロス頻度の低い（ネットワーク性能やマシン性能が高い）構成では，この

オペランドの指定値を大きくすることをお勧めします。
なお，複数メッセージの一括送受信時には，rpc_udp_lmsg_deliverychk_size オペランドで指定した値が使用されます。また，CL 同期メッセージ以外の送受信に使用するソケットの場合に，ユーザデータのサイズがこのオペランドの指定値以上になるときは，このオペランド指定値はユーザデータのサイズになります。

● rpc_recv_permanence ＝ {Y ｜ N}

～《N》
クラスタを構成する実行系と待機系の間で，RPC 要求の受信メッセージを永続化するかどうかを指定します。
このオペランドは，クラスタ連携関連定義の cluster_mode オペランドに Y を指定している場合だけ有効です。

Y：
RPC 要求の受信メッセージを永続化します。

N：
RPC 要求の受信メッセージを永続化しません。

ユーザサービス関連定義

ここでは，ユーザサービス関連定義の各オペランドについて説明します。

なお，ユーザサービス関連定義のオペランドには，HA サーバで指定できないオペランドがあります。HA サーバで指定できないオペランドについては，「HA サーバで指定できない XTC サービス定義」を参照してください。

形式

```
〔set event_trn＝サービス名〕
〔set system_start_ui＝実行系連絡トランザクションを起動するエントリポインタ名〕
〔set scd_rollback_retry_mode＝｛H｜P｝〕
〔set errtrn_serialize＝｛Y｜N｝〕
```

オペランドの説明

● event_trn ＝サービス名

～〈1 ～ 31 文字の識別子〉

イベント通知トランザクション（MV）を起動するサービス名を指定します。ユーザサービス関連定義の service オペランドで指定したサービス名を指定してください。

イベント通知トランザクションは service オペランドに指定したサービスに登録しますが，該当するサービスが閉塞中であっても service オペランドに指定したエントリポインタを起動します。イベント通知トランザクションのエラートランザクションは起動しません。なお，サービス閉塞による処理キュー引き出し禁止となっている場合（ユーザサービス関連定義の forbid_draw_service オペランドに Y を指定している場合）は，引き出し禁止が解除されるまでイベント通知トランザクションを起動しません。

なお，このオペランドを省略した場合は，イベント通知トランザクションを起動しません。

● system_start_ui ＝実行系連絡トランザクションを起動するエントリポインタ名

～〈1 ～ 31 文字の識別子〉

実行系連絡トランザクション（UI）を起動するエントリポインタ名を指定します。このオペランドで指定したエントリポインタ名は，ほかのオペランドで指定したエントリポインタ名と同じものでもかまいません。ユーザサービス関連定義の service オペランドで指定したエントリポインタ名を指定してください。

なお，このオペランドを省略した場合は，実行系連絡トランザクションを起動しません。

● scd_rollback_retry_mode ＝ ｛H ｜ P｝

～《H》

ユーザサービス関連定義の service_attr 定義コマンドの -f オプションに指定した，連続ロールバック監視回数を超えた場合の動作を指定します。

H：

サービス閉塞します。

P：

プロセスダウンします。

● errtrn_serialize ＝ {Y ｜ N}

～《Y》

ERRTRNR，ERRTRN3 を起動する障害発生時の端末キュー，処理キューから次のメッセージを引き出す契機を，ERRTRN3/ERRTRNR 終了後か，ERRTRN3/ERRTRNR の終了とは非同期かを指定します。

Y：

ERRTRNR，ERRTRN3 が終了してから端末キュー，処理キューから次のメッセージを引き出します。

N：

ERRTRNR，ERRTRN3 の終了とは非同期に，障害が発生したトランザクション決着時に端末キュー，処理キューから次のメッセージを引き出します。

このオペランドには，省略，または Y を指定することを推奨します。

このオペランドを指定できるのは，TP1/EE のバージョンが次の場合です。

- 07-7x の場合，07-73 以降
- 07-8x の場合，07-81 以降

TP1/EE のバージョンがこのオペランドを指定できないバージョンの場合は，このオペランドに N を指定した場合の動作をします。

ユーザサービス関連定義（コマンド形式）

ここでは，ユーザサービス関連定義（コマンド形式）の各定義コマンドについて説明します。

service_attr（サービス属性定義）

TP1/EE の定義コマンドです。

ここでは，TP1 キャッシュ機能使用時に使用できるオプションを示します。そのほかのオプションについては，マニュアル「TP1/Server Base Enterprise Option 使用の手引」を参照してください。

機能

set 形式のユーザサービス関連定義で定義した各サービスの付加情報を定義します。

TP1 キャッシュ機能使用時の各オプションに対応した機能の詳細は，「2.6　処理キューの制御」を参照してください。

コマンドの形式

```
{ {service_attr 〔-e {parallel｜serial}〕
                〔-f 連続ロールバックリトライ監視回数〕
                〔-g {Y｜N}〕
} }
```

オプションの説明

● -e {parallel ｜ serial}

～《parallel》

メッセージの同時引き出しの可否を指定します。

parallel：

同時引き出しできます。

serial：

同時引き出しできません。

serial を指定した場合は，ユーザサービス関連定義の service オペランドの同時処理限界数に 1 を指定するか，または同時処理限界数を指定しないでください。

-e オプションは，-v オプションを指定した場合に指定できます。-x オプションを指定した場合は指定できません。

● -f **連続ロールバックリトライ監視回数**

～〈符号なし整数〉((0 ～ 10))**《5》**

サービスの同時引き出しができないサービスで，同じメッセージを読み出すトランザクションがロールバックリトライする回数の上限を指定します。監視回数を超えてロールバックリトライした場合，ユーザサービス関連定義の

scd_rollback_retry_mode オペランドの指定に従って処理します。0 を指定した場合は，ロールバックリトライ回数を監視しません。
-f オプションは，-e オプションに serial を指定した場合にだけ指定できます。

● -g {Y | N}

～《Y》

入力キュー（ITQ）の連鎖モード連携機能を使用するかを指定します。

Y：

連鎖モード連携機能を使用します。
ee_trn_chained_commit 関数，または ee_trn_chained_rollback 関数（thkind に EETRN_KEEP を設定）による同期点取得時に読み出したメッセージを読み出し済みにするか，またはロールバックリトライします。これらの関数が実行されなかった場合は，サービス関数実行終了時にメッセージを読み出し済みにするか，またはロールバックリトライします。

N：

連鎖モード連携機能を使用しません。
サービス関数実行終了時にメッセージを読み出し済みにするか，またはロールバックリトライします。

-g オプションは，-v オプションを指定した場合に指定できます。-x オプションを指定した場合は指定できません。

uoc_func（ユーザオウンコーディング定義）

機能

ユーザオウンコーディングのエントリポインタ名を指定します。

コマンドの形式

```
〔uoc_func -s オンライン開始UOCエントリポインタ名〕
```

オプションの説明

● -s **オンライン開始** UOC **エントリポインタ名**

～〈1～31 文字の識別子〉

オンライン開始 UOC のエントリポインタ名を指定します。
省略した場合，オンライン開始 UOC を呼び出しません。

トラブルシュート関連定義

ここでは，トラブルシュート関連定義の各オペランドについて説明します。

なお，トラブルシュート関連定義のオペランドには，HA サーバで指定できないオペランドがあります。HA サーバで指定できないオペランドについては，「HA サーバで指定できない XTC サービス定義」を参照してください。

形式

〔set trb_quedump_wtor_interval_time＝eetrbwtorコマンド実行待ちメッセージ出力間隔〕
〔set trb_quedump_io_watch_time＝キューダンプファイルのI/O処理監視時間〕
〔set trb_quedump_io_watch_msg_time＝キューダンプファイルのI/O処理監視メッセージ出力最小間隔〕
〔set trb_pcerefer_output_time＝eepcerefercコマンドのファイル出力監視時間〕

オペランドの説明

● trb_quedump_wtor_interval_time ＝ eetrbwtor **コマンド実行待ちメッセージ出力間隔**

～〈符号なし整数〉((0 ～ 60))**《5》（単位：分）**
キューダンプファイルの出力障害によって eetrbwtor コマンドの実行待ちとなっている間に，KFSB85802-Q メッセージを出力する間隔を分単位で指定します。
0 を指定した場合，キューダンプファイルの出力障害発生時に一度だけ KFSB85802-Q メッセージを出力します。

● trb_quedump_io_watch_time **＝キューダンプファイルの** I/O **処理監視時間**

～〈符号なし整数〉((0 ～ 3600))**《10》（単位：秒）**
キューダンプファイルの I/O 処理 1 回当たりの監視時間を秒単位で指定します。
0 以外を指定した場合は，キューダンプファイルの I/O 処理 1 回当たりの時間が指定値を超えたときに KFSB55412-E メッセージを出力します。トランザクションタイマの監視時間には，キューダンプファイルの I/O 処理時間を含みません。
0 を指定した場合は，キューダンプファイルの I/O 処理を監視しません。

● trb_quedump_io_watch_msg_time **＝キューダンプファイルの** I/O **処理監視メッセージ出力最小間隔**

～〈符号なし整数〉((10 ～ 36000))**《**trb_quedump_io_watch_time × 10**》（単位：秒）**
I/O 処理 1 回当たりの時間が trb_quedump_io_watch_time オペランドの指定値を超えた場合に，KFSB55412-E メッセージを出力する最小間隔を秒単位で指定します。
最後に KFSB55412-E メッセージを出力した時刻からこのオペランドの指定値を経過していない場合は，KFSB55412-E メッセージを出力しません。

● trb_pcerefer_output_time ＝ eepcerefer **コマンドのファイル出力監視時間**

～〈符号なし整数〉((0 ～ 65535))**《60》（単位：秒）**

eepcerefer コマンドの出力処理の監視時間を秒単位で指定します。指定時間を超えた場合は，TP1/EE プロセスは異常終了します。0 を指定した場合は監視しません。

トランザクション関連定義

ここでは，トランザクション関連定義の各オペランドについて説明します。

なお，トランザクション関連定義のオペランドには，HA サーバで指定できないオペランドがあります。HA サーバで指定できないオペランドについては，「HA サーバで指定できない XTC サービス定義」を参照してください。

形式

〔set trn_transactional_rpcless_use＝｛Y｜N｝〕
〔set trn_expiration_time_ui＝実行系連絡トランザクション（UI）処理監視時間〕
〔set trn_expiration_time_mv＝イベント通知トランザクション（MV）処理監視時間〕
〔set trn_cl_rpc_reply_timing＝｛Y｜N｝〕

オペランドの説明

● trn_transactional_rpcless_use ＝ {Y ｜ N}

～《N》
トランザクショナル RPC の実行を抑止するかどうかを指定します。トランザクショナル RPC の抑止については，「2.5.3　トランザクショナル RPC の抑止（CL サーバ）」を参照してください。

Y：
トランザクショナル RPC の実行を抑止します。

N：
トランザクショナル RPC の実行を抑止しません。

● trn_expiration_time_ui **＝実行系連絡トランザクション（UI）処理監視時間**

～〈符号なし整数〉((0 ～ 65535))《trn_expiration_time **オペランドの指定値》（単位：秒）**
実行系連絡トランザクション（UI）でのトランザクション処理の監視時間を秒単位で指定します。
監視時間が指定値を超えた場合は，プロセス関連定義の uapabend_downmode オペランドの指定に従って異常終了します。
このオペランドに 0 を指定した場合は，時間監視しません。
このオペランドを省略した場合は，トランザクション関連定義の trn_expiration_time オペランドの指定値を省略時解釈値とします。

● trn_expiration_time_mv **＝イベント通知トランザクション（MV）処理監視時間**

～〈符号なし整数〉((0 ～ 65535))《trn_expiration_time **オペランドの指定値》（単位：秒）**

イベント通知トランザクション（MV）でのトランザクション処理の監視時間を秒単位で指定します。
監視時間が指定値を超えた場合は，プロセス関連定義の uapabend_downmode オペランドの指定に従って異常終了します。
このオペランドに 0 を指定した場合は，時間監視しません。
このオペランドを省略した場合は，トランザクション関連定義の trn_expiration_time オペランドの指定値を省略時解釈値とします。

● trn_cl_rpc_reply_timing ＝ {Y ｜ N}

～《N》

TP1/EE/XTC の CL サーバで，RPC 応答の送信タイミング変更機能を使用するかどうかを指定します。
この機能を使用した場合，RPC のレスポンスタイムが落ちます。RPC 応答メッセージに転送処理の結果を反映したいときだけ使用してください。
このオペランドは，次の条件を満たした場合だけ有効です。

- cluster_mode オペランドに Y を指定している
- clgrpdef 定義コマンドでクラスタグループ内のノード情報を 2 個以上指定している
- cluster_test_mode オペランドに Y を指定していない

Y：

RPC 応答の送信タイミング変更機能を使用します。
RPC 応答の送信は，転送処理によってリソースの永続化が完了したあとに行います。
転送処理によるリソースの永続化に失敗したときは，RPC エラー応答を送信します。

N：

RPC 応答の送信タイミング変更機能を使用しません。

ステータスファイル関連定義

ここでは，ステータスファイル関連定義の各オペランドについて説明します。

形式

```
〔set sts_fileless_use＝ {Y｜N} 〕
〔set sts_fileless_level＝ {1｜2} 〕
```

オペランドの説明

● sts_fileless_use ＝ {Y ｜ N}

～《N》

ステータスファイルレス機能を使用するかどうかを指定します。ステータスファイルレス機能については，「2.7　ステータスファイルレス機能」を参照してください。

Y：

ステータスファイルレス機能を使用します。

N：

ステータスファイルレス機能を使用しません。

XA インタフェースによるトランザクションを実行する場合，またはトランザクショナル RPC を使用する場合，Y は指定できません。

● sts_fileless_level ＝ {1 ｜ 2}

～《1》

ステータスファイルレス機能使用時，システム制御情報の取得レベルを指定します。sts_fileless_level オペランドは，sts_fileless_use オペランドに Y を指定している場合にだけ有効です。

1：

全体レスとなります（すべてのシステム制御情報を取得しません）。CL サーバの場合に，必ず指定してください。

1 を指定した場合，ステータスファイル自体を使用しません。ほかのステータスファイル関連定義を指定しても無効になります。

2：

部分レスとなります（トランザクションに関連するシステム制御情報を取得しません）。HA サーバの場合に，任意で指定してください。

2 を指定した場合，ステータスファイルは使用します。ステータスファイル関連定義の必須定義は指定してください。

サービスグループ情報関連定義（コマンド形式）

ここでは，サービスグループ情報関連定義（コマンド形式）の各定義コマンドについて説明します。

eesvgdef（相手サービスグループ情報定義）

機能

通信相手となるサービスグループの情報，および通信方法を指定します。

コマンドの形式

通信方法によって，指定できるオプションが異なります。通信方法の指定を次に示します。

通信方法が RPC，DBQ，または RAP の場合

```
{ {eesvgdef -g サービスグループ名
            -h ホスト名:ポート番号〔,ホスト名:ポート番号〕…
           〔-s ソケット数〕
           〔-a〕
           〔-t  {RPC｜DBQ｜RAP}〕
} }
```

通信方法が UDP の場合

```
{ {eesvgdef -g サービスグループ名
            -u あて先UDPグループ名
            -t UDP
} }
```

オプションの説明

● -g サービスグループ名

〜〈1 〜 31 文字の識別子〉

サービス要求を行うサービスグループ名を指定します。

● -h ホスト名：ポート番号

-t オプションに UDP 以外を指定した場合，サービス要求先のホスト名およびポート番号を指定します。

- ホスト名

  〜〈1 〜 255 文字のホスト名〉

  サービス要求を行うホスト名を指定します。-t オプションに RAP を指定した場合は，リモート API 機能によるサービスの受信口となるホスト名を指定してください。

  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。

- **ポート番号**
  ～〈符号なし整数〉((1 ～ 65535))
  サービス要求を行うホストのポート番号を指定します。相手ホストが TP1/Server Base の場合は，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。-t オプションに RAP を指定した場合は，リモート API 機能によるサービスの受信口となるポート番号を指定してください。

TP1/EE から TP1/Server Base 管理下の SPP のサービスを利用する場合，その SPP のサービスグループ名とその SPP が存在するノードのホスト名，およびスケジュールサービス定義の scd_port オペランドに指定したポート番号を指定します。リモート API 機能を使用して通信する場合（-t オプションに RAP を指定する場合）は，リモート API 機能によるサービスの受信口のホスト名とポート番号を指定します。
TP1/EE から TP1/EE のサービスを利用する場合，自サービスグループ情報定義の mysvgdef 定義コマンドで指定したホスト名とポート番号を指定します。
-h オプションには，複数のホスト名およびポート番号を指定することもできます。ただし，リモート API 機能を使用して通信する場合（-t オプションに RAP を指定する場合）は，複数のホスト名およびポート番号は指定できません。

● -u **あて先** UDP **グループ名**
～〈1 ～ 31 文字の識別子〉
-t オプションに UDP を指定した場合に，サービス要求先のあて先 UDP グループ名を指定します。
UDP プロトコルを使用して通信する場合（-t オプションに UDP を指定する場合）は，サービス要求先を定義している eeudpdef 定義コマンドのあて先 UDP グループ名を指定します。

● -s **ソケット数**
～〈符号なし整数〉((1 ～ 32))《1》
接続する最大ソケット数をホスト単位に指定します。複数スレッドから同一サービスグループ名で送信要求を行う場合，サービスグループ下のソケット数の合計は，処理スレッド数＋ 1 より大きくなるように指定することをお勧めします。
TP1/Server Base に対するサービス要求を行う場合は，TP1/Server Base のスケジュールサービス定義の max_socket_descriptors オペランドで指定するソケット数を追加してください。
ネームサービス機能を使用して，-t オプションに RPC を指定するか，または指定を省略した場合，このオプションは無視されます。
また，-t オプションに RAP または UDP を指定した場合も，このオプションは無視されます。

● -a
TP1/EE プロセス起動時に，相手ホストとコネクションを自動確立するときに指定します。相手ホストが TP1/Server Base 管理下の SPP の場合，または相手ホストがコネクションを確立してから最初のメッセージを受信するまで時間監視している場

合は，このオプションを指定しないでください。ネームサービス機能を使用して，-t オプションに RPC を指定するか，または指定を省略した場合，このオプションは無視されます。
また，-t オプションに RAP または UDP を指定した場合も，このオプションは無視されます。

● -t {RPC | DBQ | RAP | UDP}

〜《RPC》

該当するホストで使用する通信方法を指定します。

RPC：

RPC 通信を使用します。DB キュー機能で使用するホストの場合，DB キューを使用したシステム間通信を RPC 通信によって行います。また，DB キュー機能で使用するホストの場合，RPC 関連定義の name_use オペランドの指定に関係なく，ネームサービス機能は使用しません。

DBQ：

DB キューを使用したシステム間通信を TCP/IP プロトコルによって行います。DBQ を指定する場合は，UDP グループ情報関連定義の mysvgdef 定義コマンドの -t オプションで DBQ を指定して，DB キューのイベント通知用のポートとホスト名の組み合わせを最低一つ指定する必要があります。ただし，接続先が TP1/EE 01-01 の場合は，-t オプションを設定しないでください。

RAP：

リモート API 機能を使用して通信します。このオプションを指定すると，-h オプションで指定した値は，リモート API 機能によるサービスの受信口の情報となります。

UDP：

高速メッセージ送信制御によって通信します。次の機能を使用して通信します。

- トランザクション同期の一方送信要求（ee_mch_cmtsend_sync 関数）
- トランザクション非同期の一方送信要求（ee_mch_cmtsend 関数）
- UDP プロトコルによる RPC 通信（ee_rpc_call 関数（EERPC_UDP 指定））

注意事項

- 複数の eesvgdef 定義コマンドに同じサービスグループ名を指定し，かつ，それぞれの -t オプションに RAP と RPC（または UDP）を指定していた場合は，定義ファイル中で先に記述されている eesvgdef 定義コマンドの指定が有効になります。定義コマンドの指定内容および指定順に注意してください。
- 複数の eesvgdef 定義コマンドに同じサービスグループ名を指定し，それぞれの -t オプションに RPC と UDP を指定できます。ee_rpc_call 関数発行時の flags 引数として EERPC_UDP を指定しない場合は RPC の eesvgdef 定義コマンド，指定した場合は UDP の eesvgdef 定義コマンドが使用されます。
- TP1/EE は，UAP から呼び出される ee_rpc_call 関数，ee_rpc_cmtsend 関数につい

て，第 1 引数に指定されたサービスグループ名を，この定義に指定されたサービスグループ名の中から検索します。サービスグループ名が一致する定義が見つかった場合，その定義に指定されているホストとポート番号へサービス要求を送信します。

- 同一サービスグループ下に複数のホスト名を指定した場合，TP1/EE はランダムにホストを選択してサービス要求を送信します。ネームサービス機能を使用した場合は，前回送信したホストを選択してサービス要求を送信し，送信に失敗した場合は残りのホストから順次選択します。該当するホストでサービス要求できるコネクションがない場合は，残りのホストから順次選択します。すべてのホストでサービス要求ができない場合，ee_rpc_call 関数はエラーとなります。ee_rpc_call 関数または ee_rpc_cmtsend 関数によって自プロセスあてにサービス要求を行う場合は，自ホストの情報を定義する必要があります。
- リモート API 機能を使用して通信するサービスグループに対して，トランザクションとしてサービスを要求しても，非トランザクションモードとして処理します。
- リモート API 機能を使用して通信する場合は，-h オプションに自プロセスのホスト名とポート番号（RPC 関連定義の rap_listen_port オペランドの指定値）は指定しないでください。指定してサービス要求した場合の動作は保証しません。

高速メッセージ送信関連定義

ここでは，高速メッセージ送信関連定義の各オペランドについて説明します。

形式

〔set mch_send_ack_timer＝送達確認待ち監視時間〕
〔set mch_send_retry_count＝メッセージ送信エラー時の送信リトライ回数〕
〔set mch_send_retry_interval＝送信リトライ開始間隔T1，送信リトライ最大間隔T2〕
〔set mch_send_rs_retry_interval＝送信リトライ開始間隔T1，送信リトライ最大間隔T2〕
〔set mch_send_max_count＝メッセージ送信最大数〕
〔set mch_send_ack_retry_count＝送達確認メッセージ送信エラー時の送信リトライ回数〕
〔set mch_send_ack_retry_interval＝送達確認メッセージの送信リトライ間隔〕
〔set mch_send_otqwatch_time_interval＝出力キュー閉塞監視インターバル時間〕
〔set mch_send_end_otqwatch＝｛continue｜dct_continue｜timer｝〕
〔set mch_send_end_otqwatch_time＝終了時の未送信メッセージ監視時間〕
〔set mch_receive_limit＝一方送信メッセージの送信元合計数〕
〔set mch_send_otqschedule_interval＝一方送信メッセージの再送信スケジュール間隔〕
〔set mch_send_lump_count＝一括送信するメッセージ数〕

オペランドの説明

● mch_send_ack_timer **＝送達確認待ち監視時間**

～〈符号なし整数〉((1 ～ 65535))**《1》（単位：10 ミリ秒）**

ee_mch_cmtsend 関数または ee_mch_cmtsend_sync 関数による一方送信メッセージを送信してから，送達確認メッセージを受信す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ない場合は，高速メッセージ送信関連定義の mch_send_retry_count オペランドに従って再送します。

永続指定の一方送信メッセージの送信先に CL サーバが含まれる場合，送信先の CL サーバの実行系では，一方送信メッセージと待機系からの送達確認メッセージの待ち合わせを行います。このとき，最大で，クラスタ連携関連定義の mch_clsend_ack_timer オペランドで指定した時間を待ち合わせします。

そのため，永続指定の一方送信メッセージの送信先に CL サーバが含まれる場合，このオペランドに，送信先 CL サーバの mch_clsend_ack_timer オペランドの指定値よりも大きい値を指定することをお勧めします。このオペランドの指定値が送信先 CL サーバの mch_clsend_ack_timer オペランドの指定値以下の場合，送信先 CL サーバでの待ち合わせが完了できないため，送信先 CL サーバからの送達確認メッセージを受信できないことがあります。

● mch_send_retry_count **＝メッセージ送信エラー時の送信リトライ回数**

～〈符号なし整数〉((0 ～ 65535))**《10》**

ee_mch_cmtsend 関数または ee_mch_cmtsend_sync 関数による一方送信メッセージの送信時にエラーが発生した場合の，送信リトライ回数を指定します。
0 を指定した場合は，送信リトライを行いません。

● mch_send_retry_interval ＝**送信リトライ開始間隔** T1**，送信リトライ最大間隔** T2

～送信リトライ開始間隔 T1**：〈符号なし整数〉**((1 ～ 1000))《10》**（単位：ミリ秒）**

～送信リトライ最大間隔 T2**：〈符号なし整数〉**((10 ～ 65536))《1000》**（単位：ミリ秒）**

ee_mch_cmtsend 関数または ee_mch_cmtsend_sync 関数による一方送信メッセージの送信時にエラーが発生した場合の，送信リトライ間隔を指定します。

送信リトライ開始間隔 T1**：**

送信リトライ開始時の間隔を指定します。

送信リトライ最大間隔 T2**：**

送信リトライ時の最大間隔を指定します。「送信リトライ開始間隔 T1」以上の値を指定してください。

送信リトライ間隔は，次に示す計算式によって決まります。ただし，送信リトライ間隔が「送信リトライ最大間隔 T2」より大きくなったときは，以降のリトライ間隔は「送信リトライ最大間隔 T2」の指定値に従います。

送信リトライ間隔＝送信リトライ開始間隔$T1 \times 2^{n-1}$

n**：**

送信リトライ回数

オペランドの指定によるリトライ間隔の例を次に示します。

（例 1**）**mch_send_retry_interval=1,100　mch_send_retry_count=10 **と指定した場合の，送信エラーによるリトライ間隔（単位：ミリ秒）**

```
1, 2, 4, 8, 16, 32, 64, 100, 100, 100
```

（例 2**）**mch_send_retry_interval=10,10　mch_send_retry_count=10 **と指定した場合の，送信エラーによるリトライ間隔（単位：ミリ秒）**

```
10, 10, 10, 10, 10, 10, 10, 10, 10, 10
```

● mch_send_rs_retry_interval ＝**送信リトライ開始間隔** T1**，送信リトライ最大間隔** T2

～送信リトライ開始間隔 T1**：〈符号なし整数〉**((1 ～ 1000))《10》**（単位：ミリ秒）**

～送信リトライ最大間隔 T2：〈符号なし整数〉((10 ～ 65536))《1000》（単位：ミリ秒）

ee_mch_cmtsend 関数または ee_mch_cmtsend_sync 関数による一方送信メッセージの送信時に，リソース不足のエラーが発生した場合の，送信リトライ間隔を指定します。

送信リトライ開始間隔 T1：

送信リトライ開始時の間隔を指定します。

送信リトライ最大間隔 T2：

送信リトライ時の最大間隔を指定します。「送信リトライ開始間隔 T1」以上の値を設定してください。

送信リトライ間隔の仕様は，mch_send_retry_interval オペランドと同じです。詳細については，mch_send_retry_interval オペランドの説明を参照してください。

● mch_send_max_count **＝メッセージ送信最大数**

～〈符号なし整数〉((1 ～ 1024))《16》

送信スレッドで，1 トランザクション内で ee_mch_cmtsend 関数による一方送信メッセージを送信できる最大数を指定します。

1 トランザクション内でのメッセージ送信は，同一サービスグループ下の同一サービスあてのメッセージだけです。

メッセージ送信に失敗した場合は，この指定値に関係なくメッセージ送信を中断します。

● mch_send_ack_retry_count **＝送達確認メッセージ送信エラー時の送信リトライ回数**

～〈符号なし整数〉((0 ～ 255))《3》

ee_mch_cmtsend 関数または ee_mch_cmtsend_sync 関数による一方送信メッセージの受信によって送達確認メッセージを送信した際にエラーが発生した場合の，送信リトライ回数を指定します。

0 を指定した場合は，送信リトライを行いません。

● mch_send_ack_retry_interval **＝送達確認メッセージの送信リトライ間隔**

～〈符号なし整数〉((0 ～ 1000))《0》（単位：ミリ秒）

ee_mch_cmtsend 関数または ee_mch_cmtsend_sync 関数による一方送信メッセージの受信によって送達確認メッセージを送信した際にエラーが発生した場合の，送信リトライ間隔を指定します。

0 を指定した場合は，すぐに送信リトライを行います。

● mch_send_otqwatch_time_interval **＝出力キュー閉塞監視インターバル時間**

～〈符号なし整数〉((1 ～ 65535))《3》（単位：秒）

通信障害などによってサービスグループレベルで出力キュー（OTQ）が自動閉塞となった場合に，送信先サービスグループとの間で通信を行い，出力キュー閉塞解除できるかどうかの確認をする最小時間間隔を指定します。

● mch_send_end_otqwatch ＝ {continue ｜ dct_continue ｜ timer}

～《continue》
XTC 終了処理時，出力キューに未送信メッセージが残っている場合の処理を指定します。

continue：
すべての未送信メッセージがなくなるまで，XTC を終了しません。

dct_continue：
閉塞中の出力キュー以外に残っているすべての未送信メッセージがなくなるまで，XTC を終了しません。

timer：
mch_send_end_otqwatch_time オペランドで指定した時間が経過した場合，未送信メッセージが残っていても XTC を終了します。

● mch_send_end_otqwatch_time **＝終了時の未送信メッセージ監視時間**

～〈符号なし整数〉((1 ～ 65535))《180》**（単位：秒）**
mch_send_end_otqwatch オペランドで timer を指定した場合，XTC 終了処理時に出力キューに未送信メッセージがなくなるまでの最大監視時間を指定します。

● mch_receive_limit **＝一方送信メッセージの送信元合計数**

～〈符号なし整数〉((100 ～ 100000000))《256》
トランザクション非同期の一方送信メッセージとトランザクション同期の一方送信メッセージの送信元の合計数を指定します。この指定値を超えて一方送信メッセージを受信したときは，送信元に送達確認エラーメッセージを応答します。
一方送信メッセージの送信元合計数は，次の計算式を基に算出します。

一方送信メッセージの送信元合計数　≧
送信元プロセス数×（自XTCのサービス数＋送信元プロセスの処理スレッド数）

● mch_send_otqschedule_interval **＝一方送信メッセージの再送信スケジュール間隔**

～〈符号なし整数〉((1 ～ 65535))《3》**（単位：秒）**
ee_mch_cmtsend 関数による一方送信メッセージ送信時にエラーが発生し，出力キュー閉塞とならなかった場合の，出力キューからの再送信スケジュール間隔を指定します。
この指定値は，出力キュー閉塞または出力キュー閉塞解除を行った場合に無効となります。

● mch_send_lump_count **＝一括送信するメッセージ数**

～〈符号なし整数〉((1 ～ 1024))《1》
一方送信メッセージを ee_mch_cmtsend 関数で一括送信する場合に，一括送信でき

るメッセージの最大数を指定します。
mch_send_max_count オペランドの指定値以下の値を指定してください。それよりも大きい値を指定した場合は，mch_send_max_count オペランドの指定値を一括送信できるメッセージの最大数とします。
このオペランドを省略または 1 を指定した場合，メッセージを一括送信できません。

クラスタ連携関連定義

ここでは，クラスタ連携関連定義の各オペランドについて説明します。

なお，クラスタ連携関連定義のオペランドには，HA サーバで指定できないオペランドがあります。HA サーバで指定できないオペランドについては，「HA サーバで指定できない XTC サービス定義」を参照してください。

形式

```
〔set cluster_mode＝｛Y｜N｝〕
〔set cluster_test_mode＝｛Y｜N｝〕
 set standby_start_watch_time＝待機系の起動完了の最大待ち時間
〔set standby_start_error_switch＝｛stop｜continue｝〕
〔set mch_clsend_ack_timer＝CL同期待ち監視時間〕
〔set mch_clsend_next_timer＝CL同期後続待ち監視時間〕
〔set mch_clsend_retry_count＝CL間同期メッセージの送信エラー時の送信リトライ回数〕
〔set mch_clsend_retry_interval＝送信リトライ開始間隔T1，送信リトライ最大間隔T2〕
〔set mch_clsend_rs_retry_interval＝送信リトライ開始間隔T1，送信リトライ最大間隔T2〕
〔set mch_clsend_ack_retry_count＝送達確認メッセージの送信エラー時の送信リトライ回数〕
〔set mch_clsend_ack_retry_interval＝送達確認メッセージの送信リトライ間隔〕
〔set mch_clsend_sta_timer＝開始および終了時のCL同期待ち監視時間〕
〔set mch_clsend_rrn_timer＝系切り替え時のCL同期待ち監視時間〕
〔set mch_commit_wait_msg_interval＝仕掛かり中のリソースを知らせるメッセージの出力間隔〕
〔set mch_commit_wait_msg_level＝仕掛かり中のリソースの監視レベル〕
〔set mch_xdb_buf_pool_count＝XDBの更新ログ転送用のUDP用送信バッファ確保数〕
〔set send_thread_no＝送信スレッド数〕
〔set mch_clreq_rcvthd＝｛Y｜N｝〕
```

オペランドの説明

● cluster_mode ＝ {Y ｜ N}

～《Y》

CL 連携による系切り替えを行うかどうかを指定します。

Y：

CL 連携による系切り替えを行います。

N：

CL 連携による系切り替えを行いません。

● cluster_test_mode ＝ {Y ｜ N}

～《N》

CL サーバのシミュレート機能を使用するかどうかを指定します。CL サーバのシミュレート機能については，「2.10.1　CL サーバのシミュレート機能」を参照してください。

Y：

CL サーバのシミュレート機能を使用します。
CL 単独起動機能を使用する場合は，Y を指定できません。

N：

CL サーバのシミュレート機能を使用しません。

● standby_start_watch_time ＝**待機系の起動完了の最大待ち時間**

～〈符号なし整数〉((0 ～ 3600))**（単位：秒）**
実行系の起動時，すべての待機系の起動が完了するまでの待ち時間の上限を秒単位で指定します。このオペランドは必ず指定してください。
このオペランドに指定した時間を過ぎても，すべての待機系の起動が完了しない場合，standby_start_error_switch オペランドの指定に従った処理が行われます。
0 を指定した場合は，すべての待機系の起動が完了するまで待ち続けます。この場合，すべての待機系の起動が完了しないかぎり実行系は起動しません。
なお，このオペランドの指定は，待機系では無視されます。
また，cluster_test_mode オペランドに Y を指定，または CL 単独起動機能を使用した場合，このオペランドの指定は無視されます。

● standby_start_error_switch ＝ {stop ｜ continue}

～《stop》
実行系の起動時，standby_start_watch_time オペランドで指定した時間を過ぎてもすべての待機系の起動が完了しない場合の処理を指定します。

stop：

実行系の起動処理を中止します。

continue：

起動が完了している待機系がある場合は，実行系を起動します。
ただし，起動が完了している待機系がない場合は，実行系の起動処理を中止します。

● mch_clsend_ack_timer ＝ CL **同期待ち監視時間**

～〈符号なし整数〉((1 ～ 65535))**《1》（単位：10 ミリ秒）**
実行系と待機系の同期合わせ処理の通信待ち時間の上限を 10 ミリ秒単位で指定します。このオペランドに指定した時間を過ぎても応答がない場合は，
mch_clsend_retry_count オペランドに従って再送します。
実行系が，複数の待機系からの応答メッセージを受信する場合，最初に受信した応答メッセージまでの時間が，このオペランドで指定する監視時間になります。以降の待機系からの応答メッセージの監視時間は，mch_clsend_next_timer オペランド

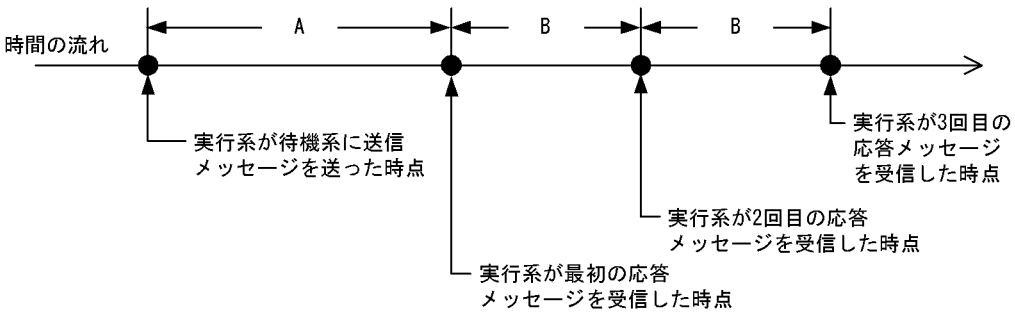
で指定します。

● mch_clsend_next_timer ＝ CL 同期後続待ち監視時間

～〈符号なし整数〉((2 ～ 128))《2》**（単位：ミリ秒）**

実行系と待機系の同期合わせ処理の通信待ち時間の上限をミリ秒単位で指定します。
このオペランドに指定した時間を過ぎても応答がない場合は，
mch_clsend_retry_count オペランドに従って再送します。
実行系が，待機系から応答メッセージを受信してから，次の応答メッセージを受信するまでの時間が，このオペランドで指定する監視時間になります。
mch_clsend_ack_timer オペランドと mch_clsend_next_timer オペランドの関係を次の図に示します。

図 5-2　mch_clsend_ack_timer **オペランドと** mch_clsend_next_timer **オペランドの関係**



（凡例）　A：mch_clsend_ack_timerの指定時間
　　B：mch_clsend_next_timerの指定時間

● mch_clsend_retry_count ＝ CL **間同期メッセージの送信エラー時の送信リトライ回数**

～〈符号なし整数〉((0 ～ 65535))《10》

実行系と待機系間での同期合わせのためのメッセージ（CL 同期メッセージなど）を送信した場合に，エラーが発生したときの送信リトライ回数の上限を指定します。
実行系の場合，送信リトライ回数がこのオペランドの値を超えた場合，以後，送信エラーとなった待機系に対してはメッセージを送信しません。
0 を指定した場合は，送信リトライを行いません。

● mch_clsend_retry_interval **＝送信リトライ開始間隔** T1**，送信リトライ最大間隔** T2

～送信リトライ開始間隔 T1**：〈符号なし整数〉**((1 ～ 1000))《1》**（単位：ミリ秒）**

～送信リトライ最大間隔 T2**：〈符号なし整数〉**((10 ～ 65536))《100》**（単位：ミリ秒）**

実行系と待機系間での同期合わせのためのメッセージ（CL 同期メッセージなど）を送信した場合に，エラーが発生したときの送信リトライ間隔をミリ秒単位で指定します。

送信リトライ開始間隔 T1：

最初の送信リトライを開始するまでの間隔を指定します。

送信リトライ最大間隔 T2：

送信リトライの最大間隔を指定します。「送信リトライ開始間隔 T1」以上の値を指定してください。

送信リトライ間隔は，次に示す計算式によって決まります。ただし，送信リトライ間隔が「送信リトライ最大間隔 T2」より大きくなったときは，以降のリトライ間隔は「送信リトライ最大間隔 T2」の指定値に従います。

送信リトライ間隔＝送信リトライ開始間隔$T1 \times 2^{n-1}$

n：

送信リトライ回数

オペランドの指定によるリトライ間隔の例を次に示します。

（例 1**）**mch_clsend_retry_interval=1,100　mch_clsend_retry_count=10 **と指定した場合の，送信エラーによるリトライ間隔（単位：ミリ秒）**

```
1, 2, 4, 8, 16, 32, 64, 100, 100, 100
```

（例 2**）**mch_clsend_retry_interval=10,10　mch_clsend_retry_count=10 **と指定した場合の，送信エラーによるリトライ間隔（単位：ミリ秒）**

```
10, 10, 10, 10, 10, 10, 10, 10, 10, 10
```

● mch_clsend_rs_retry_interval **＝送信リトライ開始間隔** T1**，送信リトライ最大間隔** T2

～送信リトライ開始間隔 T1**：〈符号なし整数〉**((1 ～ 1000))**《1》（単位：ミリ秒）**

～送信リトライ最大間隔 T2**：〈符号なし整数〉**((10 ～ 65536))**《100》（単位：ミリ秒）**

実行系と待機系間での同期合わせのためのメッセージ（CL 同期メッセージなど）を送信した場合に，リソース不足のエラーが発生したときの送信リトライ間隔をミリ秒単位で指定します。

送信リトライ開始間隔 T1：

最初の送信リトライを開始するまでの間隔を指定します。

送信リトライ最大間隔 T2：

送信リトライの最大間隔を指定します。「送信リトライ開始間隔 T1」以上の値を指定してください。

送信リトライ間隔の仕様は，mch_clsend_retry_interval オペランドと同じです。詳細については，mch_clsend_retry_interval オペランドの説明を参照してください。

● mch_clsend_ack_retry_count ＝送達確認メッセージの送信エラー時の送信リトライ回数

～〈符号なし整数〉((0 ～ 255))《3》
送達確認メッセージを待機系から実行系に送信する場合にエラーが発生したときの送信リトライ回数の上限を指定します。
0 を指定した場合は，送信リトライを行いません。

● mch_clsend_ack_retry_interval ＝送達確認メッセージの送信リトライ間隔

～〈符号なし整数〉((0 ～ 1000))《0》**（単位：ミリ秒）**
送達確認メッセージを待機系から実行系に送信する場合にエラーが発生したときの送信リトライ間隔をミリ秒単位で指定します。
0 を指定した場合は，すぐに送信リトライを行います。

● mch_clsend_sta_timer ＝開始および終了時の CL 同期待ち監視時間

～〈符号なし整数〉((1 ～ 65535))《10》**（単位：100 ミリ秒）**
XTC の開始および終了時，実行系と待機系の同期合わせ処理が行われます。このオペランドには，同期合わせ処理の通信待ち時間の上限を 100 ミリ秒単位で指定します。このオペランドに指定した時間を過ぎても応答がない場合は，
mch_clsend_retry_count オペランドの指定に従ってメッセージの再送を行います。

● mch_clsend_rrn_timer ＝系切り替え時の CL 同期待ち監視時間

～〈符号なし整数〉((1 ～ 65535))《1》**（単位：100 ミリ秒）**
系切り替え時，実行系と待機系の同期合わせ処理が行われます。このオペランドには，同期合わせ処理の通信待ち時間の上限を 100 ミリ秒単位で指定します。このオペランドに指定した時間を過ぎても応答がない場合は，mch_clsend_retry_count オペランドの指定に従ってメッセージの再送を行います。

● mch_commit_wait_msg_interval ＝仕掛かり中のリソースを知らせるメッセージの出力間隔

～〈符号なし整数〉((0 ～ 65535))《300》**（単位：100 ミリ秒）**
実行系から転送されたリソースが待機系に反映されていない場合，仕掛かり中のリソースを知らせるメッセージが出力されます。このメッセージの出力間隔を 100 ミリ秒単位で指定します。
0 を指定した場合は，仕掛かり中のリソースを知らせるメッセージは出力されません。

● mch_commit_wait_msg_level ＝仕掛かり中のリソースの監視レベル

～〈符号なし整数〉((1 ～ 2))《1》

仕掛かり中のリソースの監視レベルを指定します。

1：

トランザクション決着後の CL 同期メッセージの受信から，CL 同期済み通知メッセージの受信による処理キュー登録までの時間を監視対象とします。

2：

CL 同期済み通知メッセージの受信から，CL 同期済み通知メッセージの受信による処理キュー登録までの時間を監視対象とします。

● mch_xdb_buf_pool_count ＝ XDB **の更新ログ転送用の** UDP **用送信バッファ確保数**

～〈符号なし整数〉((1 ～ 1000))《1》

XDB の更新ログを実行系から待機系に転送する場合，UDP 用送信バッファ（UOBF）を確保します。このオペランドには，処理スレッドごとに確保する UDP 用送信バッファの最大数を指定します。

なお，このオペランドの指定によって確保される UDP 用送信バッファ数が，メモリ関連定義の udp_send_message_buf_cnt オペランドの値以上にならないように注意してください。

UDP 用送信バッファが不足した場合，不足した分の UDP 用送信バッファを新たに確保します。

また，転送に必要な UDP 用送信バッファ数が，このオペランドで指定した数の 2 倍以上で，かつ 10 面以上になる場合は，XTC 用ワーク領域（XTCPOOL）を使用して更新ログを転送します。

● send_thread_no **＝送信スレッド数**

～〈符号なし整数〉((1 ～ 50))《3》

プロセス内で使用する送信スレッド数を指定します。送信スレッド数は 3 以上を指定することをお勧めします。

送信スレッドは次に示す用途で使用します。

- ee_rpc_cmtsend 関数の再送
- ee_mch_cmtsend 関数による一方送信メッセージの送信
- 一方送信メッセージの受信による送達確認メッセージの送信
- 実行系と待機系の同期合わせ

● mch_clreq_rcvthd ＝ {Y ｜ N}

～《N》

待機系で，CL 同期メッセージを受信してから送達確認メッセージを送信するまでの処理をすべて UDP 受信スレッドで行うかどうかを指定します。

Y：

UDP 受信スレッドだけで処理を行います。

ただし，トランザクションと非同期で転送するリソースのときは，送信スレッドでも処理を行います。

N：

UDP 受信スレッドと送信スレッドで処理を行います。
この定義は，CL サーバだけ指定できます。HA サーバでは指定できません。

HA モニタ関連定義

ここでは，HA モニタ関連定義の各オペランドについて説明します。HA モニタ関連定義は，CL サーバの各サーバで指定する必要があります。HA サーバの各サーバでは指定する必要はありません。

形式

〔set ham_wait_interval_time＝HAモニタ応答待ちメッセージの出力間隔〕
〔set ham_onl_interval_time＝実行系の系監視スレッドのイベント監視時間〕
〔set ham_sby_interval_time＝待機系の系監視スレッドのイベント監視時間〕
〔set ham_stop_msg_interval_time＝TP1/EE停止待ちメッセージの出力間隔〕
〔set ham_isolate_wtor_interval_time＝eetrbwtorコマンドの入力待ちメッセージの出力間隔〕

オペランドの説明

● ham_wait_interval_time ＝ HA モニタ応答待ちメッセージの出力間隔

～〈符号なし整数〉((0 ～ 60))《5》（単位：分）
HA モニタからの応答を待っている間，KFSB55901-E メッセージが出力されます。このメッセージの出力間隔を分単位で指定します。
0 を指定した場合，KFSB55901-E メッセージは出力されません。

● ham_onl_interval_time ＝実行系の系監視スレッドのイベント監視時間

～〈符号なし整数〉{100 ｜ 250 ｜ 500 ｜ 1000}《1000》（単位：ミリ秒）
実行系の系監視スレッドが，終了などのイベントを監視する間隔をミリ秒単位で指定します。
このオペランドの値を大きくすると，イベント監視処理によるオーバヘッドは小さくなりますが，イベント発生からイベント検出までの時間が長くなります。
このオペランドの値を小さくすると，イベント発生からイベント検出までの時間は短くなりますが，イベント監視処理によるオーバヘッドが大きくなります。

● ham_sby_interval_time ＝待機系の系監視スレッドのイベント監視時間

～〈符号なし整数〉{100 ｜ 250 ｜ 500 ｜ 1000}《100》（単位：ミリ秒）
待機系の系監視スレッドが，系切り替えや，終了などのイベントを監視する間隔をミリ秒単位で指定します。
このオペランドの値を大きくすると，イベント監視処理によるオーバヘッドは小さくなりますが，イベント発生からイベント検出までの時間が長くなります。
このオペランドの値を小さくすると，イベント発生からイベント検出までの時間は短くなりますが，イベント監視処理によるオーバヘッドが大きくなります。
系切り替えの発生をなるべく早く検知したい場合は，このオペランドの値を小さくしてください。

● ham_stop_msg_interval_time ＝ TP1/EE 停止待ちメッセージの出力間隔

～〈符号なし整数〉((1 ～ 60))《5》（単位：分）

系監視スレッドからの eesvstop コマンドの実行後から，TP1/EE が停止するまでの間，KFSB55902-E メッセージが出力されます。このメッセージの出力間隔を分単位で指定します。

系監視スレッドで eesvstop コマンドを実行するのは，次のケースです。

- 実行系の TP1/EE が孤立した場合の，eetrbwtor 実行時
- 実行系の TP1/EE が eesvstop で正常終了する場合の，待機系の TP1/EE の終了時

● ham_isolate_wtor_interval_time ＝ eetrbwtor コマンドの入力待ちメッセージの出力間隔

～〈符号なし整数〉((0 ～ 60))《5》（単位：分）

実行系孤立状態が発生した場合，eetrbwtor コマンドの入力を促す KFSB85801-Q メッセージが出力されます。このメッセージの出力間隔を分単位で指定します。

0 を指定した場合，KFSB85801-Q メッセージは 1 回だけしか出力されません。

送信先サービス関連定義（コマンド形式）

ここでは，送信先サービス関連定義（コマンド形式）の各定義コマンドについて説明します。

eemchsrvdef（出力キュー情報定義）

機能

高速メッセージ送信制御による一方送信メッセージ送信先のサービスグループ名とサービス名を指定します。指定されたサービスグループ名とサービス名の組み合わせごとに出力キュー（OTQ）を作成し，出力キュー状態を管理します。

サービスグループ名は，最大 1024 個を定義できます。

同じサービスグループ名を複数の eemchsrvdef 定義コマンドで定義した場合，-a オプションと -b オプションは最初に定義した値が有効となります。

コマンドの形式

```
{ {eemchsrvdef -g サービスグループ名
                -v サービス名〔,サービス名…〕
               〔-a〕
               〔-b〕
} }
```

オプションの説明

● -g **サービスグループ名**

～〈1 ～ 31 文字の識別子〉

一方送信メッセージの送信要求を行うサービスグループ名を指定します。
サービスグループ名は，サービスグループ情報関連定義の eesvgdef 定義コマンドで定義しておく必要があります。

● -v **サービス名**

～〈1 ～ 31 文字の識別子〉

一方送信メッセージの送信要求を行うサービス名を指定します。
同一サービスグループ下に，最大 32000 個のサービス名を指定できます。

● -a

オンライン開始時に出力キューの自動閉塞を行う場合に指定します。

● -b

一方送信メッセージの送信失敗時に，エラー要因に応じて出力キューの自動閉塞を行う場合に指定します。出力キューが自動閉塞する要因については，「2.2.6　出力キューによるメッセージ管理」を参照してください。

UDP グループ情報関連定義（コマンド形式）

ここでは，UDP グループ情報関連定義（コマンド形式）の各定義コマンドについて説明します。

なお，UDP グループ情報関連定義（コマンド形式）の定義コマンドには，HA サーバで指定できない定義コマンドがあります。HA サーバで指定できない定義コマンドについては，「HA サーバで指定できない XTC サービス定義」を参照してください。

eeudpdef（あて先 UDP グループ情報定義）

機能

UDP 通信機能を使用して他プロセスにサービス要求する場合，要求先プロセスごとに，通信種別，ホスト名，ポート番号，通信パラメタなどを指定します。

この定義コマンドは，次の通信をする場合に必要になります。

- トランザクション同期の一方送信要求（ee_mch_cmtsend_sync 関数）
- トランザクション非同期の一方送信要求（ee_mch_cmtsend 関数）
- UDP プロトコルを使用した RPC 通信（ee_rpc_call 関数に EERPC_UDP オプションを指定）

この定義コマンドの指定値と，サービス要求先の定義との関連を次の表に示します。

表 5-8　eeudpdef 定義コマンドの指定値とサービス要求先の定義との関連

項番	eeudpdef 定義コマンドの指定値	サービス要求先	
		ユニキャストによるメッセージ送信	マルチキャストによるメッセージ送信
1	ホスト名	要求先プロセスの LAN アダプタのホスト名	要求先プロセスの UDP グループ情報関連定義の myudprcvdef 定義コマンドまたは clgrpdef 定義コマンドの -a オプションの指定値
2	ポート番号	要求先プロセスの UDP グループ情報関連定義の myudprcvdef 定義コマンドまたは clgrpdef 定義コマンドの -p オプションの指定値	要求先プロセスの UDP グループ情報関連定義の myudprcvdef 定義コマンドまたは clgrpdef 定義コマンドの -p オプションの指定値

複数の eeudpdef 定義コマンドで，同じあて先 UDP グループ名を指定した場合は，あて先 UDP グループが同じになります。送信先のポート番号は，あて先 UDP グループのポートの中からランダムで選択します。

eeudpdef 定義コマンドごとに，LAN アダプタの異なる送信 UDP グループを指定した場合は，該当するあて先 UDP グループに対する送信時に，複数の LAN アダプタを使用し

た負荷分散通信ができます。ただし，同じあて先 UDP グループ名を指定する場合は，eeudpdef 定義コマンドの -c オプションの指定値を一致させる必要があります。不一致の場合は，XTC の起動に失敗します。

負荷分散通信の詳細については，「2.4.8　負荷分散機能」を参照してください。

コマンドの形式

```
{ {eeudpdef -g あて先UDPグループ名
             -c {UC | MC}
             -a ホスト名
             -p ポート番号〔,ポート番号〕…
            〔-s 送信UDPグループ名〕
            〔-r 受信UDPグループ名〕
            〔-w 再送タイマ値〕
            〔-W 輻輳時の再送タイマ値〕
            〔-R 再送タイマのタイムアウト上限回数〕
} }
```

オプションの説明

● -g **あて先** UDP **グループ名**

～〈1 ～ 31 **文字の識別子〉**

あて先 UDP グループを識別するためのあて先 UDP グループ名を指定します。ただし，「CL」はクラスタグループを表す予約語であるため指定できません。「CL」を指定した場合は，XTC の起動に失敗します。

複数の UDP グループ情報関連定義の eeudpdef 定義コマンドで同じあて先 UDP グループ名を指定した場合は，あて先 UDP グループは同じになります。送信先のポート番号は，あて先 UDP グループのポートの中からランダムで選択します。

● -c {UC | MC}

あて先 UDP グループに対する通信方式を指定します。

UC：

ユニキャストでメッセージを送信します。

MC：

マルチキャストでメッセージを送信します。

● -a **ホスト名**

～〈1 ～ 255 **文字のホスト名〉**

送信先のホスト名を指定します。指定するホスト名は，通信方式によって異なります。

通信方式がユニキャスト（-c オプションに UC を指定）の場合は，LAN アダプタのホスト名を指定します。通信方式がマルチキャスト（-c オプションに MC を指定）の場合は，ホストグループのホスト名を指定します。

ホスト名は，/etc/hosts ファイル，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。

● -p **ポート番号**

～〈符号なし整数〉((5001 ～ 65535))

送信先のポート番号を指定します。ポート番号を複数指定した場合は，送信のたびにランダムで送信先ポートを選択します。ポート番号は，128 個まで指定できます。

● -s **送信** UDP **グループ名**

～〈1 ～ 31 **文字の識別子〉**

該当するあて先 UDP グループに対するサービス要求時に，メッセージ送信で使用する送信 UDP グループ名を指定します。省略した場合，XTC の定義ファイル内で最初に定義している送信 UDP グループを使用します。

● -r **受信** UDP **グループ名**

～〈1 ～ 31 **文字の識別子〉**

該当するあて先 UDP グループに対するサービス要求時に，肯定応答（ACK）または否定応答（NACK）の受信で使用する受信 UDP グループ名を指定します。受信 UDP グループ名として「CL」を指定した場合は，クラスタグループ（UDP グループ情報関連定義の clgrpdef 定義コマンドで指定）を使用します。

省略した場合は，XTC の定義ファイル内で最初に定義している受信 UDP グループ（clgrpdef 定義コマンドで指定している受信 UDP グループを含む）を使用します。

● -w **再送タイマ値**

～〈符号なし整数〉((1 ～ 65535))**《送信** UDP **グループの再送タイマ》（単位：ミリ秒）**

メッセージ送信から肯定応答または否定応答を受信するまでの待ち時間をミリ秒単位で指定します。待ち時間が指定値を超えてタイムアウトした場合は，再送処理を行います。

● -W **輻輳時の再送タイマ値**

～〈符号なし整数〉((1 ～ 65535))**《**-w **オプションの指定値》（単位：ミリ秒）**

輻輳状態での，メッセージ送信から肯定応答または否定応答を受信するまでの待ち時間をミリ秒単位で指定します。待ち時間が指定値を超えてタイムアウトした場合は，再送処理を行います。

● -R **再送タイマのタイムアウト上限回数**

～〈符号なし整数〉((0 ～ 65535))**《送信** UDP **グループの再送タイマのタイムアウト上限回数》**

再送タイマのタイムアウト発生時の再送処理回数を指定します。タイムアウト回数がこのオプションの指定値を超えた場合は，送信失敗となります。

このオプションに 0 を指定した場合は，タイムアウト時の再送処理を行いません。

注意事項

指定した送信先 XTC と通信できない場合，次のことを確認してください。

- -c オプションの指定値が UC の場合，-a オプションで指定したホスト上で XTC が起

動されているかどうか。
- -c オプションの指定値が MC の場合，次の条件が満たされているかどうか。
  - 送信先 XTC が起動されている。
  - -a オプションで指定したホスト名が，送信先 XTC の UDP グループ情報関連定義の myudprcvdef 定義コマンドの -a オプション，または clgrpdef 定義コマンドの -m オプションもしくは -a オプションの指定値と同じである。
- -p オプションで指定したポート番号が，送信先 XTC の UDP グループ情報関連定義の myudprcvdef 定義コマンドまたは clgrpdef 定義コマンドの -p オプションの指定値が同じかどうか。
- 送信先ホストが自ホストの場合に，UDP グループ情報関連定義の myudpsnddef 定義コマンドの -k オプションを指定しているかどうか。

myudpsnddef（送信 UDP グループ情報定義）

機能

UDP 通信機能を使用して他プロセスと通信する場合，送信用ポート，通信パラメタなどを指定します。この定義コマンドは，必ず一つ以上定義します。

送信用ポートのポート数とスレッド数を基に，次の表に示す規則に従い，送信用ポートの占有または共有を決定します。処理スレッドと送信スレッドは，送信用ポートの使用頻度が高いため，項番 2 の条件を満たすことをお勧めします。

表 5-9　送信用ポートの占有，共有を決定する規則

項番	条件	占有／共有
1	送信用ポート数≧メインスレッドを除いた全スレッド数	すべてのスレッドが占有※になります。
2	送信用ポート数＞処理スレッド数＋送信スレッド数	処理スレッドおよび送信スレッドは占有，それ以外のスレッドは共有になります。
3	上記以外	すべてのスレッドが共有になります。

注※

送信用ポート数が全スレッド数を超過している場合，送信用ポートは，全スレッドの数分作成します。

コマンドの形式

```
{ {myudpsnddef -g 送信UDPグループ名
              {-p ポート番号〔,ポート番号〕…｜-P ポート数}
              〔-a LANアダプタのホスト名〕
              〔-b システムのUDPソケット送信バッファのサイズ〕
              〔-T マルチキャスト時のTTL〕
              〔-w 再送タイマ値〕
              〔-W 輻輳時の再送タイマ値〕
              〔-R 再送タイマのタイムアウト上限回数〕
              〔-L パケットロス上限〕
```

```
                    〔-l パケットロス下限〕
                    〔-U 連続送信パケット数下限〕
                    〔-k〕
} }
```

オプションの説明

● -g **送信** UDP **グループ名**

～〈1 ～ 31 **文字の識別子〉**

送信 UDP グループを識別するための送信 UDP グループ名を指定します。
同一の送信 UDP グループ名がすでに定義されている場合は，XTC の起動に失敗します。

● -p **ポート番号**

～〈符号なし整数〉((5001 ～ 65535))

送信用ポートのポート番号を明示的に指定する場合，このオペランドでポート番号を指定します。ポート番号は，128 個まで指定できます。
指定されたポート番号がすでに使用されていた場合は，XTC の起動に失敗します。
-p オプションまたは -P オプションのどちらか一方だけを指定する必要があります。
-p オプションおよび -P オプションの両方を指定，または両方を省略した場合は，XTC プロセスの起動に失敗します。

● -P **ポート数**

～〈符号なし整数〉((1 ～ 128))

送信用ポートのポート番号を OS による自動割り当てにする場合は，このオペランドでポート数を指定します。指定値分のポートを作成できなかった場合は，XTC の起動に失敗します。
-p オプションまたは -P オプションのどちらか一方だけを指定する必要があります。
-p オプションおよび -P オプションの両方を指定，または両方を省略した場合は，XTC の起動に失敗します。

● -a LAN **アダプタのホスト名**

～〈1 ～ 255 **文字のホスト名〉**

複数の LAN アダプタを持つ構成にした場合に，送信に使用する LAN アダプタのホスト名を指定します。省略した場合は，送信に使用する LAN アダプタを OS が自動的に決定します。
ホスト名は，/etc/hosts ファイル，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。

● -b **システムの** UDP **ソケット送信バッファのサイズ**

～〈符号なし整数〉((2048 ～ 2147483647))**《**2097152**》（単位：バイト）**

システムの UDP ソケット送信バッファのサイズをバイト単位で指定します。
指定できる範囲は OS に依存します。詳細については，OS のマニュアルを参照してください。

● -T **マルチキャスト時の** TTL

〜〈符号なし整数〉((0 〜 255))

マルチキャストメッセージ送信時のマルチキャストパケットの生存期間（TTL）を指定します。省略した場合は，OS のデフォルト値となります。

● -w **再送タイマ値**

〜〈符号なし整数〉((1 〜 65535))**《10》（単位：ミリ秒）**

メッセージ送信から肯定応答または否定応答を受信するまでの待ち時間をミリ秒単位で指定します。待ち時間が指定値を超えてタイムアウトした場合は，再送処理を行います。

メッセージ送信時に，UDP ソケット送信バッファが送信できる状態になるまでの，待ち時間としても使用します。

● -W **輻輳時の再送タイマ値**

〜〈符号なし整数〉((1 〜 65535))**《20》（単位：ミリ秒）**

輻輳状態での，メッセージ送信から肯定応答または否定応答を受信するまでの待ち時間をミリ秒単位で指定します。待ち時間が指定値を超えてタイムアウトした場合は，再送処理を行います。

輻輳状態でのメッセージ送信時に，UDP ソケット送信バッファが送信できる状態になるまでの，待ち時間としても使用します。

● -R **再送タイマのタイムアウト上限回数**

〜〈符号なし整数〉((0 〜 65535))**《10》**

再送タイマのタイムアウト発生時の再送処理回数を指定します。タイムアウト回数がこのオプションの指定値を超えた場合は，送信失敗となります。

このオプションに 0 を指定した場合は，タイムアウト時の再送処理を行いません。

● -L **パケットロス上限**

〜〈符号なし整数〉((1 〜 100))**《30》（単位：%）**

監視対象範囲内（rpc_udp_congestion_packet_count オペランドで指定）での再送パケット比率が，このオプションの指定値以上になった場合に輻輳制御を開始します。

パケットロス上限は，パケットロス下限より大きくなるように設定してください。

パケットロス上限がパケットロス下限以下の場合は，XTC の起動に失敗します。

● -l **パケットロス下限**

〜〈符号なし整数〉((0 〜 99))**《20》（単位：%）**

監視対象範囲内（rpc_udp_congestion_packet_count オペランドで指定）での再送パケット比率が，このオプションの指定値以下になった場合に輻輳制御を終了します。

パケットロス下限は，パケットロス上限より小さくなるように設定してください。

パケットロス下限がパケットロス上限以上の場合は，XTC の起動に失敗します。

● -U **連続送信パケット数下限**

～〈符号なし整数〉((1 ～ 65536))《5》
輻輳時の連続送信パケット数の最低値を指定します。
輻輳時は，連続送信パケット数を自動計算します。自動計算した連続送信パケット数が少ない場合，ネットワークの負荷は軽減されます。しかし，通信時間が増加することがあります。このオプションによって連続送信パケット数の下限値を設定することで，通信時間の増加を回避できます。

● -k
自マシンをマルチキャストメッセージの送信対象にする場合に指定します。省略した場合は，自マシンはマルチキャストメッセージの送信対象外になります。
CL サーバ同士でのマルチキャストによる通信時にこのオプションを指定した場合は，受信負荷が増加します。そのため，CL サーバ同士の通信に限定して使用する場合は，このオプションを省略することをお勧めします。

myudprcvdef（受信 UDP グループ情報定義）

機能

UDP 通信機能を使用して他プロセスと通信する場合に，受信用ポート，システムの UDP の受信バッファサイズなどを指定します。指定した受信用ポートごとに，UDP 受信スレッドを作成します。

HA サーバの場合，必ず一つ以上定義します。CL サーバの場合，クラスタグループで代用できるため省略できます。ただし，クラスタ構成のサーバで CL 単独起動機能を使用する場合は，必ず一つ以上定義してください。

コマンドの形式

```
{ {myudprcvdef -g 受信UDPグループ名
                -p ポート番号〔,ポート番号〕…
              〔-a ホストグループのホスト名〔:LANアダプタのホスト名〕
              〔,ホストグループのホスト名〔:LANアダプタのホスト名〕〕…〕
              〔-B UDPの受信バッファサイズ〕
} }
```

オプションの説明

● -g **受信** UDP **グループ名**
～〈1 ～ 31 **文字の識別子〉**
受信 UDP グループを識別するための受信 UDP グループ名を指定します。ただし，「CL」はクラスタグループを表す予約語であるため指定できません。同一の受信 UDP グループ名がすでに定義されている場合，または「CL」を指定した場合は，XTC プロセスの起動に失敗します。

● -p **ポート番号**
～〈符号なし整数〉((5001 ～ 65535))
受信用ポートのポート番号を指定します。ポート番号は，128 個まで指定できます。

指定されたポート番号がすでに使用されていた場合は，XTC プロセスの起動に失敗します。

● -a **ホストグループのホスト名：**LAN **アダプタのホスト名**
ホストグループのホスト名～〈1 ～ 255 **文字のホスト名〉**
LAN **アダプタのホスト名～〈**1 ～ 255 **文字のホスト名〉**
所属するホストグループのホスト名を指定します。また，複数の LAN アダプタを持つ構成の場合，マルチキャストメッセージの受信に使用する LAN アダプタのホスト名も指定します。LAN アダプタのホスト名を省略した場合は，OS が自動的に LAN アダプタを決定します。
ホスト名は，/etc/hosts ファイル，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。
このオプションを省略した場合，該当する受信用ポートは，ユニキャストメッセージだけを受信します。

● -B UDP **の受信バッファサイズ**
～〈符号なし整数〉((2048 ～ 2147483647))**《**3145728**》（単位：バイト）**
システムの UDP の受信バッファサイズをバイト単位で指定します。
指定できる範囲は OS に依存します。詳細については，OS のマニュアルを参照してください。

clgrpdef（クラスタグループ情報定義）

機能

CL サーバの場合，クラスタ情報，受信用ポート，通信パラメタなどを指定します。指定した受信用ポートごとに，UDP 受信スレッドを作成します。

HA サーバの場合は，この定義コマンドを指定しないでください。

この定義コマンドを省略すると，CL 単独起動機能が有効になります。

この定義コマンドは，受信 UDP グループ情報定義として使用することもできます。その場合は，受信 UDP グループ名は「CL」となります。

この定義コマンドを複数指定した場合，送信先のポート番号は，すべての clgrpdef 定義コマンドのポートの中からランダムで選択されます。

clgrpdef 定義コマンドごとに，LAN アダプタが異なる送信 UDP グループを指定した場合に，CL サーバ同士の送信をするときは，複数の LAN アダプタを使用した負荷分散通信ができます。負荷分散通信の詳細については，「2.4.8　負荷分散機能」を参照してください。

なお，この定義コマンドを複数指定する場合は，次のオプションの指定値を一致させる必要があります。

- -c

- -p
- -m
- -n
- -a
- -B

コマンドの形式

```
{ {clgrpdef -c クラスタ識別子
             -p ポート番号〔,ポート番号〕…
             -m ホストグループのホスト名〔:LANアダプタのホスト名〕
            -n ノード識別子:HAモニタのホスト名〔:LANアダプタのホスト名〕〔,
ノード識別子:HAモニタのホスト名〔:LANアダプタのホスト名〕〕…
            〔-a ホストグループのホスト名〔:LANアダプタのホスト名〕〔,ホスト
グループのホスト名〔:LANアダプタのホスト名〕〕…〕
            〔-s 送信UDPグループ名〕
            〔-w 再送タイマ値〕
            〔-W 輻輳時の再送タイマ値〕
            〔-R 再送タイマのタイムアウト上限回数〕
            〔-B UDPの受信バッファサイズ〕
} }
```

オプションの説明

● -c **クラスタ識別子**

～〈4 文字の識別子〉

CL サーバを一意に識別するためのクラスタ識別子を指定します。
クラスタグループを構成する全サーバ間で，同一のクラスタ識別子を指定します。
異なる識別子を指定した場合は，XTC の起動に失敗します。
複数のクラスタグループが存在する場合は，各クラスタグループ間で異なる識別子を指定します。同一の識別子を指定した場合は，ee_mch_cmtsend_sync 関数または ee_mch_cmtsend 関数のメッセージが破棄されるおそれがあります。

● -p **ポート番号**

～〈符号なし整数〉((5001 ～ 65535))

受信用ポートのポート番号を指定します。ポート番号は，128 個まで指定できます。
指定されたポート番号がすでに使用されていた場合は，XTC プロセスの起動に失敗します。
送信時は，送信先ポート番号として使用します。

● -m **ホストグループのホスト名：**LAN **アダプタのホスト名**

ホストグループのホスト名～〈1 ～ 255 **文字のホスト名〉**
LAN **アダプタのホスト名～〈**1 ～ 255 **文字のホスト名〉**

CL サーバ同士でのマルチキャストメッセージによる通信時に使用するホストグループのホスト名を指定します。また，複数の LAN アダプタを持つ構成の場合，マルチキャストメッセージの受信に使用する LAN アダプタのホスト名も指定します。

LAN アダプタのホスト名を省略した場合は，OS が自動的に LAN アダプタを決定します。
ホスト名は，/etc/hosts ファイル，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。

● -n **ノード識別子：**HA **モニタのホスト名：**LAN **アダプタのホスト名**

クラスタグループ内の全サーバのノード情報を指定します。ノード情報は，最大で 3 つ指定できます。4 つ以上を指定した場合は，XTC の起動に失敗します。
自ノード情報は必ず指定しなければなりません。自ノード情報を指定していない場合は，XTC の起動に失敗します。
すべてのノード情報で，ノード識別子，HA モニタのホスト名，および LAN アダプタのホスト名に異なる値を指定する必要があります。同一の値を指定していた場合は，XTC の起動に失敗します。
また，このオプションに指定する値は実行系と待機系で同じ順番で指定してください。実行系と待機系で異なる順番で指定した場合は，XTC の起動に失敗します。

- **ノード識別子**
  ～〈4 文字の識別子〉
  該当するサーバの TP1/EE サービス定義で指定した RPC 関連定義のノード識別子（node_id オペランドの指定値）を指定します。
- HA **モニタのホスト名**
  ～〈1 ～ 255 文字のホスト名〉
  該当するサーバの HA モニタの環境設定（sysdef ファイル）の name オペランドで指定したホスト名を指定します。
- LAN **アダプタのホスト名**
  ～〈1 ～ 255 文字のホスト名〉
  該当するサーバの LAN アダプタのホスト名を指定します。省略時は，HA モニタのホスト名を LAN アダプタのホスト名として使用します。

● -a **ホストグループのホスト名：**LAN **アダプタのホスト名**

ホストグループのホスト名～〈1 ～ 255 文字のホスト名〉
LAN **アダプタのホスト名～〈1 ～ 255 文字のホスト名〉**

該当する受信用ポートを使用して，他プロセスからのサービス要求を受け付ける場合，参加するホストグループのホスト名を指定します。また，複数の LAN アダプタを持つ構成の場合，マルチキャストメッセージの受信に使用する LAN アダプタのホスト名も指定します。LAN アダプタのホスト名を省略した場合は，OS が自動的に LAN アダプタを決定します。
このオプションを省略した場合，該当する受信用ポートに対するサービス要求は，ユニキャストメッセージだけでできるようになります。
ホスト名は，/etc/hosts ファイル，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。

● -s **送信** UDP **グループ名**

～〈1 ～ 31 文字の識別子〉

該当するあて先 UDP グループに対するサービス要求時に，メッセージ送信で使用する送信 UDP グループ名を指定します。省略した場合は，XTC の定義ファイル内で，最初に定義している送信 UDP グループを使用します。

● -w **再送タイマ値**

～〈符号なし整数〉((1 ～ 65535))**《送信** UDP **グループの再送タイマ》（単位：ミリ秒）**

メッセージ送信から肯定応答または否定応答を受信するまでの待ち時間をミリ秒単位で指定します。待ち時間が指定値を超えてタイムアウトした場合は，再送処理を行います。

● -W **輻輳時の再送タイマ値**

～〈符号なし整数〉((1 ～ 65535))**《**-w **オプションの指定値》（単位：ミリ秒）**

輻輳状態での，メッセージ送信から肯定応答または否定応答を受信するまでの待ち時間をミリ秒単位で指定します。待ち時間が指定値を超えてタイムアウトした場合は，再送処理を行います。

● -R **再送タイマのタイムアウト上限回数**

～〈符号なし整数〉((0 ～ 65535))**《送信** UDP **グループの再送タイマのタイムアウト上限回数》**

再送タイマのタイムアウト発生時の再送処理回数を指定します。タイムアウト回数がこのオプションの指定値を超えた場合は，送信失敗となります。
なお，0 ～ 2 を指定した場合でも，タイムアウト上限回数は 3 となります。
実行系から待機系にリソース情報を送信するときに，再送に対する肯定応答（ACK），または否定応答（NACK）を再送処理の間に一度も受信しなかった場合，待機系をメッセージ送信の対象外とします。

● -B UDP **の受信バッファサイズ**

～〈符号なし整数〉((2048 ～ 2147483647))**《**3145728**》（単位：バイト）**

システムの UDP の受信バッファサイズをバイト単位で指定します。
指定できる範囲は OS に依存します。詳細については，OS のマニュアルを参照してください。

HA サーバで指定できない XTC サービス定義

ここでは，HA サーバで指定できない XTC サービス定義について説明します。

HA サーバで指定できない XTC サービス定義

HA サーバで指定できない XTC サービス定義の一覧を次の表に示します。

表 5-10　HA サーバで指定できない XTC サービス定義の一覧

項番	定義	指定できないオペランドおよび定義コマンド
1	メモリ関連定義	• memory_xdb_area_size • memory_xdb_limit_size
2	RPC 関連定義	• rpc_recv_permanence
3	ユーザサービス関連定義	• system_start_ui
4	トラブルシュート関連定義	• trb_quedump_io_watch_time • trb_quedump_io_watch_msg_time
5	トランザクション関連定義	• trn_expiration_time_ui
6	クラスタ連携関連定義	• cluster_test_mode • standby_start_watch_time • standby_start_error_switch • mch_clsend_ack_timer • mch_clsend_next_timer • mch_clsend_retry_count • mch_clsend_retry_interval • mch_clsend_rs_retry_interval • mch_clsend_ack_retry_count • mch_clsend_ack_retry_interval • mch_commit_wait_msg_interval • mch_commit_wait_msg_level • mch_xdb_buf_pool_count • mch_clsend_sta_timer • mch_clsend_rrn_timer
7	HA モニタ関連定義	すべてのオペランドが指定できません。
8	UDP グループ情報関連定義（コマンド形式）	• clgrpdef 定義コマンド

6 運用

この章では，XTC の開始・終了方法，および運用コマンドについて説明します。

6.1 XTC の開始

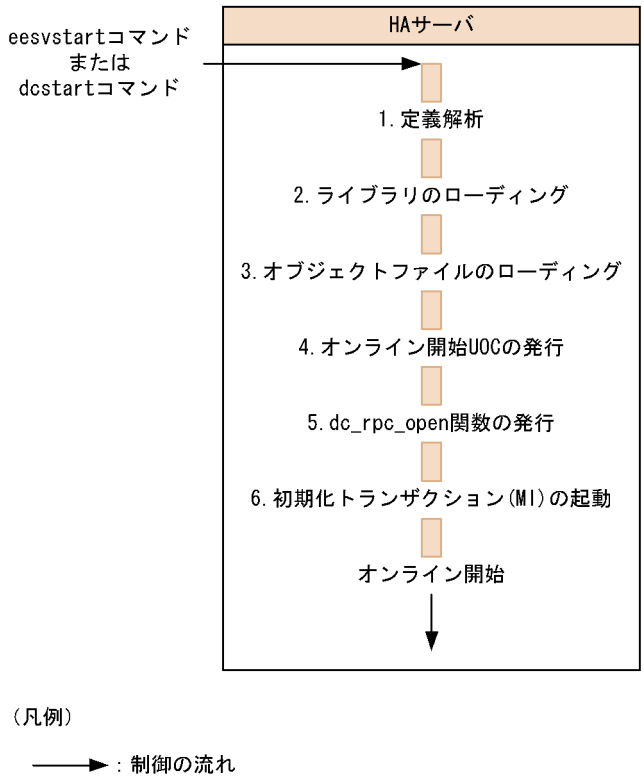
ここでは，XTC の開始方法，開始モード，および開始時のシステムの処理について説明します。

6.1.1 HA サーバでの XTC の開始

HA サーバの場合，開始方法と開始モードは TP1 キャッシュ機能を使用しないときと同様です。詳細については，マニュアル「TP1/Server Base Enterprise Option 使用の手引」を参照してください。

システムの処理を次の図に示します。

図 6-1 HA サーバでの XTC の開始



説明

1. 定義解析
   TP1/EE の定義解析を実行します。定義解析では，定義の形式および値の妥当性をチェックします。この処理によって，サーバの構成が決定します。
2. ライブラリのローディング

ライブラリをローディングします。次の表に，使用する機能とロードするライブラリの対応を示します。ライブラリのローディングでエラーが発生した場合は，プロセスダウンします。

表 6-1 使用する機能とロードするライブラリの対応

項番	使用する機能	ロードするライブラリ
1	XTC	XTC 用のライブラリ
2	XDB	XDB 用のライブラリ
3	MCP	MCP 用のライブラリ

3. MCP のオブジェクトファイルのローディング
   MCP を使用する場合は，MCP 定義オブジェクト生成ユティリティで生成したオブジェクトファイルをローディングします。オブジェクトファイルのローディングで障害が発生した場合は，プロセスダウンします。
4. オンライン開始 UOC の発行
   オンライン開始 UOC を発行します。オンライン開始 UOC については，マニュアル「TP1/Server Base Enterprise Option プログラム作成の手引」を参照してください。
   オンライン開始 UOC のリターン後の動作を次の表に示します。

表 6-2 オンライン開始 UOC リターン後の動作

項番	リターン値	動作
1	EEMCH_UOC_OK	開始処理続行
2	EEMCH_UOC_NG	プロセスダウン

5. dc_rpc_open 関数の発行
   dc_rpc_open 関数を発行します。
6. 初期化トランザクション（MI）の起動
   初期化トランザクション（MI）を起動します。

6.1.2 CL サーバでの XTC の開始

CL サーバの場合，次のどちらかの方法で XTC を開始します。

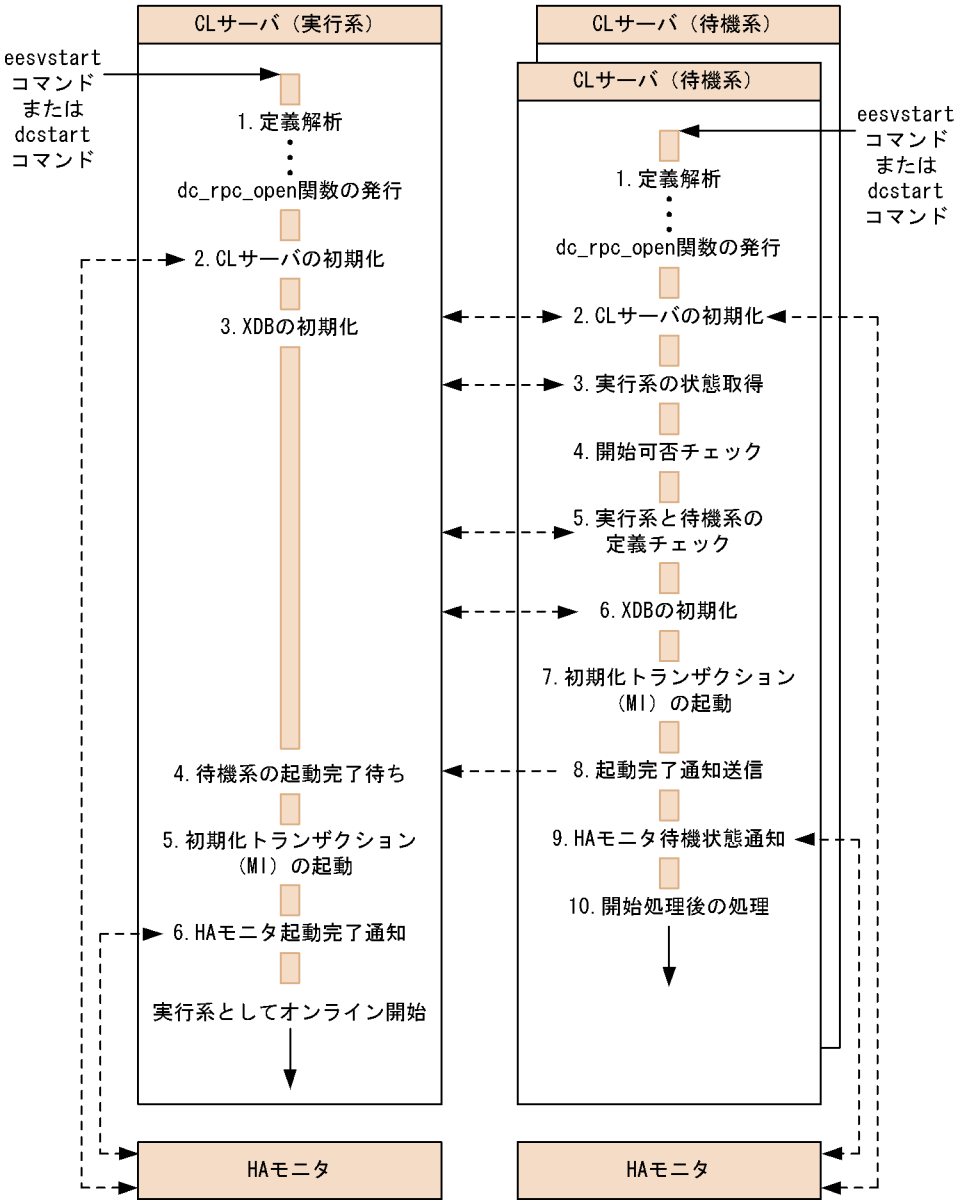
- TP1/EE に対して eesvstart コマンドを実行する。
- TP1/Server Base に対して dcstart コマンドを実行する。

コマンドは，実行系と待機系の両方で実行してください。

開始モードは，前回オンラインの終了状態に関係なく，常に正常開始となります。また，XTC を開始するとその延長で XDB も開始されます。

システムの処理を次の図に示します。

図 6-2 CL サーバでの XTC の開始



（凡例）

：制御の流れ

：通信の流れ

実行系の処理の説明

1. 定義解析～ dc_rpc_open 関数の発行
   HA サーバの場合と同様です。
2. CL サーバの初期化
   HA モニタや CL サーバに関する初期化処理を実行します。この処理によって，自 TP1/EE の系が決定します。
3. XDB の初期化
   XDB を初期化します。XDB の開始の詳細については，マニュアル「TP1/EE/Extended Data Cache 使用の手引」を参照してください。
4. 待機系の起動完了待ち
   待機系の起動完了を待ちます。
   CL 連携する場合（クラスタ連携関連定義の cluster_mode オペランドに Y を指定した場合）は，次のどちらかの条件を満たすまで，すべての待機系からの起動完了通知の受信を待ちます。
   ・すべての待機系から起動完了通知を受信した。
   ・起動完了通知の受信タイムアウト（クラスタ連携関連定義の standby_start_watch_time オペランドで指定）となった。
   起動完了通知の受信タイムアウト時の動作を次の表に示します。

表 6-3　起動完了通知の受信タイムアウト時の動作

項番	standby_start_error_switch オペランドの指定値	動作
1	continue	起動が完了している待機系がある場合は，実行系を起動します。 起動が完了している待機系がない場合は，実行系の起動処理を中止します。
2	stop	実行系の起動処理を中止します。

   CL 連携しない場合（クラスタ連携関連定義の cluster_mode オペランドに N を指定した場合），および CL サーバのシミュレート機能を使用する場合（クラスタ連携関連定義の cluster_test_mode オペランドに Y を指定した場合）は，何も行いません。
5. 初期化トランザクション（MI）の起動
   初期化トランザクション（MI）を起動します。このとき，実行系での起動であることをユーザに通知するため，トランザクションインタフェース情報の系情報に EERPC_SYSTEM_ONLINE（実行系を意味する値）を設定して起動します。
6. HA モニタ起動完了通知
   HA モニタに対して起動完了通知を行います。以降，自 TP1/EE に障害が発生した場合，系切り替えが発生します。

待機系の処理の説明

1. 定義解析～ dc_rpc_open 発行
   HA サーバの場合と同様です。

2. CL サーバの初期化
   実行系の場合と同様です。
   なお，この処理の間に実行系との通信が行われます。そのため，以降の処理は実行系が起動していることが前提となります。
3. 実行系の状態取得
   実行系と通信を行い，現時点での実行系の状態を取得します。
4. 開始可否チェック
   実行系の状態が，待機系の起動完了待ちよりも先に進んでいる場合は，プロセスダウンします。
5. 実行系と待機系の定義チェック
   実行系と通信を行い，実行系の定義内容に対して自 TP1/EE の定義内容が妥当であるかどうかをチェックします。自 TP1/EE の定義内容が妥当ではない場合，プロセスダウンします。
6. XDB の初期化
   実行系と通信を行い，実行系の状態を取得します。取得した内容を基に，XDB のデータベースを作成します。
7. 初期化トランザクション（MI）の起動
   初期化トランザクション（MI）を起動します。このとき，待機系での起動であることをユーザに通知するため，トランザクションインタフェース情報の系情報に EERPC_SYSTEM_STANDBY（待機系を意味する値）を設定して起動します。
8. 起動完了通知送信
   起動が完了したことを実行系に通知します。
9. HA モニタ待機状態通知
   HA モニタに対して待機状態通知を行います。以降，自 TP1/EE は系切り替え対象となります。
10. 開始処理後の処理
    開始処理後の処理として，待機系としてのオンライン処理を行います。

参考

実行系が起動していない状態で待機系が起動すると，HA モニタ関連定義の ham_wait_interval_time オペランドに従って，実行系が起動するまで KFSB55901-E メッセージが出力されます。

CL サーバでの XTC の開始時に発生する障害については，「8.1.1　XTC の開始時に発生する障害」を参照してください。

6.2 XTC の終了

ここでは，XTC の終了方法，終了モード，および終了時のシステムの処理について説明します。

6.2.1 HA サーバでの XTC の終了

HA サーバの場合，終了方法，終了モード，および終了時のシステムの処理は TP1 キャッシュ機能を使用しない場合と同様です。詳細については，マニュアル「TP1/Server Base Enterprise Option 使用の手引」を参照してください。

6.2.2 CL サーバでの XTC の終了

CL サーバの場合の終了方法，終了モード，および終了時のシステムの処理について説明します。なお，XTC を終了すると，その延長で XDB も終了されます。

(1) 終了方法

CL サーバの場合，次のどちらかの方法で XTC を終了します。

- 実行系の TP1/EE に対して eesvstop コマンドを実行する。
- 実行系が起動している TP1/Server Base に対して dcstop コマンドを実行する。

実行系を終了させることで，すべての待機系を連動して終了させます。

系切り替え中に終了コマンドを実行すると，系切り替え後に終了処理を開始します。ただし，終了モードに強制停止を指定した場合は，系切り替え中であっても終了します。

なお，通信障害などによって TP1/EE が待機系を切り離した場合，HA モニタは切り離しを検知しません。この場合，切り離された待機系を次のどちらかの方法で終了させてください。

- 待機系の TP1/EE に対して eesvstop コマンドを実行する。
- 待機系が起動している TP1/Server Base に対して dcstop コマンドを実行する。

切り離されていない待機系だけを終了する場合，系切り替えのタイミングによっては，実行系，およびすべての待機系が終了するおそれがあります。

(2) 終了モード

XTC の終了モードには次の四つがあります。

- 正常終了
- 計画停止 A
- 計画停止 B
- 強制停止

各終了モードの意味については，マニュアル「OpenTP1 運用と操作」を参照してください。ここでは，TP1 キャッシュ機能使用時の処理について説明します。

(a) eesvstop コマンドで終了する場合

終了モードは，eesvstop コマンドのオプションの指定で決定します。ただし，強制停止（eesvstop コマンドに -f オプションを指定）はできません。

(b) dcstop コマンドで終了する場合

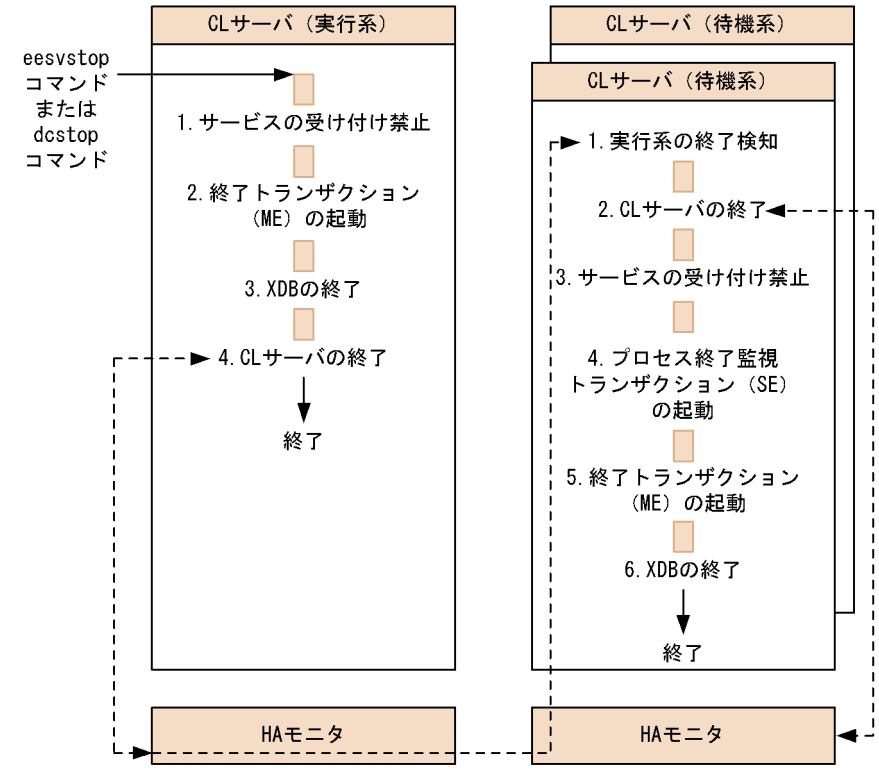
終了モードは，dcstop コマンドのオプションの指定で決定します。

強制停止の場合，実行中のサービス要求の完了を待たないで終了します。処理キュー上の受け付け済みのサービス要求はすべて破棄します。また，XDB の内容は永続化されません。

(3) 終了時のシステムの処理

終了時のシステムの処理を次の図に示します。

図 6-3 CL サーバでの XTC の終了



実行系の処理の説明

1. サービスの受け付け禁止
   新たなサービスの受け付けを禁止します。
2. 終了トランザクション（ME）の起動
   終了トランザクション（ME）を起動します。このとき，実行系での起動であることをユーザに通知するため，トランザクションインタフェース情報の系情報にEERPC_SYSTEM_ONLINE（実行系を意味する値）を設定して起動します。
3. XDB の終了
   XDB の終了処理を行います。
4. CL サーバの終了
   HA モニタや CL サーバに関する終了処理を行います。この処理以降は，系切り替えが発生しません。

待機系の処理の説明

1. 実行系の終了検知

HA モニタ経由で，実行系が終了したことを検知します。

2. CL サーバの終了
   HA モニタや CL サーバに関する終了処理を行います。この処理以降は，系切り替えが発生しません。
3. サービスの受け付け禁止
   新たなサービスの受け付けを禁止します。
4. プロセス終了監視トランザクション（SE）の起動
   プロセス終了監視トランザクション（SE）を起動します。
5. 終了トランザクション（ME）の起動
   終了トランザクション（ME）を起動します。このとき，待機系での起動であることをユーザに通知するため，トランザクションインタフェース情報の系情報に EERPC_SYSTEM_STANDBY（待機系を意味する値）を設定して起動します。
6. XDB の終了
   XDB の終了処理を行います。

参考

待機系に対して終了コマンドを実行する場合も，処理の流れは同様です。

XTC の終了時に CL サーバで発生する障害については，「8.1.2　XTC の終了時に発生する障害」を参照してください。

6.3 ファイルの運用

ここでは，XTC 固有のファイルの運用について説明します。

ここに記載のないファイルの運用については，マニュアル「TP1/Server Base Enterprise Option 使用の手引」のファイルの運用の説明を参照してください。

6.3.1 回線トレースファイルの運用

ここでは，回線トレースファイルの運用で，XTC 固有の部分についてだけ説明します。

(1) 回線トレース情報の取得量の変更

送受信データごとに回線トレース情報を取得する場合，I/O 処理の増加によって性能が低下したり，重要な回線トレース情報が欠落したりすることがあります。これを防止するために，RPC 関連定義の rpc_udp_linetrace オペランドを指定することで，送受信時の回線トレース情報の取得内容を変更できます。

ここでは，rpc_udp_linetrace オペランドの指定値別の取得内容，取得サイズの差異，およびオペランドの指定値の組み合わせについて説明します。

(a) 取得内容

メッセージ送受信が正常終了した場合の取得内容を，rpc_udp_linetrace オペランドの指定値によって変更することができます。オペランドの指定値別に，回線トレース情報の取得内容について説明します。

00000000（デフォルト）

すべての回線トレース情報を，送信側と受信側で取得します。
送受信処理が正常に完了した場合は，最大 374 バイト取得します。

00000001

W-send 機能や再送などによって重複して受信したパケットについて，回線トレース情報を取得しません。
回線トレース情報は，受信側で取得します。

00000002

次のすべての条件を満たす場合に，回線トレース情報を取得しません。

- myudpsnddef 定義コマンドの -k オプションを指定している
- クラスタ内でのマルチキャストによって自 TP1/EE が送信したメッセージを，自 TP1/EE で受信した

回線トレース情報は，受信側で取得します。

00000004

メッセージが複数のパケットに分割されている場合，先頭パケットは UDP 通信制御

ヘッダおよびデータ部分を最大 374 バイト取得します。残りのパケットは UDP 通信制御ヘッダだけを取得します。
回線トレース情報は送信側と受信側の両方で取得します。

00000020
送受信データの先頭にある製品ヘッダを回線トレース情報として送信側と受信側で取得します。
取得する製品ヘッダはメッセージ種別によって異なります。それぞれの場合について，次に示します。

メッセージ種別が ACK の場合
UDP 通信制御ヘッダだけを取得します。

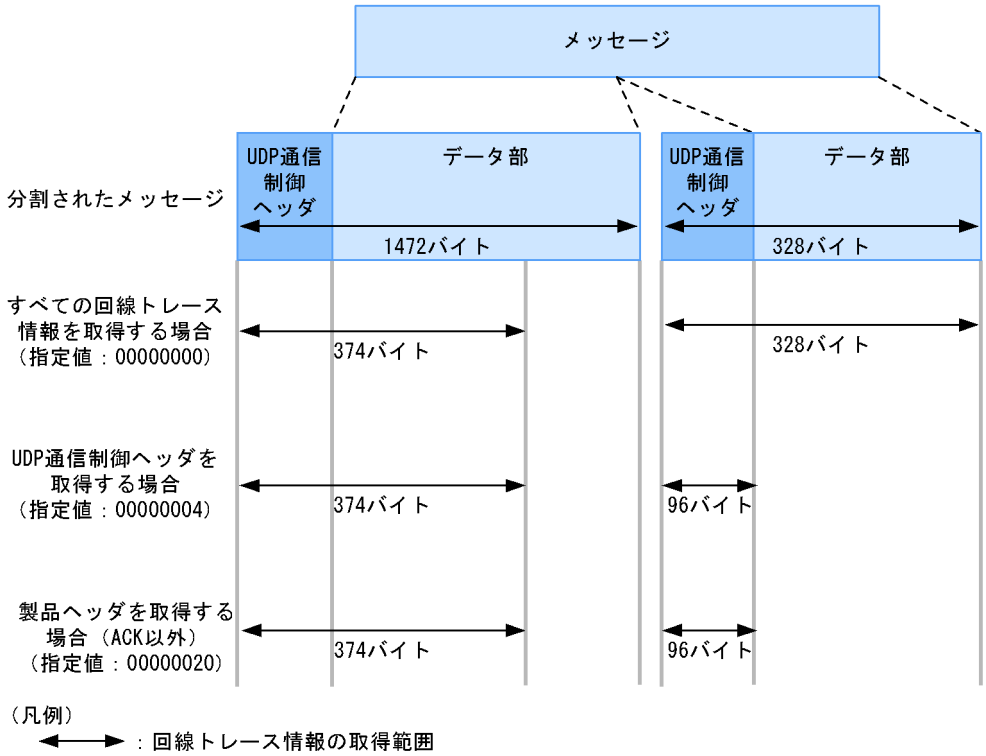
メッセージ種別が ACK 以外の場合
UDP 通信制御ヘッダおよびデータ部分を最大 374 バイト取得します。

メッセージ種別が ACK 以外のメッセージが複数のパケットに分割されている場合
先頭パケットは UDP 通信制御ヘッダおよびデータ部分を最大 374 バイト取得し，残りのパケットは UDP 通信制御ヘッダだけを取得します。

（b）取得サイズの差異

回線トレース情報の取得サイズの差異について，次の図に示します。この例では，UDP 通信機能を使用した RPC のメッセージを 2 回に分けて受信したため，メッセージが二つ（1472 バイトと 328 バイト）に分割されています。

図 6-4 回線トレース情報の取得サイズの差異



(c) rpc_udp_linetrace オペランドの指定値の組み合わせ

rpc_udp_linetrace オペランドの指定値は，論理和を使用して複数選択できます。複数選択する場合に可能な組み合わせを次の表に示します。

表 6-4 rpc_udp_linetrace オペランドの指定値の組み合わせ

指定値	組み合わせる指定値				
	00000000	00000001	00000002	00000004	00000020
00000000	－	◎	◎	－	－
00000001	○	－	◎	○	○
00000002	○	◎	－	○	○
00000004	－	◎	◎	－	－
00000020	－	◎	◎	－	－

（凡例）

◎：組み合わせて指定できます。

○：どれか一つを組み合わせて指定できます。

－：組み合わせて指定できません。

6.4 運用コマンド

ここでは，XTC で使用する運用コマンドについて説明します。運用コマンドの入力方法や記述形式などは，マニュアル「TP1/Server Base Enterprise Option 使用の手引」を参照してください。

運用コマンドの一覧

XTC の運用コマンドの一覧を次の表に示します。

表 6-5 XTC の運用コマンド一覧

項番	機能		コマンド	オフライン	オンライン	API	実行系	待機系
1	システム制御	サービスの閉塞	eedctsv ※	×	○	○	○	×
2		サービスの閉塞解除	eeactsv ※	×	○	○	○	×
3		処理キュー制御用バッファの状態表示およびサービスの最大同時処理限界数の変更	eelspce ※	×	○	○	○	×
4	高速メッセージ送信制御	出力キューの閉塞	eemchotqdct	×	○	○	○	×
5		出力キューの閉塞解除	eemchotqact	×	○	○	○	×
6		出力キューの未送信メッセージのスキップ	eemchotqskip	×	○	○	○	×
7		出力キューの状態表示	eemchotqls	×	○	○	○	○
8		出力キューの未送信メッセージの監視終了	eemchotqend	×	○	×	○	×
9	入力キュー（ITQ）制御	滞留メッセージのファイル出力	eepcerefer	×	○	×	○	×
10		処理キュー制御用バッファの状態表示（通番付き）	eelspcenum	×	○	○	○	×
11		滞留メッセージのスキップ	eepceskip	×	○	○	○	×
12	キューダンプ制御	キューダンプファイルの編集	eetrbqueed	○	○	○	○	○
13	系切り替え	系の状態表示	eehamls	×	○	○	○	○
14	トラブルシュート	メッセージへの応答	eetrbwtor ※	×	○	○	○	○

（凡例）

オフライン：TP1/EE が起動していない状態での実行を示します。
オンライン：TP1/EE が起動している状態での実行を示します。
API：ee_adm_call_command 関数での実行を示します。
実行系：CL サーバの実行系での実行を示します。
待機系：CL サーバの待機系での実行を示します。
○：実行できます。
×：実行できません。

注※

TP1/EE の運用コマンドです。このマニュアルでは，TP1 キャッシュ機能使用時の差分だけを説明します。マニュアル「TP1/Server Base Enterprise Option 使用の手引」もあわせて参照してください。

eeactsv（サービスの閉塞解除）

TP1/EE の運用コマンドです。

TP1 キャッシュ機能を使用する場合，注意事項が追加になります。

機能

サービスの閉塞を解除します。

注意事項

ユーザサービス関連定義の forbid_draw_service オペランドに Y を指定した場合は，該当するサービスの処理キューの引き出し禁止を解除します。

eedctsv（サービスの閉塞）

TP1/EE の運用コマンドです。

TP1 キャッシュ機能を使用する場合，注意事項が追加になります。

機能

サービスを閉塞します。

注意事項

ユーザサービス関連定義の forbid_draw_service オペランドに Y を指定した場合は，該当するサービスの処理キューが引き出し禁止となります。

eehamls（系の状態表示）

機能

CL サーバの実行系および待機系の状態を表示します。

系切り替えできる状態かを確認できます。

コマンドの形式

```
eehamls -g サービスグループ名
```

オプションの説明

● -g サービスグループ名

～〈1 ～ 31 文字の識別子〉

TP1/EE プロセスのサービスグループ名を指定します。

出力形式

eehamls コマンドの出力形式を次の図に示します。

ONL の系が 1 つ，SBY の系が 2 つ表示された場合は，系切り替えできる状態です。

図 6-5　eehamls コマンドの出力形式

■日本語の場合

```
  ホスト名        IPアドレス          ノード識別子    状態
a bb....bb        ccc.ccc.ccc.ccc     dddd            ee....ee
    :
```

■英語の場合

```
  Host name       IP address          node ID         Status
a bb....bb        ccc.ccc.ccc.ccc     dddd            ee....ee
    :
```

変数の説明

変数	意味
a	コマンドを実行した系に * が表示されます。ほかの系には何も表示されません。
bb....bb	ホスト名 HA モニタの environment 定義文の name オペランドで指定した名称が表示されます。

変数	意味
ccc.ccc.ccc.ccc	IP アドレス
dddd	ノード識別子
ee....ee	系の状態 • ONL：実行系として稼働中 • SBY：待機系として稼働中 • (ONL)：実行系として起動処理中 • (SBY)：待機系として起動処理中 • ***：クラスタグループから切り離されている状態 • ONLY：CL 単独起動機能を使用中

出力メッセージ

メッセージ ID	内容	出力先
KFSB95900-E	メモリを確保できません。	標準エラー出力
KFSB95901-E	コマンドの形式が不正です。	標準エラー出力
KFSB95902-E	オプションが不正です。	標準エラー出力
KFSB95921-I	ヘルプメッセージ	標準出力
KFSB95950-E	コマンドが失敗しました（TP1/EE プロセスでエラーを検知しました）。	標準エラー出力
KFSB95951-E	コマンドが失敗しました（コマンドプロセスでエラーを検知しました）。	標準エラー出力

注意事項

eehamls コマンドに関する注意事項を次に示します。

- 実行系と待機系の間で通信障害が発生した場合，待機系はクラスタグループから切り離されます。この状態の場合，実行系の eehamls コマンドと，HA モニタの monshow コマンドでは表示結果が異なります。
  実行系の eehamls コマンドでは，切り離された系は "***" が表示されます。
  HA モニタの monshow コマンドでは "SBY" が表示されます。
- 待機系でこのコマンドを実行しても最新の状態が表示されません。そのため，このコマンドは，実行系で実行してください。
- CL 単独機能を使用する場合，ホスト名は「*」が 32 個，IP アドレスは「0.0.0.0」が表示されます。

eelspce（処理キュー制御用バッファの状態表示およびサービスの最大同時処理限界数の変更）

TP1/EE の運用コマンドです。

TP1 キャッシュ機能を使用する場合，注意事項が追加になります。

機能

処理キュー制御用バッファ（PCE）の滞留状態や処理状態を表示したり，サービスの最大同時処理限界数を変更したりします。

注意事項

最大同時処理限界数の変更は，ユーザサービス関連定義の service_attr 定義コマンドで，-e オプションに parallel を指定するか，または -e オプションを省略したサービスに対してだけ実行できます。

eelspcenum（処理キュー制御用バッファの状態表示（通番付き））

機能

処理キュー制御用バッファ（PCE）の滞留状態，処理状態，およびメッセージ通番を表示します。

コマンドの形式

```
eelspcenum -g サービスグループ名
          〔-v サービス名〕
```

オプションの説明

● -g **サービスグループ名**

～〈1～31 文字の識別子〉

TP1/EE プロセスのサービスグループ名を指定します。

● -v **サービス名**

～〈1～31 文字の識別子〉

滞留状態を表示するサービス名を指定します。

指定を省略した場合，すべてのサービスの滞留状態を表示します。

出力形式

eelspcenum コマンドの出力形式を次の図に示します。

図 6-6　eelspcenum **コマンドの出力形式**

■日本語の場合

```
種別      滞留数     最大数  処理数  書込み通番     読出し通番     サービス名
 a        bb....bb   ccc     ddd     ee....ee       ff....ff       vv....vv
   :
```

■英語の場合

```
 Type     Retain     Max     In proc  Write num      Read num       Service name
 a        bb....bb   ccc     ddd      ee....ee       ff....ff         vv....vv
   :
```

変数の説明

変数	意味
a	キューの種別 • p：優先キュー • n：通常キュー
bb....bb	滞留メッセージ数（最大 10 けた）
ccc	同時に処理できる処理キューの最大数（最大 3 けた）
ddd	処理中の処理キューの数（最大 3 けた）
ee....ee	書き込み通番（最大 10 けた）
ff....ff	読み出し通番（最大 10 けた）
vv....vv	サービス名（最大 31 文字）

出力メッセージ

メッセージ ID	内容	出力先
KFSB95301-E	コマンドの形式が不正です。	標準エラー出力
KFSB95302-E	フラグ引数が不正です。	標準エラー出力
KFSB95304-E	オプションの組み合わせが不正です。	標準エラー出力
KFSB95305-E	コマンドが失敗しました。	標準エラー出力
KFSB95306-E	領域の確保に失敗しました。	標準エラー出力
KFSB95307-E	内部矛盾が発生しました。	標準エラー出力
KFSB95372-I	ヘルプメッセージ	標準出力

注意事項

このコマンドを頻繁に使用すると，スループットが悪化するおそれがあります。

eemchotqact（出力キューの閉塞解除）

機能

出力キュー（OTQ）の閉塞を解除します。オプションによって次の指定ができます。

- システムレベルでの出力キューの閉塞解除
  すべての出力キューの閉塞を解除します。
- サービスグループレベルでの出力キューの閉塞解除
  指定したサービスグループ下の全サービスについて，出力キューの閉塞を解除します。
- サービスレベルでの出力キューの閉塞解除
  指定したサービスグループ下のサービスについて，出力キューの閉塞を解除します。

サービスグループレベルでの出力キューの閉塞解除の場合，閉塞解除対象のサービスグループと通信して，閉塞解除ができるかどうかをチェックしたあとで出力キューの閉塞を解除します。

コマンドの形式

```
eemchotqact -g サービスグループ名
        〔{ -m サービスグループ名 -v サービス名｜-m サービスグループ名}〕
```

オプションの説明

● -g サービスグループ名

～〈1 ～ 31 文字の識別子〉

TP1/EE プロセスのサービスグループ名を指定します。

● -m サービスグループ名

～〈1 ～ 31 文字の識別子〉

閉塞解除の対象とするサービスグループ名を指定します。
サービスグループ名を省略すると，システムレベルでの出力キューの閉塞解除となります。

● -v サービス名

～〈1 ～ 31 文字の識別子〉

閉塞解除の対象とするサービス名を指定します。
サービス名を指定すると，サービスレベルでの出力キューの閉塞解除となります。
サービス名を省略すると，サービスグループレベルでの出力キューの閉塞解除となります。

出力メッセージ

メッセージ ID	内容	出力先
KFSB98001-E	コマンドの形式が不正です。	標準エラー出力
KFSB98002-E	フラグ引数が不正です。	標準エラー出力
KFSB98004-E	オプションの組み合わせが不正です。	標準エラー出力
KFSB98005-E	コマンドが失敗しました。	標準エラー出力
KFSB98006-E	領域の確保に失敗しました。	標準エラー出力
KFSB98023-I	システムレベルで出力キューの閉塞を解除しました。	標準出力
KFSB98024-I	サービスグループレベルで出力キューの閉塞を解除しました。	標準出力
KFSB98025-I	サービスレベルで出力キューの閉塞を解除しました。	標準出力
KFSB98032-E	出力キューの閉塞解除に失敗しました。	標準エラー出力
KFSB98033-E	一部のサービスグループで出力キューの閉塞解除に失敗しました。	標準エラー出力
KFSB98051-I	ヘルプメッセージ	標準出力
KFSB98099-E	内部矛盾が発生しました。	標準エラー出力

注意事項

eemchotqact コマンドに関する注意事項を次に示します。

- サービスグループレベルで出力キューが閉塞中（出力キューの自動閉塞中を含む）の場合，サービスレベルでの出力キュー閉塞解除はできません。
- サービスグループレベルでの出力キューの自動閉塞中に，サービスグループレベルでの出力キューの閉塞解除要求を行った場合，出力キューの閉塞監視を終了します。
- CL サーバの実行系で，システムレベルでの出力キューの閉塞解除要求を行った場合，KFSB98033-E メッセージが表示されると，実行系と待機系で結果が異なっています。eemchotqls コマンドで実行系と待機系の出力キューの状態を確認してから，eemchotqdct コマンドを待機系で実行して，実行系の出力キューの状態と一致させてください。

eemchotqdct（出力キューの閉塞）

機能

出力キュー（OTQ）を閉塞します。オプションによって次の指定ができます。

- システムレベルでの出力キューの閉塞
  すべての出力キューを閉塞します。
- サービスグループレベルでの出力キューの閉塞
  指定したサービスグループ下の全サービスについて，出力キューを閉塞します。
- サービスレベルでの出力キューの閉塞
  指定したサービスグループ下のサービスについて，出力キューを閉塞します。

コマンドの形式

```
eemchotqdct -g サービスグループ名
        〔{ -m サービスグループ名 -v サービス名｜-m サービスグループ名} 〕
```

オプションの説明

● -g サービスグループ名

～〈1 ～ 31 文字の識別子〉

TP1/EE プロセスのサービスグループ名を指定します。

● -m サービスグループ名

～〈1 ～ 31 文字の識別子〉

閉塞の対象とするサービスグループ名を指定します。
サービスグループ名を省略した場合，システムレベルでの出力キューの閉塞となります。

● -v サービス名

～〈1 ～ 31 文字の識別子〉

閉塞の対象とするサービス名を指定します。
サービス名を指定すると，サービスレベルでの出力キューの閉塞となります。
サービス名を省略すると，サービスグループレベルでの出力キューの閉塞となります。

出力メッセージ

メッセージ ID	内容	出力先
KFSB98001-E	コマンドの形式が不正です。	標準エラー出力
KFSB98002-E	フラグ引数が不正です。	標準エラー出力
KFSB98004-E	オプションの組み合わせが不正です。	標準エラー出力

メッセージ ID	内容	出力先
KFSB98005-E	コマンドが失敗しました。	標準エラー出力
KFSB98006-E	領域の確保に失敗しました。	標準エラー出力
KFSB98020-I	システムレベルで出力キューを閉塞しました。	標準出力
KFSB98021-I	サービスグループレベルで出力キューを閉塞しました。	標準出力
KFSB98022-I	サービスレベルで出力キューを閉塞しました。	標準出力
KFSB98030-E	出力キューの閉塞に失敗しました。	標準エラー出力
KFSB98050-I	ヘルプメッセージ	標準出力
KFSB98099-E	内部矛盾が発生しました。	標準エラー出力

注意事項

サービスグループレベルでの出力キューの自動閉塞中に，サービスグループレベルでの出力キューの閉塞要求を行った場合，出力キューの閉塞監視を終了します。サービスレベルでの出力キューの閉塞要求を行った場合は，出力キューの閉塞監視は続行します。

eemchotqend（出力キューの未送信メッセージの監視終了）

機能

出力キュー（OTQ）の未送信メッセージの監視を終了します。出力キューに残った未送信メッセージは，キューダンプファイルに出力して終了します。

このコマンドは，出力キューに未送信メッセージが残っていて，オンライン終了ができない場合に使用します。

コマンドの形式

```
eemchotqend -g サービスグループ名
```

オプションの説明

● -g **サービスグループ名**

～〈1 ～ 31 文字の識別子〉
TP1/EE プロセスのサービスグループ名を指定します。

出力メッセージ

メッセージ ID	内容	出力先
KFSB98001-E	コマンドの形式が不正です。	標準エラー出力
KFSB98002-E	フラグ引数が不正です。	標準エラー出力
KFSB98005-E	コマンドが失敗しました。	標準エラー出力
KFSB98006-E	領域の確保に失敗しました。	標準エラー出力
KFSB98054-I	ヘルプメッセージ	標準出力
KFSB98099-E	内部矛盾が発生しました。	標準エラー出力

eemchotqls（出力キューの状態表示）

機能

出力キュー（OTQ）の状態を表示します。

コマンドの形式

```
eemchotqls -g サービスグループ名
          〔-m サービスグループ名〕
           -s {act｜dct｜all}
```

オプションの説明

● -g **サービスグループ名**

～〈1 ～ 31 文字の識別子〉

TP1/EE プロセスのサービスグループ名を指定します。

● -m **サービスグループ名**

～〈1 ～ 31 文字の識別子〉

出力キューの状態を表示するサービスグループ名を指定します。
このオプションを省略した場合，すべてのサービスグループが対象となります。

● -s {act ｜ dct ｜ all}

表示する出力キューの状態を指定します。

act：
閉塞していないサービス名をすべて表示します。

dct：
閉塞しているサービス名をすべて表示します。

all：
すべてのサービス名を表示します。

出力形式

eemchotqls コマンドの出力形式を次の図に示します。

図 6-7 eemchotqls **コマンドの出力形式**

■日本語の場合

```
サービスグループ名   サービス名 状態      書込み通番 送信済通番 未送信数
aa....aa             bb....bb  xxx(y)    cc....cc   dd....dd   ee....ee
    :
```

■英語の場合

```
Service group name  Service name Status   Write num  Sent num   Unsent cnt
aa....aa            bb....bb     xxx(y)   cc....cc   dd....dd   ee....ee
     :
```

変数の説明

変数	意味
aa....aa	サービスグループ名
bb....bb	サービス名
cc....cc	メッセージ送信要求済み出力キューの通番（10 進数） メッセージ送信要求がない場合は 0 が表示されます。
dd....dd	メッセージ送信済み出力キューの通番（10 進数） メッセージ送信要求がない場合は 0 が表示されます。
ee....ee	未送信メッセージ数（10 進数）
xxx	出力キューの状態 • ACT：閉塞解除状態 • DCT：自動閉塞中 • DCC：コマンドによる閉塞中 • DCA：API 関数による閉塞中
y	閉塞レベル • G：サービスグループレベル • S：サービスレベル • *：閉塞解除状態

出力対象のサービス名がない場合は，「対象のサービスはありません。」が表示されます。

出力メッセージ

メッセージ ID	内容	出力先
KFSB98001-E	コマンドの形式が不正です。	標準エラー出力
KFSB98002-E	フラグ引数が不正です。	標準エラー出力
KFSB98004-E	オプションの組み合わせが不正です。	標準エラー出力
KFSB98005-E	コマンドが失敗しました。	標準エラー出力
KFSB98006-E	領域の確保に失敗しました。	標準エラー出力
KFSB98053-I	ヘルプメッセージ	標準出力
KFSB98099-E	内部矛盾が発生しました。	標準エラー出力

注意事項

CL サーバの場合，待機系では実行系よりも遅れて出力キューの状態が更新されます。

eemchotqskip（出力キューの未送信メッセージのスキップ）

機能

指定したサービスグループ下のサービス名に対応する出力キューの未送信のメッセージ（送信に失敗したメッセージを含む）をスキップして，送信済みにします。

eemchotqskip コマンドは，該当する出力キューが閉塞中の場合にだけ実行できます。

コマンドの形式

```
eemchotqskip -g サービスグループ名
             -m サービスグループ名
             -v サービス名
             -s {all｜one｜num}
            〔-n 出力キュー通番〕
```

オプションの説明

● -g **サービスグループ名**

～〈1 ～ 31 文字の識別子〉

TP1/EE プロセスのサービスグループ名を指定します。

● -m **サービスグループ名**

～〈1 ～ 31 文字の識別子〉

スキップ対象とする出力キューのサービスグループ名を指定します。

● -v **サービス名**

～〈1 ～ 31 文字の識別子〉

スキップ対象とする出力キューのサービス名を指定します。

● -s {all ｜ one ｜ num}

送信済みにするメッセージを指定します。

all：

未送信メッセージをすべてスキップし，送信済みにします。

one：

未送信の先頭 1 メッセージをスキップし，送信済みにします。

num：

-n オプションで指定された出力キュー通番までの未送信メッセージをスキップし，送信済みにします。

● -n 出力キュー通番

〜 ((1 〜 4294967295))
送信済みにする出力キューの出力キュー通番を指定します。
指定した出力キュー通番と一致する未送信メッセージがある場合は，指定した通番までのメッセージをスキップし，送信済みにします。指定した出力キュー通番と一致する未送信メッセージがない場合は，メッセージをスキップしないで正常終了します。
このオプションは，-s オプションで num を指定した場合だけ指定できます。

出力メッセージ

メッセージ ID	内容	出力先
KFSB98001-E	コマンドの形式が不正です。	標準エラー出力
KFSB98002-E	フラグ引数が不正です。	標準エラー出力
KFSB98004-E	オプションの組み合わせが不正です。	標準エラー出力
KFSB98005-E	コマンドが失敗しました。	標準エラー出力
KFSB98006-E	領域の確保に失敗しました。	標準エラー出力
KFSB98026-I	未送信メッセージをスキップしました。	標準出力
KFSB98034-E	未送信メッセージのスキップに失敗しました。	標準エラー出力
KFSB98052-I	ヘルプメッセージ	標準出力
KFSB98099-E	内部矛盾が発生しました。	標準エラー出力

注意事項

eemchotqskip コマンドに関する注意事項を次に示します。

- CL サーバの待機系では，このコマンドは実行できません。
- 送信スキップ対象のメッセージがない場合は，正常終了します。

eepcerefer（滞留メッセージのファイル出力）

機能

引き出し禁止状態のサービスの入力キュー（ITQ）に滞留しているメッセージを，指定したファイルに出力して参照できるようにします。

出力対象となるのは，一つのサービスの通常キューまたは優先キューのどちらかのキューに滞留しているすべてのメッセージです。

ファイルへの出力は，TP1/EE プロセスで行います。eepcerefer コマンドは，TP1/EE プロセスが出力要求を受け付けた時点で終了します。TP1/EE プロセスは，ファイル出力終了時に KFSB85407-I メッセージを出力します。

コマンドの形式

```
eepcerefer -g サービスグループ名
           -v サービス名
           -p {p|n}
           -o 出力ファイル名
          〔-r〕
```

オプションの説明

● -g **サービスグループ名**

～〈1 ～ 31 文字の識別子〉
TP1/EE プロセスのサービスグループ名を指定します。

● -v **サービス名**

～〈1 ～ 31 文字の識別子〉
滞留メッセージを出力するサービス名を指定します。

● -p {p | n}

優先キュー，通常キューのどちらの滞留メッセージを出力するかを指定します。

p：
　優先キュー

n：
　通常キュー

● -o **出力ファイル名**

～〈510 文字以内のパス名〉
滞留しているメッセージを出力するファイル名を絶対パス名で指定します。ファイルの容量は，次に示す計算式で見積もってください。

滞留メッセージ数×（メッセージ長×5＋200）＋2000（単位：バイト）

● -r

このオプションを指定すると，-o オプションで指定したファイルがすでにある場合にファイルを上書きします。上書きした場合，元のファイルの内容は失われます。
このオプションを指定する場合は，必要に応じて手動でバックアップを取得してください。
このオプションを省略すると，-o オプションで指定したファイルがすでにある場合は，ファイル出力を中止します。

出力形式

eepcerefer コマンドの出力形式を次の図に示します。

図 6-8　eepcerefer コマンドの出力形式

■日本語の場合

```
*******    滞留メッセージ情報  バージョン(TP1/EE :aa-aa-aa)  *******
*******                        (TP1/XTC:bb-bb-bb)  *******
サービスグループ名：cc....cc
ランID：0xddddddddd
サービス名:ee....ee
種別:f

メッセージ通番:gg....gg
メッセージ種別:hhh
メッセージ長:ll....ll
   xxxxxxxx    yyyyyyyy yyyyyyyy yyyyyyyy yyyyyyyy  zzzzzzzzzzzzzzzz
   xxxxxxxx    yyyyyyyy yyyyyyyy yyyyyyyy yyyyyyyy  zzzzzzzzzzzzzzzz
   xxxxxxxx    TO NEXT LINE LOCATION SAME AS ABOVE
   xxxxxxxx    yyyyyyyy yyyyyyyy yyyyyyyy yyyyyyyy  zzzzzzzzzzzzzzzz
```

※

■英語の場合

```
*******    Remaining message information  Version(TP1/EE :aa-aa-aa)  *******
*******                                          (TP1/XTC:bb-bb-bb)  *******
Service group name : cc....cc
Run ID : 0xddddddddd
Service name:ee....ee
Type:f

Message number:gg....gg
Message type:hhh
Message length:ll....ll
   xxxxxxxx    yyyyyyyy yyyyyyyy yyyyyyyy yyyyyyyy  zzzzzzzzzzzzzzzz
   xxxxxxxx    yyyyyyyy yyyyyyyy yyyyyyyy yyyyyyyy  zzzzzzzzzzzzzzzz
   xxxxxxxx    TO NEXT LINE LOCATION SAME AS ABOVE
   xxxxxxxx    yyyyyyyy yyyyyyyy yyyyyyyy yyyyyyyy  zzzzzzzzzzzzzzzz
```

※

注※

滞留メッセージの数だけ繰り返します。

変数の説明

変数	意味
aa-aa-aa	コマンドを実行した TP1/EE のバージョン
bb-bb-bb	コマンドを実行した XTC のバージョン
cc....cc	サービスグループ名（最大 31 文字）
ddddddddd	TP1/EE のラン ID（16 進数，8 けた）
ee....ee	サービス名（最大 31 文字）
f	キューの種別（1 文字） • p：優先キュー • n：通常キュー
gg....gg	メッセージ通番（10 進数，最大 10 けた）
hhh	メッセージ種別 • RPC：RPC メッセージ • MCH：MCH メッセージ • MCP：MCP メッセージ • TIM：タイマトランザクション • RLn：MCP 後処理トランザクション RLO：コネクション確立 RLC：コネクション解放 RLF：論理端末閉塞解除 RLH：論理端末閉塞 • MV：イベントトランザクション
ll....ll	メッセージ長（10 進数，最大 10 けた） メッセージ長が 0 の場合は，以降の項目は出力しません。

変数	意味
xxxxxxxx	メッセージの先頭からのロケーション（16 進数，8 けた）
yyyyyyyy	メッセージの内容（16 進数，最大 8 けた） 1 行前に出力した行と同一の場合，次の文字列を出力します。 TO NEXT LINE LOCATION SAME AS ABOVE 複数行同一の場合，次の文字列を一度だけ出力します。 TO NEXT LINE LOCATION SAME AS ABOVE ただし，最終行の 2 行が同一の場合は，「TO NEXT LINE LOCATION SAME AS ABOVE」を出力しません。
zzzzzzzzzzzzzzzz	メッセージの内容（ASCII 文字で最大 16 文字） 表示できない ASCII 文字の場合は「.」を出力します。

出力メッセージ

メッセージ ID	内容	出力先
KFSB95301-E	コマンドの形式が不正です。	標準エラー出力
KFSB95302-E	フラグ引数が不正です。	標準エラー出力
KFSB95304-E	オプションの組み合わせが不正です。	標準エラー出力
KFSB95305-E	コマンドが失敗しました。	標準エラー出力
KFSB95306-E	領域の確保に失敗しました。	標準エラー出力
KFSB95307-E	内部矛盾が発生しました。	標準エラー出力
KFSB95378-I	ヘルプメッセージ	標準出力
KFSB95379-I	コマンドを受け付けました。	標準出力
KFSB95380-E	コマンドを受け付けませんでした。	標準エラー出力

注意事項

eepcerefer コマンドに関する注意事項を次に示します。

- このコマンドを頻繁に使用すると，スループットが悪化するおそれがあります。
- 引き出し禁止状態ではないサービスの滞留メッセージは出力できません。
- 滞留メッセージのファイル出力中に引き出し禁止状態を解除した場合は，出力途中のメッセージのファイル出力が終了した時点で，ファイル出力を終了します。後続の滞留メッセージはファイルに出力されません。
- 滞留メッセージのファイル出力中に eepceskip コマンドを実行した場合は，出力途中のメッセージのファイル出力が終了した時点で，ファイル出力を終了します。後続の滞留メッセージはファイルに出力されません。
- 滞留メッセージのファイル出力中は，出力するメッセージのサイズによって 1 キロバイト～ 8 メガバイトのワーク領域，または XTC 用ワーク領域（XTCPOOL）を使用します。

eepceskip（滞留メッセージのスキップ）

機能

入力キュー（ITQ）の滞留メッセージを読み出し済みにして，入力キューから破棄します。

指定したサービス名に対応する入力キューのメッセージを読み出し済みにします。スキップ要求を受け付けるのは，該当するサービスの処理キューが引き出し禁止状態であり，かつ該当するサービスのトランザクションが動作していない場合だけです。

コマンドの形式

```
eepceskip -g サービスグループ名
          -v サービス名
```

オプションの説明

● -g **サービスグループ名**

～〈1 ～ 31 文字の識別子〉
TP1/EE プロセスのサービスグループ名を指定します。

● -v **サービス名**

～〈1 ～ 31 文字の識別子〉
滞留メッセージをスキップするサービス名を指定します。

出力メッセージ

メッセージ ID	内容	出力先
KFSB95301-E	コマンドの形式が不正です。	標準エラー出力
KFSB95302-E	フラグ引数が不正です。	標準エラー出力
KFSB95304-E	オプションの組み合わせが不正です。	標準エラー出力
KFSB95305-E	コマンドが失敗しました。	標準エラー出力
KFSB95306-E	領域の確保に失敗しました。	標準エラー出力
KFSB95307-E	内部矛盾が発生しました。	標準エラー出力
KFSB95375-I	ヘルプメッセージ	標準出力
KFSB95376-I	滞留メッセージのスキップが成功しました。	標準出力
KFSB95377-E	滞留メッセージのスキップが失敗しました。	標準エラー出力

注意事項

スキップ対象のメッセージがない場合は，正常終了します。

eetrbqueed（キューダンプファイルの編集）

機能

キューダンプファイルの内容を編集して出力します。

コマンドの形式

```
eetrbqueed 〔-e {m｜f}〕
           〔-q {i｜o}〕
           〔-g サービスグループ名〕
           〔-v サービス名〕
           〔-o 編集結果出力ファイル名〕
            キューダンプファイル名 〔キューダンプファイル名〕…
```

オプションの説明

● -e {m ｜ f}

～《f》

編集種別を指定します。

m：

メッセージの内容を出力します。

f：

キューダンプファイルの一覧を出力します。

● -q {i ｜ o}

キュー種別を指定します。
このオプションを省略した場合は，入力キュー（ITQ）および出力キュー（OTQ）の両方のメッセージを出力します。

i：

入力キューのメッセージだけ出力します。

o：

出力キューのメッセージだけ出力します。

● -g **サービスグループ名**

～〈1 ～ 31 文字の識別子〉

表示対象とする出力キューのサービスグループ名を指定します。
このオプションを省略した場合は，すべてのサービスグループを対象として表示します。
このオプションは，-q オプションに o を指定した場合にだけ指定できます。

● -v サービス名

～〈1 ～ 31 文字の識別子〉
表示対象とする入力キューまたは出力キューのサービス名を指定します。
このオプションを省略した場合は，すべてのサービスを対象として表示します。
このオプションは，-q オプションを指定した場合にだけ指定できます。

● -o 編集結果出力ファイル名

～〈パス名〉
編集結果を出力するファイル名を相対パス名で指定します。-e オプションに m を指定した場合は，必ずこのオプションを指定してください。-e オプションに f を指定した場合は指定できません。

●キューダンプファイル名 〔キューダンプファイル名〕…

～〈パス名〉
編集するキューダンプファイル名を相対パス名で指定します。-e オプションに f を指定した場合は，キューダンプファイル名を複数指定できます。その場合，キューダンプファイル名は最大 1024 個指定できます。

出力形式

-e オプションに m を指定した場合

eetrbqueed コマンドの出力形式（-e オプションに m を指定した場合）を次の図に示します。

図 6-9　eetrbqueed コマンドの出力形式（-e オプションに m を指定した場合）

■日本語の場合

```
*******    キューダンプ情報     バージョン(TP1/EE :aa-aa-aa)  *******
*******                                (TP1/XTC:bb-bb-bb)  *******
出力指定：cc....cc
ファイル名：dd....dd
ファイル作成日付：eeee/ee/ee ee:ee:ee eee.eee
ファイル作成バージョン：(TP1/EE :ff-ff-ff)
                        (TP1/XTC:gg-gg-gg)
                        (TP1/MCP:g1-g1-g1)
サービスグループ名：hh....hh
ランID：0xiiiiiiii

*******  ITQ  *******                                                    ┐
サービス名：jj....jj                                                     │
メッセージ種別：kkk                                                      │※1
メッセージ長：ll....ll                                                   │
   xxxxxxxx    yyyyyyyy yyyyyyyy yyyyyyyy yyyyyyyy   zzzzzzzzzzzzzzzz  │
   xxxxxxxx    yyyyyyyy yyyyyyyy yyyyyyyy yyyyyyyy   zzzzzzzzzzzzzzzz  │
   xxxxxxxx    TO NEXT LINE LOCATION SAME AS ABOVE                      │
   xxxxxxxx    yyyyyyyy yyyyyyyy yyyyyyyy yyyyyyyy   zzzzzzzzzzzzzzzz  ┘

*******  ITQ(仮登録)  *******                                            ┐
サービス名     ：jj....jj                                                │
メッセージ種別：kkk                                                      │
メッセージ長：ll....ll                                                   │
   xxxxxxxx    yyyyyyyy yyyyyyyy yyyyyyyy yyyyyyyy   zzzzzzzzzzzzzzzz  │
   xxxxxxxx    yyyyyyyy yyyyyyyy yyyyyyyy yyyyyyyy   zzzzzzzzzzzzzzzz  │
   xxxxxxxx    TO NEXT LINE LOCATION SAME AS ABOVE                      │※2
   xxxxxxxx    yyyyyyyy yyyyyyyy yyyyyyyy yyyyyyyy   zzzzzzzzzzzzzzzz  ┐│
送信先サービスグループ名：nn....nn                                      ││
送信先サービス名：oo....oo                                              ││
メッセージ長：ll....ll                                                  ※3│
   xxxxxxxx    yyyyyyyy yyyyyyyy yyyyyyyy yyyyyyyy   zzzzzzzzzzzzzzzz  ││
   xxxxxxxx    yyyyyyyy yyyyyyyy yyyyyyyy yyyyyyyy   zzzzzzzzzzzzzzzz  ││
   xxxxxxxx    TO NEXT LINE LOCATION SAME AS ABOVE                      ┘│
   xxxxxxxx    yyyyyyyy yyyyyyyy yyyyyyyy yyyyyyyy   zzzzzzzzzzzzzzzz  ┘

*******  OTQ  *******                                                    ┐
送信先サービスグループ名：nn....nn                                      │
送信先サービス名：oo....oo                                              │※4
メッセージ長：ll....ll                                                   │
   xxxxxxxx    yyyyyyyy yyyyyyyy yyyyyyyy yyyyyyyy   zzzzzzzzzzzzzzzz  │
   xxxxxxxx    yyyyyyyy yyyyyyyy yyyyyyyy yyyyyyyy   zzzzzzzzzzzzzzzz  │
   xxxxxxxx    TO NEXT LINE LOCATION SAME AS ABOVE                      │
   xxxxxxxx    yyyyyyyy yyyyyyyy yyyyyyyy yyyyyyyy   zzzzzzzzzzzzzzzz  ┘
```

■英語の場合

```
*******    Queue dump information  Version(TP1/EE :aa-aa-aa)  *******
*******                                   (TP1/XTC:bb-bb-bb)  *******
Specify output : cc....cc
File name : dd....dd
File creation date : eeee/ee/ee ee:ee:ee eee.eee
File creation version : (TP1/EE :ff-ff-ff)
                        (TP1/XTC:gg-gg-gg)
                        (TP1/MCP:g1-g1-g1)
Service group name : hh....hh
Run ID : 0xiiiiiiii

*******  ITQ  *******
Service name : jj....jj
Message type : kkk
Message length : ll....ll
   xxxxxxxx    yyyyyyyy yyyyyyyy yyyyyyyy yyyyyyyy   zzzzzzzzzzzzzzzz
   xxxxxxxx    yyyyyyyy yyyyyyyy yyyyyyyy yyyyyyyy   zzzzzzzzzzzzzzzz
   xxxxxxxx    TO NEXT LINE LOCATION SAME AS ABOVE
   xxxxxxxx    yyyyyyyy yyyyyyyy yyyyyyyy yyyyyyyy   zzzzzzzzzzzzzzzz

*******  ITQ (provisional registration)  *******
Service name : jj....jj
Message type : kkk
Message length : ll....ll
   xxxxxxxx    yyyyyyyy yyyyyyyy yyyyyyyy yyyyyyyy   zzzzzzzzzzzzzzzz
   xxxxxxxx    yyyyyyyy yyyyyyyy yyyyyyyy yyyyyyyy   zzzzzzzzzzzzzzzz
   xxxxxxxx    TO NEXT LINE LOCATION SAME AS ABOVE
   xxxxxxxx    yyyyyyyy yyyyyyyy yyyyyyyy yyyyyyyy   zzzzzzzzzzzzzzzz
Transmission destination service group : nn....nn
Transmission destination service name : oo....oo
Message length : ll....ll
   xxxxxxxx    yyyyyyyy yyyyyyyy yyyyyyyy yyyyyyyy   zzzzzzzzzzzzzzzz
   xxxxxxxx    yyyyyyyy yyyyyyyy yyyyyyyy yyyyyyyy   zzzzzzzzzzzzzzzz
   xxxxxxxx    TO NEXT LINE LOCATION SAME AS ABOVE
   xxxxxxxx    yyyyyyyy yyyyyyyy yyyyyyyy yyyyyyyy   zzzzzzzzzzzzzzzz

*******  OTQ  *******
Transmission destination service group name : nn....nn
Transmission destination service name : oo....oo
Message length : ll....ll
   xxxxxxxx    yyyyyyyy yyyyyyyy yyyyyyyy yyyyyyyy   zzzzzzzzzzzzzzzz
   xxxxxxxx    yyyyyyyy yyyyyyyy yyyyyyyy yyyyyyyy   zzzzzzzzzzzzzzzz
   xxxxxxxx    TO NEXT LINE LOCATION SAME AS ABOVE
   xxxxxxxx    yyyyyyyy yyyyyyyy yyyyyyyy yyyyyyyy   zzzzzzzzzzzzzzzz
```

※1

※2

※3

※4

注※1

入力キューの滞留メッセージ数分繰り返し出力します。

注※2

入力キュー仮登録の滞留メッセージ数分繰り返し出力します。

注※3

入力キュー仮登録の滞留メッセージに対応する送信メッセージ数分繰り返し出力します。

注※ 4

出力キューの送信メッセージ数分繰り返し出力します。

変数の説明

変数	意味
aa-aa-aa	コマンドを実行した TP1/EE のバージョン
bb-bb-bb	コマンドを実行した XTC のバージョン
cc....cc	編集コマンドの引数
dd....dd	キューダンプファイル名
eeee/ee/ee ee:ee:ee eee.eee	ファイルに出力を開始した日付 年号 / 月 / 日 時間：分：秒 ミリ秒．マイクロ秒
ff-ff-ff	ファイルを作成した TP1/EE のバージョン
gg-gg-gg	ファイルを作成した XTC のバージョン
g1-g1-g1	ファイルを作成した MCP のバージョン MCP の受信メッセージを永続化している場合にだけ出力します。
hh....hh	サービスグループ名（最大 31 文字）
iiiiiiii	TP1/EE のラン ID（16 進数，8 けた）
jj....jj	サービス名（最大 31 文字）
kkk	メッセージ種別 • RPC：RPC メッセージ • MCP：MCP メッセージ • MCH：MCH メッセージ
ll....ll	メッセージ長（10 進数，最大 10 けた） メッセージ長が 0 の場合，xxxxxxxx から zzzzzzzzzzzzzzzz の項目は出力しません。
nn....nn	送信先サービスグループ名（最大 31 文字）
oo....oo	送信先サービス名（最大 31 文字）
xxxxxxxx	メッセージの先頭からのロケーション（16 進数，8 けた）
yyyyyyyy	メッセージの内容（16 進数，最大 8 けた） 1 行前に出力した行と同一の場合，次の文字列を出力します。 TO NEXT LINE LOCATION SAME AS ABOVE 複数行同一の場合，次の文字列を一度だけ出力します。 TO NEXT LINE LOCATION SAME AS ABOVE ただし，最終行の 2 行が同一の場合は，「TO NEXT LINE LOCATION SAME AS ABOVE」を出力しません。
zzzzzzzzzzzzzzzz	メッセージの内容（ASCII 文字で最大 16 文字） 表示できない ASCII 文字の場合は「.」を出力します。

-e オプションに f を指定した場合

eetrbqueed コマンドの出力形式（-e オプションに f を指定した場合）を次の図に示します。

図 6-10 eetrbqueed コマンドの出力形式（-e オプションに f を指定した場合）

■日本語の場合

```
*******    キューダンプ情報    バージョン(TP1/EE :aa-aa-aa)  *******
*******                        (TP1/XTC:bb-bb-bb)  *******
出力指定：cc....cc
ファイル名：dd....dd                                                    ┐
ファイル作成日付：eeee/ee/ee ee:ee:ee  eee.eee                          │
ファイル作成バージョン：(TP1/EE :ff-ff-ff)                              │※1
                        (TP1/XTC:gg-gg-gg)                              │
サービスグループ名：hh....hh                                            │
ランID：0xiiiiiiii                                                      ┘

ファイル名：dd....dd                                                    ┐
キューダンプファイルではありません。                                    ┘※2
```

■英語の場合

```
*******   Queue dump information        Version(TP1/EE :aa-aa-aa)  *******
*******                                       (TP1/XTC:bb-bb-bb)  *******
Specify output：cc....cc
File name：dd....dd                                                     ┐
File creation date：eeee/ee/ee ee:ee:ee  eee.eee                        │
File creation version：(TP1/EE :ff-ff-ff)                               │※1
                       (TP1/XTC:gg-gg-gg)                               │
Service group name：hh....hh                                            │
Run ID：0xiiiiiiii                                                      ┘

File name：dd....dd                                                     ┐
This is not a queue dump file.                                          ┘※2
```

注※ 1

指定されたファイルがキューダンプファイルの場合，キューダンプファイルの情報を繰り返し出力します。

注※ 2

指定されたファイルがキューダンプファイル以外の場合に出力します。

変数の説明

変数	意味
aa-aa-aa	コマンドを実行した TP1/EE のバージョン
bb-bb-bb	コマンドを実行した XTC のバージョン

変数	意味
cc....cc	編集コマンドの引数
dd....dd	キューダンプファイル名
eeee/ee/ee ee:ee:ee eee.eee	ファイルに出力を開始した日付 年号 / 月 / 日　時間：分：秒 ミリ秒．マイクロ秒
ff-ff-ff	ファイルを作成した TP1/EE のバージョン
gg-gg-gg	ファイルを作成した XTC のバージョン
hh....hh	サービスグループ名（最大 31 文字）
iiiiiiii	TP1/EE のラン ID（16 進数，8 けた）

出力メッセージ

メッセージ ID	内容	出力先
KFSB95400-E	メモリを確保できません。	標準エラー出力
KFSB95401-E	コマンドの形式が不正です。	標準エラー出力
KFSB95402-E	コマンドのオプションが不正です。	標準エラー出力
KFSB95403-E	コマンドのオプションの組み合わせが不正です。	標準エラー出力
KFSB95409-E	バージョンが不一致です。	標準エラー出力
KFSB95447-I	ヘルプメッセージ	標準出力

注意事項

編集対象の滞留メッセージがない場合，ヘッダ情報だけを表示します。

eetrbwtor（メッセージへの応答）

TP1/EE の運用コマンドです。

TP1 キャッシュ機能を使用する場合，応答するメッセージに次のメッセージが追加になります。

- KFSB85801-Q メッセージ
- KFSB85802-Q メッセージ

機能

eetrbwtor コマンド実行待ちメッセージを出力した TP1/EE に応答します。

オプションの説明

応答するメッセージによって，指定できるオプションが異なります。メッセージごとの指定できるオプションを次の表に示します。

表 6-6　eetrbwtor コマンドに指定できるオプション

メッセージ ID	内容	オプション	オプションの意味
KFSB85801-Q	実行系孤立時の，孤立終了モードの指定	{-a ｜ -b}	-a：孤立終了モード A で終了します。 -b：孤立終了モード B で終了します。
KFSB85802-Q	キューダンプファイルの出力障害	{-r ｜ -c}	-r：キューダンプファイルの出力をリトライします。 -c：キューダンプファイルの出力を中止して，オンラインを終了します。

7 CL 連携による系切り替え機能

この章では，CL 連携による系切り替え機能の仕組みと運用方法について説明します。

7.1 CL 連携の概要

CL サーバでは，CL 連携による系切り替えを行って高可用性を実現しています。ここでは，CL 連携による系切り替えの概要について説明します。

なお，この章の説明は，HA モニタによる系切り替え機能の基礎的な知識があることを前提としています。HA モニタについては，マニュアル「高信頼化システム監視機能 HA モニタ Linux(R) 編」を参照してください。

7.1.1 CL 連携による系切り替え

CL 連携による系切り替え機能には，次の 3 種類があります。

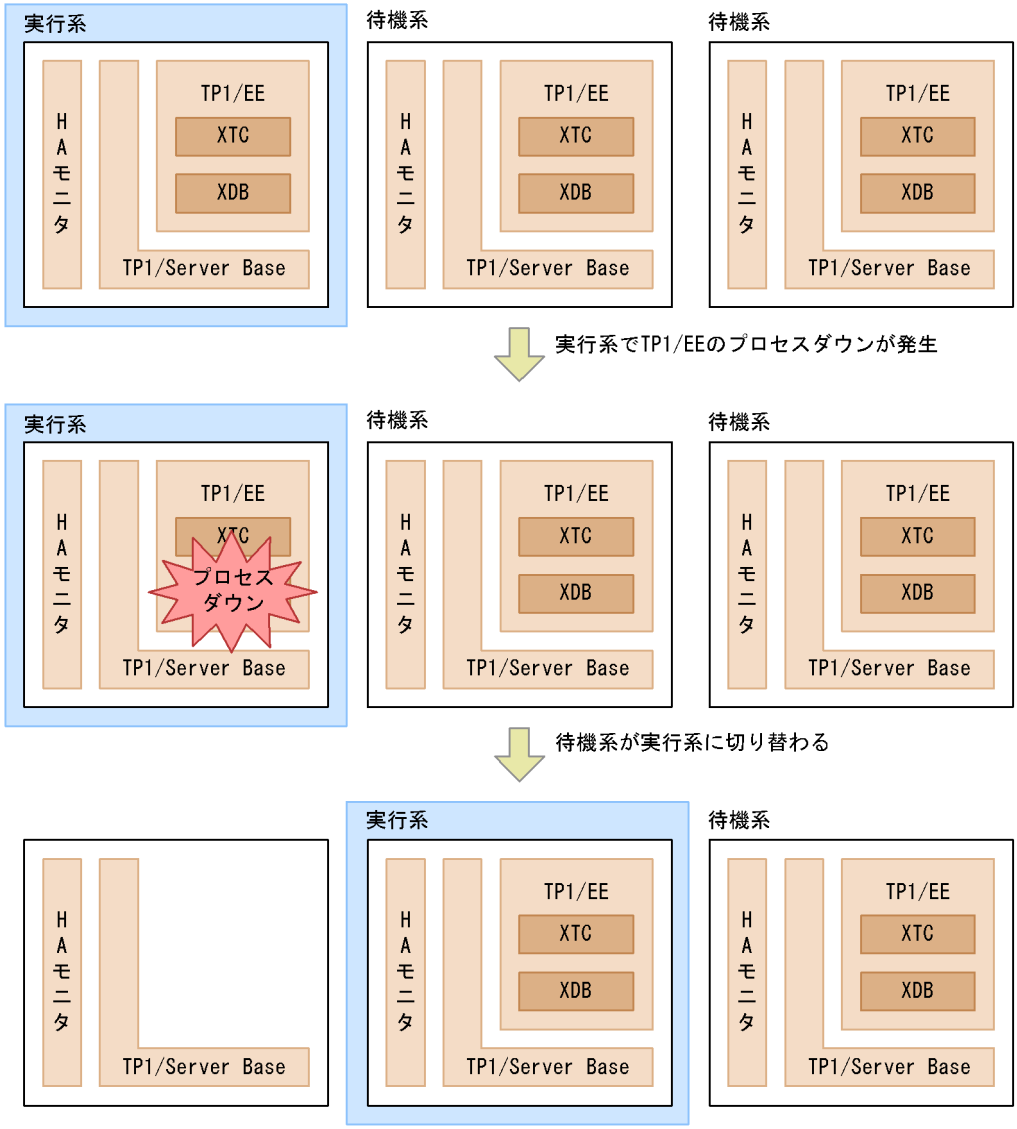
- 自動系切り替え
- 計画系切り替え
- 連動系切り替え

(1) 自動系切り替え

CL サーバでは，HA モニタのマルチスタンバイ機能を使用して，システムを三重化して障害に備えています。処理を実行しているマシンで障害が発生した場合，待機しているマシンに自動的に業務処理が切り替わります。これを自動系切り替えといいます。

自動系切り替えを次の図に示します。

図 7-1　自動系切り替え



説明

実行系で TP1/EE のプロセスダウンが発生した場合，待機系の一つを実行系に切り替えて，オンライン処理を続行します。
なお，オンライン中に業務処理を行っている系を**実行系**，待機している系を**待機系**といいます。XTC の開始時，どのサーバが実行系，待機系になるかは HA モニタの定義で設定します。

なお，CL 連携を行うサーバのグループを**クラスタグループ**といいます。クラスタグループは，UDP グループ情報関連定義の clgrpdef 定義コマンドで定義します。

参考

HA サーバは，マルチキャストによって CL サーバの各サーバにメッセージを送信しているため，どのサーバが実行系であるかを意識する必要はありません。

(2) 計画系切り替え

HA モニタの計画系切り替えコマンドを実行すると，実行系を強制停止して待機系に切り替えます。これを，計画系切り替えといいます。

(3) 連動系切り替え

実行系で複数のサーバが動作している場合に，障害発生や計画系切り替えコマンドの実行を契機として，複数のサーバを一括して切り替える機能を連動系切り替えといいます。

連動系切り替えは，複数のサーバをグループ化することで実現します。グループ化したサーバでは，そのうちの一つに障害が発生した際に，グループ単位で系を切り替えることができます。なお，グループ化したサーバ内でも，障害発生によって系を切り替えるサーバと系を切り替えないサーバを個別に設定できます。

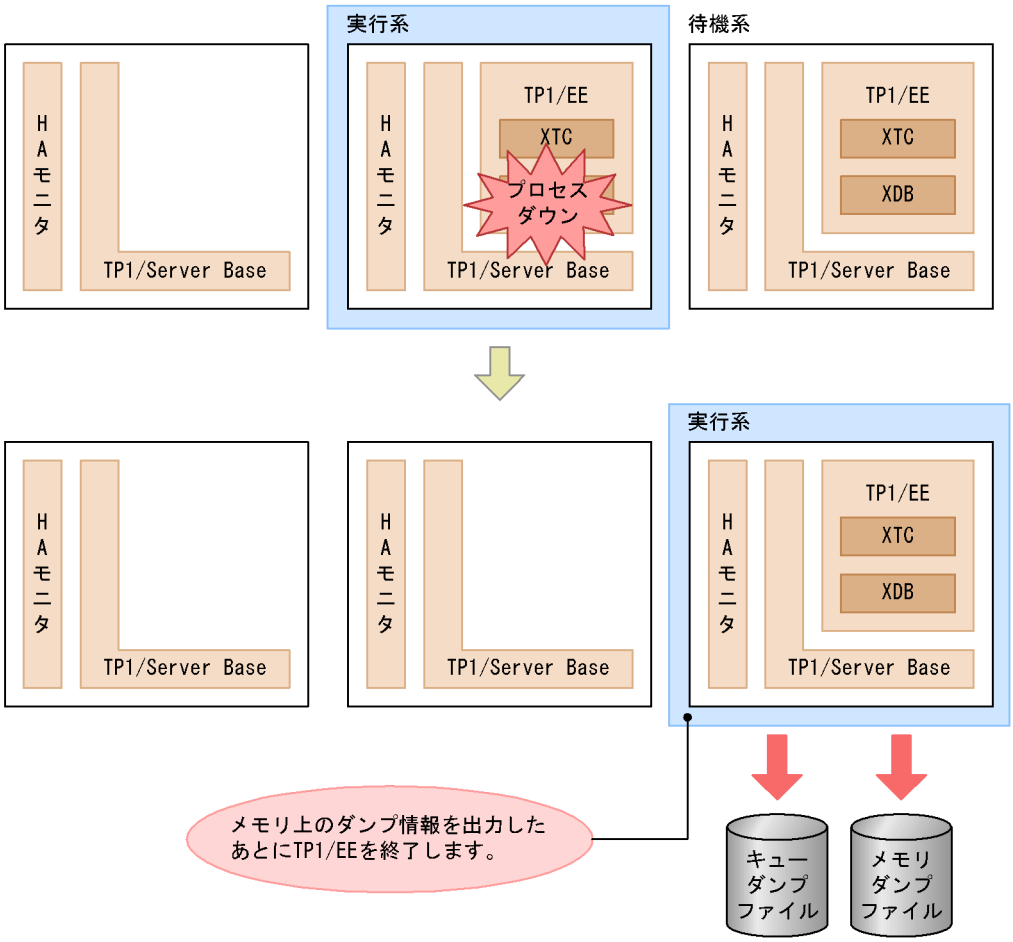
連動系切り替えの環境設定は，HA モニタで行います。

7.1.2 待機系がなくなった場合の XTC の処理

TP1/EE のプロセスダウンが続くと，待機系がなくなり，実行系だけの 1 台構成になることがあります。この状態を**実行系孤立状態**といいます。実行系孤立状態となった場合，XTC はデータの保全を最優先して，メモリ上のダンプ情報をファイルに出力して停止します。

実行系孤立状態となったときの XTC の処理を次の図に示します。

図 7-2　実行系孤立状態となったときの XTC の処理



説明

実行系孤立状態となった場合，キューダンプとメモリダンプをファイルに出力して，TP1/EE を終了します。

CL サーバが 1 台のサーバだけで稼働することはありません。

実行系孤立状態となった場合の対処については，「7.4　実行系孤立状態になった場合の対処」を参照してください。

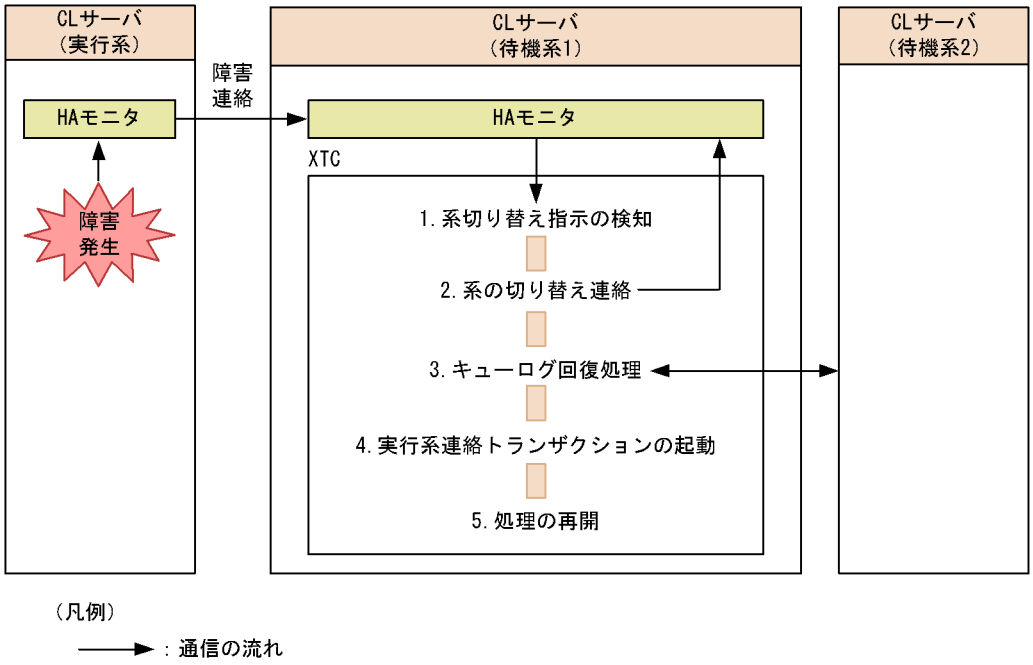
なお，実行系と待機系で通信ができないために，実行系孤立状態になることがあります。通信障害が発生したときの XTC の処理については，「7.5　実行系と待機系の間で通信障害が発生した場合の対処」を参照してください。

7.1.3　系が切り替わるときの処理の流れ

系が切り替わるときの処理の流れを次の図に示します。ここでは，障害発生を契機とし

た系切り替えを例に説明します。

図 7-3　系が切り替わるときの処理の流れ



説明

1. HA モニタからの系切り替え指示を検知します。
2. 自系の HA モニタに自系の TP1/EE が待機系から実行系に切り替わることを連絡します。
3. キューログ回復処理によって，自系の TP1/EE と待機系のリソースの状態を一致させます。キューログ回復処理については，「7.10　系切り替えが発生したときの XTC の処理」を参照してください。
4. 実行系連絡トランザクション（UI）を起動します。
   系切り替えが発生した場合，待機系が実行系に切り替わったことをユーザに通知するために実行系連絡トランザクションが起動されます。
5. 切り替わった系で処理を再開します。

注意事項

- XTC の開始処理中は系切り替えが発生しないため，実行系連絡トランザクションが起動することはありません。しかし，XTC の終了処理中に障害が発生すると，タイミングによっては実行系連絡トランザクションが起動することがあります。
- 実行系連絡トランザクションでは，系切り替え前の実行系の状態を参照できます。実行系連絡トランザクション（UI）の XTC 拡張トランザクションインタフェース情報に EERPC_STATUS_BEFORE_ME が設定されている場合は，実行系連絡トランザクションの終了後に XTC の終了処理を行います。そのため，新規のトランザクションや，リソースの転送が発生する API またはコマンドを実行しないでください。XTC の終了処理が完了する前に系切り替えが発生すると，再度実行系連絡トランザクションが起動されます。このとき，同じ API またはコマンドを実行すると，データベースの二重更新や，メッセージの二重送信などが発生するおそれがあります。

7.1.4　CL 連携時の制限事項

CL 連携による系切り替えの制限事項を次に示します。

- CL サーバの稼働中は，プロセスダウンした TP1/EE を再開始できません。そのため，TP1/EE のプロセスダウンによって 2 台構成になった CL サーバを 3 台構成に戻すには，CL サーバの各サーバをいったん停止し，3 台構成に戻したあと，CL サーバを再起動する必要があります。
- CL サーバのシミュレート機能を使用している場合は，HA モニタと連携しません。CL サーバのシミュレート機能については，「2.10.1　CL サーバのシミュレート機能」を参照してください。

7.1.5　HA モニタ情報ファイル

XTC が HA モニタに接続したとき，$DCDIR/spool/dceeinf 下に HA モニタ情報ファイルが作成されます。このファイルは XTC と HA モニタの間で使用され，XTC の正常終了時，XTC が HA モニタ情報ファイルを自動的に削除します。

XTC の稼働中に HA モニタ情報ファイルの読み込みまたは書き込みに失敗すると，XTC は異常終了します。この場合，原因を取り除いたあとに，XTC を再起動する必要があります。

なお，XTC が異常終了した場合，HA モニタ情報ファイルが自動的に削除されないことがあります。XTC の再起動時に HA モニタ情報ファイルが存在すると，KAMN308-E メッセージ（待機サーバが異常終了した旨のメッセージ）が出力されますが，XTC の再起動には影響ありません。そのため，このメッセージは無視してください。

7.2 自動系切り替えの監視対象となる障害

HA モニタは，TP1/EE のプロセスを監視しています。HA モニタが TP1/EE のプロセスダウンを検知したときに，HA モニタが系を切り替えます。また，そのほかに，ハードウェア障害や，カーネル障害が発生したときも，HA モニタによって系が切り替えられます。

7.2.1 HA モニタが監視する障害

HA モニタが監視する障害を次の表に示します。これらの障害が発生したときに系が切り替わります。

表 7-1 HA モニタが監視する障害

項番	障害の種類	系切り替えの単位※
1	ハードウェア障害	HA モニタ単位
2	ハードウェアの電源断	
3	カーネル障害	
4	HA モニタの異常終了	
5	TP1/Server Base の異常終了	TP1/Server Base 単位
6	スレッドハングアップ時間監視や，トランザクション処理時間監視による TP1/EE のプロセスダウン	TP1/EE プロセス単位
7	TP1/EE プロセスの論理エラー	
8	リソースの障害（障害によってシステムの運転が続行できない場合）	

注※
系切り替えの単位の詳細については，「7.2.2 系切り替えの単位」を参照してください。

通信障害は，HA モニタの監視対象外となります。通信障害が発生した場合の XTC の処理については，「7.5 実行系と待機系の間で通信障害が発生した場合の対処」を参照してください。

■スローダウンの検知

TP1/EE のスレッドハングアップ時間監視によってシステムのスローダウンを検知しています。スレッドハングアップ時間監視によって TP1/EE のプロセスをダウンさせて，系を切り替える仕組みになっています。

7.2.2 系切り替えの単位

発生した障害の種類によって系が切り替わる単位が異なります。次に示す三つのうちのどれかの単位で系が切り替わります。

- HA モニタ単位
- TP1/Server Base 単位
- TP1/EE プロセス単位

(1) HA モニタ単位の系切り替え

ハードウェア障害，カーネル障害，HA モニタが異常終了した場合などは，系障害として系切り替えが行われます。この場合，このサーバ下の全 TP1/EE プロセスの系が切り替わります。これを HA モニタ単位の系切り替えといいます。

(2) TP1/Server Base 単位の系切り替え

TP1/Server Base は HA モニタの監視対象外のため，TP1/Server Base が異常終了しても，TP1/Server Base の系は切り替わりません。TP1/Server Base が異常終了した場合，TP1/EE のプロセスもダウンします。この TP1/EE のプロセスダウンを HA モニタが検知して，系を切り替える仕組みになっています。この場合，TP1/Server Base 下の全 TP1/EE プロセスの系が切り替わります。これを TP1/Server Base 単位の系切り替えといいます。

注意事項

CL 連携を行う場合，CL サーバの TP1/Server Base は常に実行系として動作するため，TP1/Server Base を HA モニタの監視対象にしないでください。そのためには，システムサービス構成定義の ha_conf オペランドに N を指定してください。

(3) TP1/EE プロセス単位の系切り替え

TP1/EE のプロセスダウンが発生した場合，プロセスダウンした TP1/EE プロセスの系だけが切り替わります。ほかの TP1/EE プロセスの系は切り替わりません。これを TP1/EE プロセス単位の系切り替えといいます。

7.3 系の状態表示

CL サーバの各サーバの状態は，HA モニタの monshow コマンド，または TP1/EE/XTC の eehamls コマンドで確認できます。確認できる内容を次の表に示します。

コマンド	確認できる内容
monshow	• XTC が稼働中かを確認できます。 • 実行系 XTC が稼働中のホストを確認できます。
eehamls	• 実行系の XTC で eehamls コマンドを実行すると系切り替えできる状態かを確認できます。 ONL の系が 1 つ，SBY の系が 2 つ表示された場合は，系切り替えできる状態です。 ONL の系が 1 つ，SBY の系が 1 つの場合，系切り替え後，孤立終了モードとなる状態です。

7.4　実行系孤立状態になった場合の対処

ここでは，待機系がなくなり，実行系孤立状態になったときの XTC の処理と管理者の対処について説明します。

(1) XTC の処理

実行系孤立状態となった場合，XTC は次に示す順序で終了処理を行います。

1. 受信メッセージに対してエラーを応答するか，または受信メッセージを破棄します。
   送信元で発行した API によって処理が異なります。XTC の処理は次に示すどちらかになります。
   - 一方送信メッセージを受信した場合，受信メッセージを破棄します。送信元では，エラートランザクションを起動します。
   - RPC メッセージを受信した場合，受信メッセージを破棄します。送信元で発行した API がエラーとなります。
2. 出力キューを閉塞します。
3. トランザクションの受け付けを休止します。
   実行中のトランザクションの終了を待ちます。新規トランザクションは起動しません。
4. キューログ回復処理を行います。
   キューログ回復処理については，「7.10　系切り替えが発生したときの XTC の処理」を参照してください。
5. キューダンプを出力します。
   XTC の終了時に出力するキューダンプと同じ情報が出力されます。キューダンプの出力先については，「2.9.2(2)(b) 実行系孤立状態になったとき」を参照してください。
6. XDB のデータベースのデータを格納したメモリダンプを出力します。
   メモリダンプについては，マニュアル「TP1/EE/Extended Data Cache 使用の手引」を参照してください。
7. eetrbwtor コマンドの入力待ち状態になります。
   管理者は，孤立終了モードを指定して eetrbwtor コマンドを実行してください。詳細については，「(2) 管理者の対処」を参照してください。
   この間，トランザクションの受け付けを休止しているため，新規トランザクションは起動されません。
8. 実行系連絡トランザクション（UI）を起動します。
9. トランザクションの受け付けを開始します。
10. 出力キューの閉塞を解除します。
11. XTC の終了処理を行います。
    eetrbwtor コマンドに指定された孤立終了モードに従って XTC を終了します。
    このとき，終了トランザクションのトランザクションインタフェース情報のプロセス終了要因に孤立終了モードが設定されます。

参考

XTC の開始処理中に実行系孤立状態になった場合，XTC の開始処理を中止します。

(2) 管理者の対処

XTC がメモリダンプを出力したあと，eetrbwtor コマンドの入力待ち状態になります。管理者は，孤立終了モードを指定して eetrbwtor コマンドを実行し，XTC を終了してください。

■孤立終了モード

孤立終了モードには，次に示す 2 種類があります。

- **孤立終了モード** A
  正常終了と同様に，処理キュー上の受け付け済みのサービス要求をすべて処理したあとに XTC を終了します。出力キュー（OTQ）の終了監視は，mch_send_end_otqwatch オペランドの指定値に従います。このとき，ERRTRN4 を起動します。
- **孤立終了モード** B
  計画停止 B と同様に処理キュー上の受け付け済みのサービス要求をすべて破棄します。出力キューの終了監視は行いません。

なお，次に示す場合は，孤立終了モード A で XTC が終了します。

- eetrbwtor コマンドを実行する前に eesvstop コマンドを実行した場合
- XTC の終了処理中に実行系孤立状態となった場合

(3) DB 管理者の対処

メモリダンプファイルから表データを抽出する場合，最新のメモリダンプファイルを使用してください。最新のメモリダンプファイルの判断方法は，すべての系のメモリダンプファイル名称を比較し，一番進んでいるコミット通番から判断してください。プロセスダウンなどでメモリダンプファイルが出力されないものは比較対象から除いてください。

メモリダンプファイル名称は次のようになります。

”サービスグループ名 _ ノード ID_ ラン ID_CL 通番 _xdb_ コミット通番 ”

- ラン ID：16 進数 8 けた固定
- CL 通番：16 進数 16 けた固定
- コミット通番：16 進数 16 けた固定

7.5　実行系と待機系の間で通信障害が発生した場合の対処

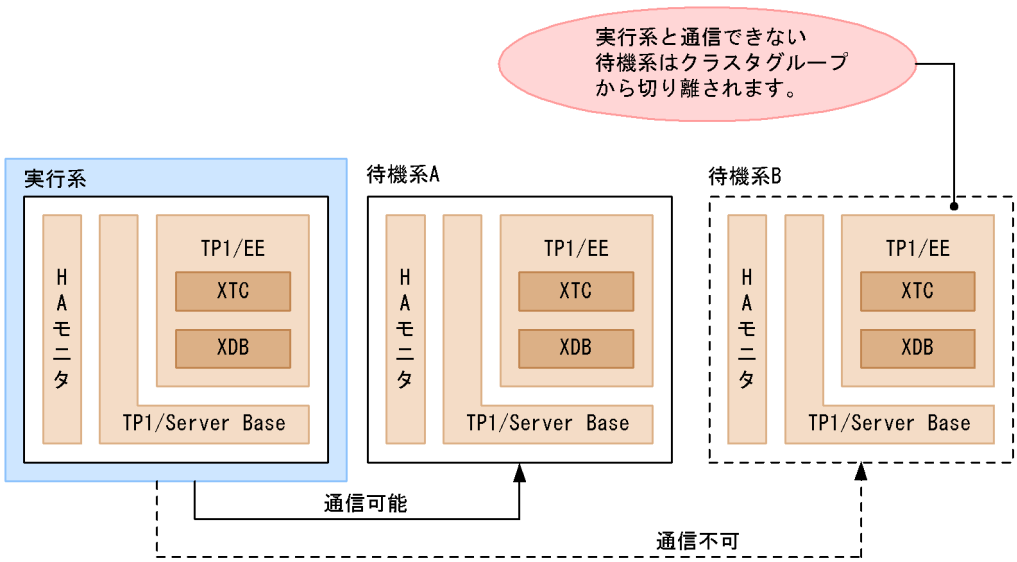
ここでは，実行系と待機系の間で通信障害が発生した場合の XTC の処理と管理者の対処ついて説明します。

7.5.1　通信障害が発生した場合の XTC の処理と管理者の対処

CL 連携では，実行系と待機系の間で通信を行い，実行系の状態を待機系に逐次反映しています。実行系と待機系の間で通信障害が発生した場合，通信障害が発生した待機系をクラスタグループから切り離します。

実行系と待機系の間で通信障害が発生した場合の XTC の処理を次の図に示します。

図 7-4　実行系と待機系の間で通信障害が発生した場合の XTC の処理



■管理者の対処

切り離された待機系（図中の待機系 B）で eesvstop コマンドなどを実行して，待機系をすぐに終了してください。

注意事項

切り離された待機系（図中の待機系 B）が終了する前に実行系が異常終了した場合，切り離された待機系が実行系になることがあります。この場合，リソースなどが正しく待機系に転送されないで，メモリ内の情報が古くなっているおそれがあります。そのため，管理者は次に示す手順で実行系を切り替えてください。

1. 切り替わった実行系と待機系の間（図中の待機系 A と待機系 B の間）で通信ができることを確認してください。
2. 切り替わった実行系（図中の待機系 B）を終了させて，もう一つの待機系（図中の待機系 A）を実行系にしてください。
   このとき，実行系孤立状態になります。対処方法については，「7.4　実行系孤立状態になった場合の対処」を参照してください。

なお，切り替わった実行系と待機系の間で通信ができない場合は，系を切り替えることはできません。

通信障害によって待機系がなくなり，実行系孤立状態になった場合は，メモリ上のダンプ情報をファイルに出力して停止します。詳細については，「7.1.2　待機系がなくなった場合の XTC の処理」を参照してください。

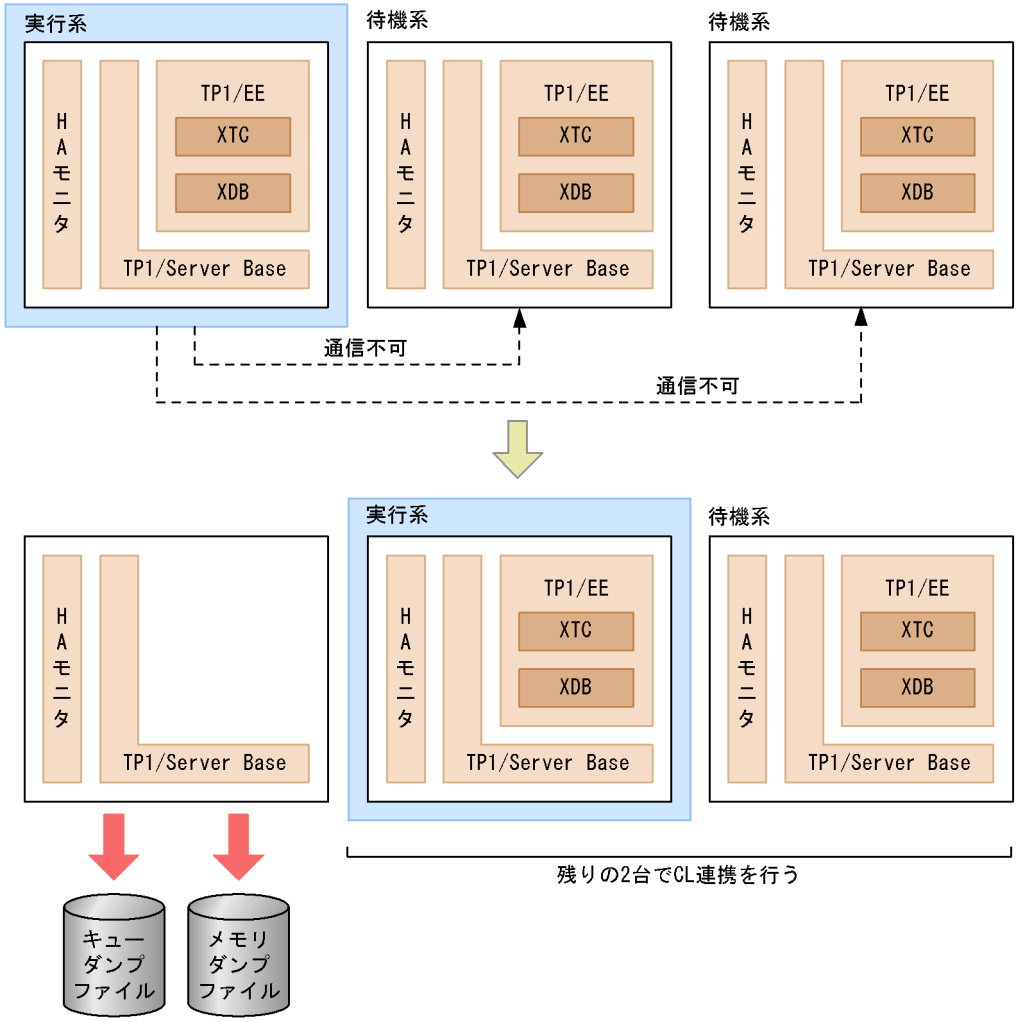
なお，通信障害は HA モニタで監視できません。そのため，HA モニタが管理しているサーバの状態と，XTC が管理しているサーバの状態が異なることがあります。

7.5.2 実行系とすべての待機系との間で通信ができない場合の XTC の処理

実行系とすべての待機系との間で通信ができない場合は，キューダンプとメモリダンプをファイルに出力して，実行系を終了します。そして，待機系のうちの一つを実行系にして業務を続行します。

実行系とすべての待機系との間で通信ができない場合の XTC の処理を次の図に示します。

図 7-5　実行系とすべての待機系との間で通信ができない場合の XTC の処理



7.6 転送処理によるリソースの永続化

XTC では，次に示す情報を待機系に逐次転送して永続化しています。

- 実行系で更新されたリソースの情報
- 実行系で処理されたトランザクションの情報

これによって，実行系で TP1/EE のプロセスダウンが発生しても，実行系のリソース更新結果とトランザクションを待機系で保証しています。これら一連の処理を**転送処理**といいます。

ここでは，転送処理によるリソースの永続化処理について説明します。

7.6.1 待機系に転送されるリソース

ここでは，転送処理によって待機系に転送されるリソースについて説明します。

待機系に転送されるリソースの一覧を次の表に示します。XTC では，これらのリソースを待機系に転送することでリソースを永続化しています。

表 7-2 待機系に転送されるリソースの一覧

項番	待機系に転送されるリソース		説明
1	入力メッセージ※1		実行系で受信した次の入力メッセージです。 • RPC メッセージ RPC 要求メッセージ（RPC 関連定義の rpc_recv_permanence オペランドに Y を指定）です。 • MCP メッセージ MCP が受信したメッセージ（MCP 構成定義の eemcpfunc 定義コマンドで -m オプションの recvpermanence オペランドに yes を指定）です。 • MCH メッセージ 永続指定のトランザクション同期の一方送信メッセージ，およびトランザクション非同期の一方送信メッセージです。
2	トランザクションと同期して転送されるリソース※2	送信メッセージ	UAP からの ee_mch_cmtsend 関数（永続指定）要求時のメッセージを永続化します。
3		XDB の更新ログ	データベースの更新情報を永続化します。
4		タイマトランザクションの起動要求	UAP からの ee_tim_execap 関数（永続指定）要求時のメッセージを永続化します。
5		滞留メッセージ	UAP からの ee_scd_msg_receive 関数の発行によって受信した永続指定の入力メッセージを永続化します。

項番	待機系に転送されるリソース		説明
6	トランザクションと非同期で転送されるリソース※3	サービスの閉塞状態	次に示す閉塞状態を永続化します。 • スレッドダウンによるサービスの自動閉塞 • eedctsv コマンド，eeactsv コマンド，または ee_thd_abdctl 関数で変更されたサービスの閉塞状態
7		出力キュー（OTQ）の閉塞状態	次に示す閉塞状態を永続化します。 • 一方送信メッセージの障害による出力キューの自動閉塞 • eemchotqact コマンド，eemchotqdct コマンド，または ee_mch_otqbkctl 関数で変更された出力キューの閉塞状態
8		サービスの最大同時処理限界数	eelspce コマンドで変更したサービスの最大同時処理限界数を永続化します。
9		出力キュー中の未送信メッセージのスキップ状態	eemchotqskip コマンド，または ee_mch_otqskip 関数で変更した出力キュー中の未送信メッセージのスキップ状態を永続化します。
10		滞留メッセージのスキップ状態	eepceskip コマンド，または ee_scd_msg_skip 関数で変更した滞留メッセージのスキップ状態を永続化します。

注※ 1

詳細については，「7.6.2　入力メッセージの永続化」を参照してください。

注※ 2

詳細については，「7.6.3　トランザクションと同期して転送されるリソースの永続化」を参照してください。

注※ 3

詳細については，「7.6.4　トランザクションと非同期に転送されるリソースの永続化」を参照してください。

7.6.2　入力メッセージの永続化

ここでは，入力メッセージの永続化の処理方式について，メッセージの種類別に説明します。

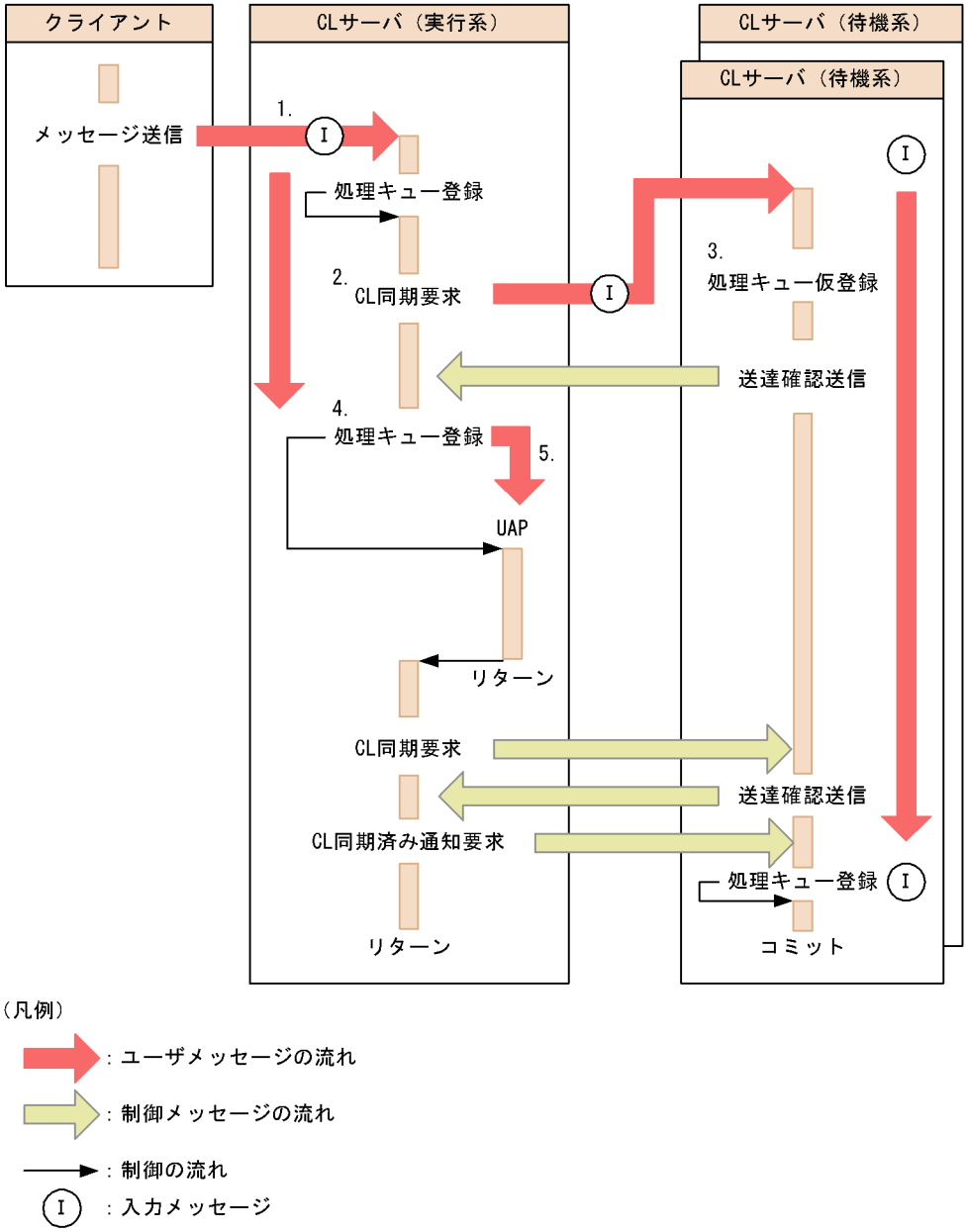
(1) RPC メッセージの永続化

クライアントから CL サーバに対してメッセージの送信要求を行います。RPC メッセージは，実行系が待機系に転送することで永続化されます。

RPC メッセージを永続化するには，RPC 関連定義の rpc_recv_permanence オペランドに Y を指定します。また，永続化できるのは，RPC 要求メッセージの受信メッセージだけです。

RPC メッセージの永続化処理の流れを次の図に示します。

図 7-6　RPC メッセージの永続化処理の流れ



説明

1. 実行系はクライアントから RPC メッセージを受信したあと，メッセージ転送用の処理キューを登録します。処理キューは，サービスごとにシリアライズされます。サービス名の不正を検知した場合は，処理キューを登録しないで ERRTRN1

を起動します。

2. 送信スレッドで処理キューが引き出され，受信した RPC メッセージを待機系に転送します。このメッセージのように，実行系から待機系にリソースの永続化を要求するメッセージを CL **同期メッセージ**といいます。CL 同期メッセージを送信したあと，待機系からの送達確認メッセージの待ち合わせを行います。
3. 待機系は実行系から転送された RPC メッセージを受信したあと，処理キュー制御用バッファ（PCE）を確保し，処理キューを仮登録します。そのあと，実行系に対して送達確認メッセージを送信します。
4. 実行系は待機系からの送達確認メッセージをすべて受信したあとに，RPC メッセージを処理キューに登録します。
5. 処理スレッドで処理キューが引き出され，UAP に RPC メッセージを引き渡します。

以降の処理については，次の項以降で説明します。

なお，RPC メッセージの永続化の場合，次の制限があります。

- TCP/IP プロトコルの RPC メッセージを永続化した場合，次の機能は使用できません。
  - RPC 応答メッセージのトランザクションまたがり送信機能
  - ERRTRNR による RPC 応答メッセージ送信機能
- ee_rpc_cmtsend 関数の自プロセス通信の場合，RPC メッセージを永続化できません。
- CL 構成での RPC メッセージ受信であるため，トランザクション連携する RPC メッセージは永続化に失敗します。

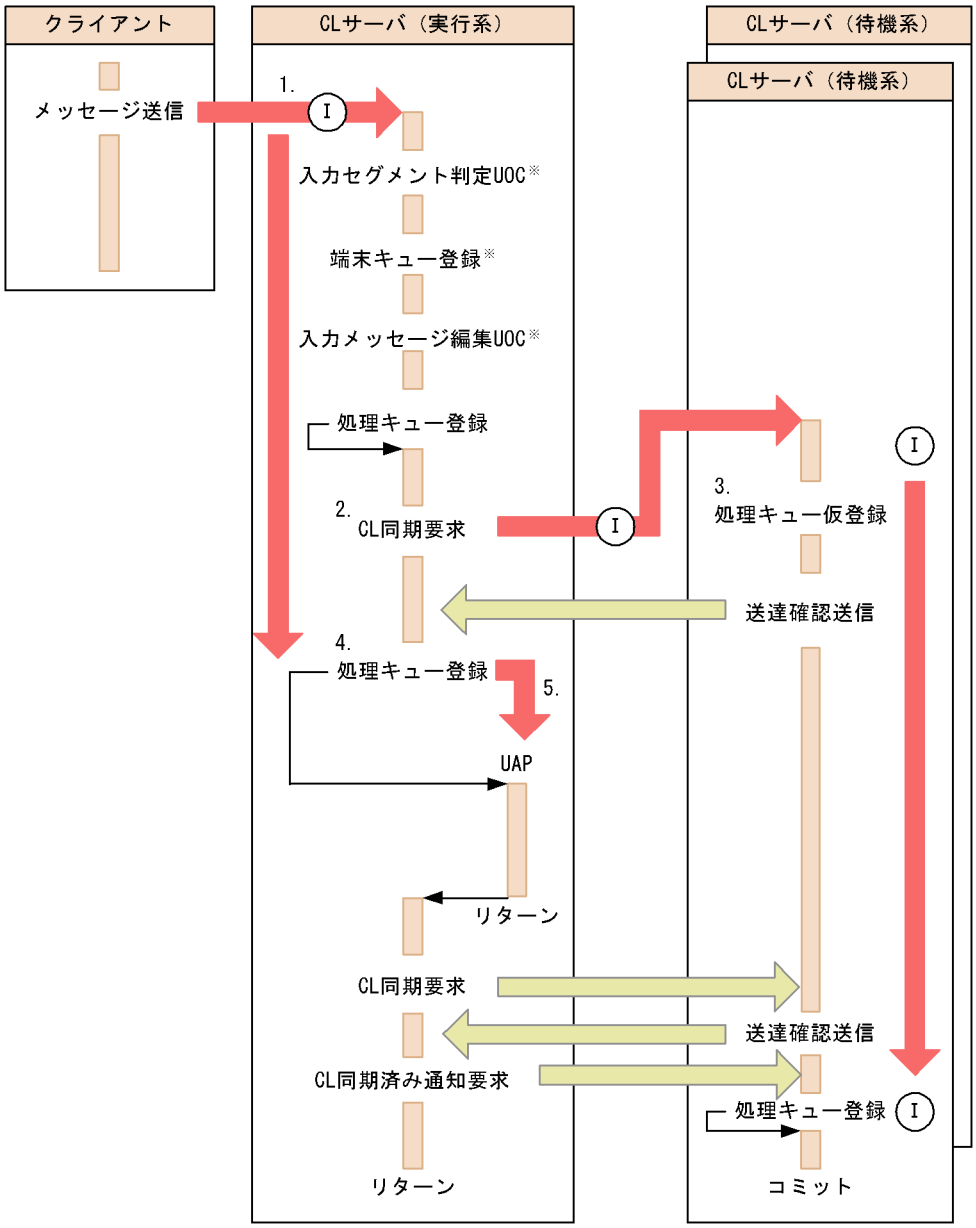
（2） MCP メッセージの永続化

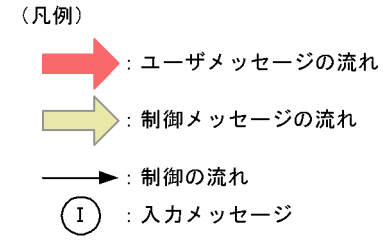
クライアントから CL サーバの MCP あてにメッセージの送信要求を行います。MCP メッセージは，実行系が待機系に転送することで永続化されます。

MCP メッセージを永続化するには，MCP 構成定義の eemcpfunc 定義コマンドで -m オプションの recvpermanence オペランドに yes を指定します。また，永続化できるのは，MCP メッセージの受信メッセージだけです。

MCP メッセージの永続化処理の流れを次の図に示します。

図 7-7　MCP メッセージの永続化処理の流れ





注※
実行するかどうかは，MCP構成定義の指定によります。

説明

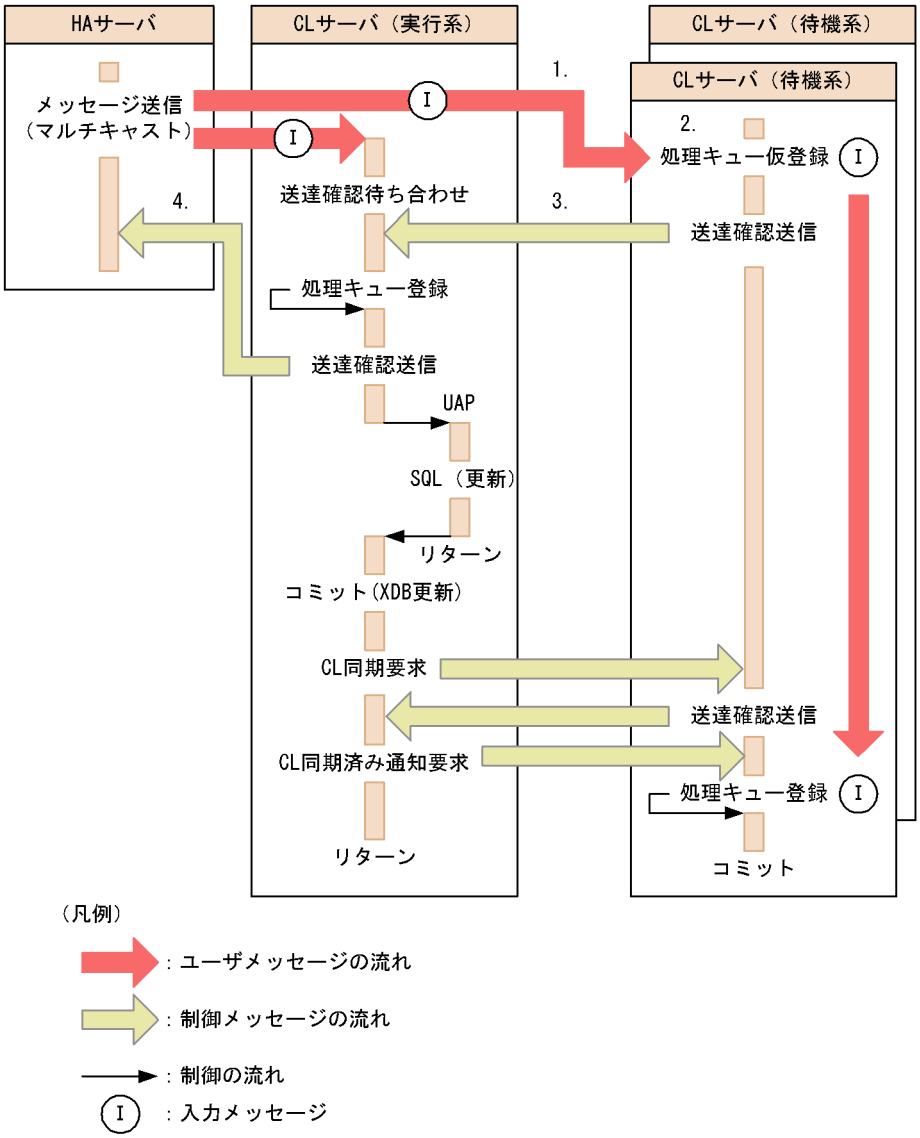
1. 実行系はクライアントから MCP メッセージを受信したあと，MCP 構成定義の指定値に従って，入力セグメント判定 UOC，端末キュー登録，および入力メッセージ編集 UOC を実行します。そのあと，メッセージ転送用の処理キューを登録します。処理キューは，サービスごとにシリアライズされます。サービス名の不正を検知した場合は，処理キューを登録しないで ERRTRN1 を起動します。
2. 送信スレッドで処理キューが引き出され，受信した MCP メッセージを待機系に CL 同期メッセージとして転送したあと，待機系からの送達確認メッセージの待ち合わせを行います。
3. 待機系は実行系から転送された MCP メッセージを受信したあと，処理キュー制御用バッファ（PCE）を確保し，処理キューを仮登録します。そのあと，実行系に対して送達確認メッセージを送信します。
4. 実行系は待機系からの送達確認メッセージをすべて受信したあとに，MCP メッセージを処理キューに登録します。
5. 処理スレッドで処理キューが引き出され，UAP に MCP メッセージを引き渡します。

以降の処理については，次の項以降で説明します。

（3） MCH メッセージの永続化

HA サーバから CL サーバの各サーバに対して，マルチキャストでメッセージの送信要求を行います。MCH メッセージは，実行系と待機系がそれぞれ保持することで永続化されます。MCH メッセージの永続化処理の流れを次の図に示します。

図 7-8　MCH メッセージの永続化処理の流れ



説明

1. HA サーバから CL サーバの各サーバに対して，マルチキャストでメッセージが送信されます。
2. 待機系は，HA サーバからメッセージを受信したあと，処理キュー制御用バッファ（PCE）を確保し，処理キューを仮登録します。そのあと，実行系に対して送達確認メッセージを送信します。
3. 実行系は，HA サーバからのメッセージを受信したあと，待機系からの送達確認メッセージの待ち合わせを行います。
4. 実行系は，待機系からの送達確認メッセージをすべて受信したあとに，HA サー

バに対して送達確認メッセージを送信します。

以降の処理については，次の項以降で説明します。

参考

ここでは，MCH メッセージの送信元として HA サーバを仮定しています。CL サーバから別のクラスタグループの CL サーバへマルチキャストでメッセージを送信することもできます。その場合，HA サーバが CL サーバになります。

7.6.3 トランザクションと同期して転送されるリソースの永続化

次に示すリソースは，トランザクションと同期して待機系に転送されます。

- 送信メッセージ
- XDB の更新ログ
- タイマトランザクションの起動要求
- 滞留メッセージ

ここでは，トランザクションと同期して転送されるリソースの永続化について説明します。永続トランザクションの場合と，非永続トランザクションの場合に分けて説明します。

■永続トランザクションと非永続トランザクション

待機系にトランザクション起動元となる入力メッセージがあるトランザクションを，**永続トランザクション**といいます。

待機系に入力メッセージがないトランザクションを，**非永続トランザクション**といいます。

(1) 永続トランザクションでのリソースの永続化

永続トランザクションでのリソースの永続化の処理方式について説明します。

(a) 送信メッセージの永続化

UAP から送信要求された永続指定のメッセージを待機系に転送して，送信メッセージを永続化します。送信メッセージの永続化処理の流れを次に示します。

1. 実行系の UAP から永続指定のメッセージ送信要求を受け付けると，コミット確定後に出力キュー（OTQ）にメッセージを登録（出力キュー通番を採番）し，メッセージ送信指示を行います。
2. 永続化するメッセージを CL 同期メッセージとして待機系に転送します。CL 同期メッセージを受信した待機系は，出力キュー通番の通番抜けがないことを確認後，仮登録済みの処理キューに送信メッセージを追加し，出力キューにメッセージを登録します。そのあと，送達確認メッセージを応答します。出力キュー通番の通番抜けがあ

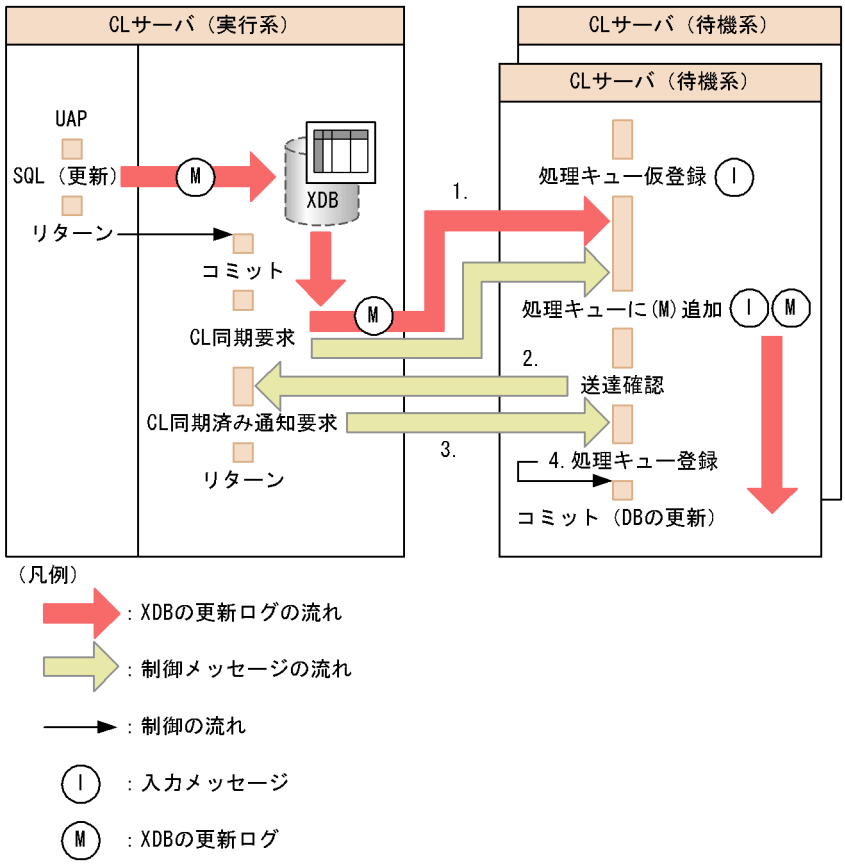
る場合は，送達確認メッセージの応答を待ち合わせます。

3. 実行系は，UAP から送信要求されたメッセージの送信指示を行ったあと，待機系に対して送信が完了した旨のメッセージを転送します。このメッセージを CL **同期済み通知メッセージ**といいます。
4. CL 同期済み通知メッセージを受信した待機系は，仮登録済みの処理キューを登録します。

(b) XDB の更新ログの永続化

XDB の更新ログ（実行系のデータベースの更新情報）を待機系に転送して，更新ログを永続化します。待機系では，更新ログに付加されたコミット通番に従ってデータベースへの反映を行います。更新ログの永続化処理の流れを次の図に示します。

図 7-9　更新ログの永続化処理の流れ



説明

1. UAP から SQL の更新要求を受け付けると，コミット確定後，実行系は待機系に更新ログを CL 同期メッセージとして転送します。
2. CL 同期メッセージを受信した待機系は，仮登録済みの処理キューに転送された更新ログを追加し，送達確認メッセージを応答します。更新ログの通番抜けがあ

る場合は，送達確認メッセージの応答を待ち合わせます。
3. 実行系から待機系に，CL 同期済み通知メッセージが転送されます。
4. CL 同期済み通知メッセージを受信した待機系は，仮登録済みの処理キュー（同時実行数 1）を登録します。更新ログの通番抜けがある場合は，処理キュー登録を待ち合わせます。

(c) タイマトランザクションの起動要求の永続化

UAP からのタイマトランザクションの起動要求時に永続指定がある場合，待機系にタイマトランザクションの情報を転送して，タイマトランザクションの起動要求を永続化します。タイマトランザクションの起動要求の永続化の流れを次に示します。

1. UAP からの永続指定のタイマトランザクション起動要求を実行系で受け付けると，コミット確定後，実行系は待機系に CL 同期メッセージを送信します。
2. CL 同期メッセージを受信した待機系は，タイマトランザクション起動用の処理キューを仮登録し，送達確認メッセージを応答します。
3. 送達確認メッセージを受信した実行系は，CL 同期済み通知メッセージを待機系に送信します。
4. CL 同期済み通知メッセージを受信した待機系は，仮登録済みの処理キューを登録します。
5. 実行系は，タイマトランザクションのコミット確定後，待機系に CL 同期メッセージを送信します。
6. CL 同期メッセージを受信した待機系は，送達確認メッセージを応答します。
7. 送達確認メッセージを受信した実行系は，CL 同期済み通知メッセージを待機系に送信します。
8. CL 同期済み通知メッセージを受信した待機系は，仮登録済みのタイマトランザクション用処理キューを登録します。

(d) 滞留メッセージの永続化

UAP からの ee_scd_msg_receive 関数の発行によって受信した入力メッセージを待機系に転送します。このとき，待機系で受信済みの入力メッセージは破棄されます。

(2) 非永続トランザクションでのリソースの永続化

非永続トランザクションで，永続指定の送信メッセージや，XDB の更新ログなどの永続化が必要なリソースがある場合，待機系にリソースを転送してリソースを永続化します。ただし，リソースを転送する前に実行系が異常終了した場合，待機系には仮登録した処理キューがないため，UAP の再起動によるリソースの回復はできません。

非永続トランザクションでの XDB の更新ログの永続化処理の流れを次に示します。

1. 実行系の UAP から SQL（更新）要求を受け付けると，コミット確定後に待機系に更新ログを転送します（CL 同期メッセージ）。
2. CL 同期メッセージを受信した待機系は，仮登録済みの処理キューがないため，処理キュー制御用バッファ（PCE）を確保して処理キューの仮登録を行います。そのあ

と，実行系に送達確認メッセージを応答します。

以降の処理は，永続トランザクションでのリソースの永続化処理の流れと同じになります。

なお，非永続トランザクションの場合，待機系に転送するリソースがないときは，CL 同期メッセージまたは CL 同期済み通知メッセージは送信されません。

7.6.4 トランザクションと非同期に転送されるリソースの永続化

次に示すリソースは，リソースの状態に変更があったときに待機系に転送されます。

- サービスの閉塞状態
- 出力キュー（OTQ）の閉塞状態
- サービスの最大同時処理限界数
- 出力キュー中の未送信メッセージのスキップ状態
- 滞留メッセージのスキップ状態

ただし，実行系の異常終了のタイミングによっては，実行系のリソースの状態が待機系に正しく引き継がれないことがあります。そのため，次に示すリソースについては，系が切り替わったあとに運用コマンドでリソースの状態を確認してください。

■サービスの閉塞状態

系が切り替わった場合に，閉塞中のサービスがある旨のメッセージが出力されたときは，eelssv コマンドでサービスの状態を確認してください。そのあと，必要に応じて eedctsv または eeactsv コマンドで，サービスの閉塞状態を変更してください。

■出力キュー（OTQ）の閉塞状態

系が切り替わったときに，出力キューが閉塞中のサービスがある旨のメッセージが出力された場合，eemchotqls コマンドで出力キューの状態を確認してください。そのあと，必要に応じて eemchotqdct または eemchotqact コマンドで，出力キューの閉塞状態を変更してください。

7.6.5 UAP からロールバック要求があった場合の転送処理

ここでは，UAP からロールバック要求があった場合の XTC の処理について説明します。

同時引き出しなしのサービスの場合と，同時引き出しありのサービスの場合で処理が異なります。それぞれの処理を説明します。

(1) 同時引き出しなしのサービスの場合

実行される関数によって処理が異なります。

(a) ee_trn_chained_rollback 関数によるロールバック要求の場合

UAP から ee_trn_chained_rollback 関数によるロールバック要求があった場合，入力メッセージを未読み出し状態に戻したあと（処理キューを引き戻したあと），再度処理キューを引き出してトランザクションを実行します。

ロールバック時の処理の流れを次に示します。

1. UAP からロールバック要求を受け付けると，同期してロールバック決着を行い，入力メッセージを未読み出し状態に戻します。ただし，連鎖モード連携機能の使用有無によっては，UAP からのリターン後に入力メッセージを未読み出し状態に戻す場合があります。
2. UAP からのリターン後，コミット決着を行いリターンします。永続化が必要なリソースがある場合は，CL 同期メッセージを待機系に送信します。
3. 引き戻した処理キューを再度引き出し，UAP に制御を渡します。

(b) ee_trn_rollback_mark 関数によるロールバック要求の場合

UAP から ee_trn_rollback_mark 関数によるロールバック要求があった場合，入力メッセージを未読み出し状態に戻したあと（処理キューを引き戻したあと），再度処理キューを引き出してトランザクションを実行します。

ロールバック時の処理の流れを次に示します。

1. UAP からロールバック要求を受け付けると，UAP からのリターン後にロールバック決着を行い，入力メッセージを未読み出し状態に戻します。
2. 引き戻した処理キューを再度引き出し，UAP に制御を渡します。

(2) 同時引き出しありのサービスの場合

実行される関数によって処理が異なります。

(a) ee_trn_chained_rollback 関数によるロールバック要求の場合

UAP から ee_trn_chained_rollback 関数によるロールバック要求があった場合，ロールバック情報を待機系へ転送し，入力メッセージを削除します。

ロールバック時の処理の流れを次に示します。

1. UAP からロールバック要求を受け付けると，同期してロールバック決着を行い，ロールバック決着情報を転送（CL 同期メッセージ）し，続けて CL 同期済み通知メッセージを送信します。ただし，連鎖モード連携機能の使用有無によっては，UAP からのリターン後に CL 同期メッセージまたは CL 同期済み通知メッセージを送信する場合があります。
2. CL 同期済み通知メッセージを受信した待機系は，仮登録済みの処理キューを登録し，ロールバック決着を行います。

（b） ee_trn_rollback_mark 関数によるロールバック要求の場合

UAP から ee_trn_rollback_mark 関数によるロールバック要求があった場合，UAP リターン後にロールバック情報を待機系に転送します。

7.6.6 UAP が異常終了した場合の転送処理

UAP が異常終了した場合，待機系に転送したリソースは削除されます。

このとき，実行系はロールバック処理を行い，ERRTRN3 を起動して UAP に制御を戻します。

7.7　システム間通信でのメッセージ転送処理

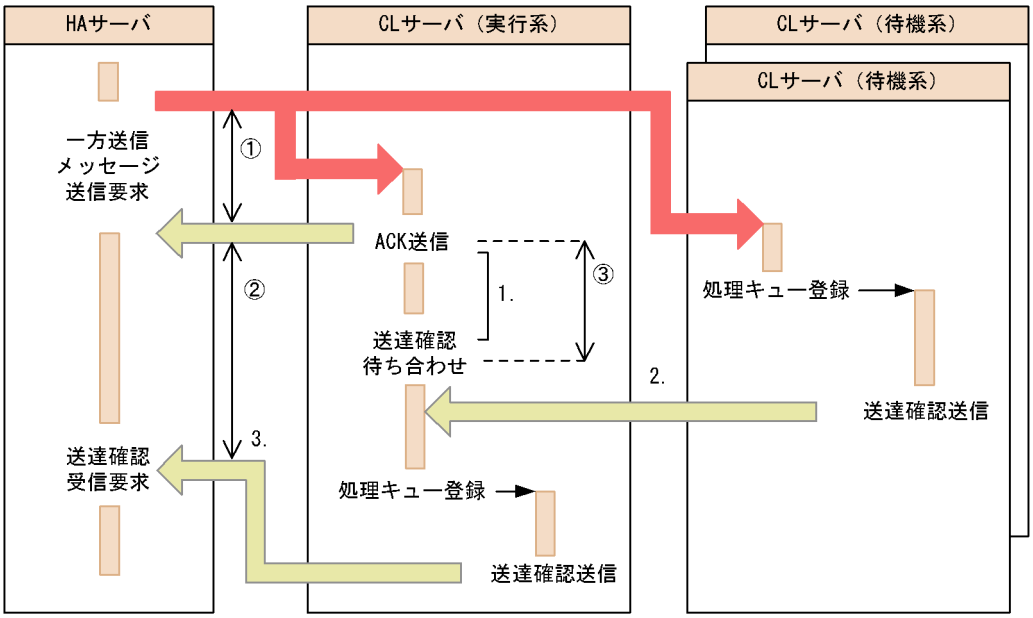
CL サーバでは，UDP 通信機能を使用してメッセージのシステム間通信を行っています。

ここでは，UDP 通信機能を使用したシステム間通信の処理の流れについて，メッセージの種類ごとに説明します。

7.7.1　一方送信メッセージの処理の流れ

一方送信メッセージの処理の流れを次の図に示します。

図 7-10　一方送信メッセージの処理の流れ



（凡例）

：ユーザメッセージの流れ

：制御メッセージの流れ

：制御の流れ

：タイマ監視範囲

ACK　：肯定応答

①　：eeudpdef定義コマンドまたはmyudpsnddef定義コマンドの
-wオプションまたは-Wオプションの指定値

②　：高速メッセージ送信関連定義のmch_send_ack_timerオペランドの指定値

③　：高速メッセージ送信関連定義のmch_clsend_ack_timerオペランドの指定値

説明

1. 実行系は，一方送信メッセージに対する肯定応答（ACK）を送信したあと，待機系からの送達確認メッセージの待ち合わせを行います。
2. 待機系は，一方送信メッセージを受信したあと，実行系に送達確認メッセージを肯定応答不要で送信します。
3. 実行系は，待機系から送達確認メッセージを受信したあと，HA サーバに送達確認メッセージを肯定応答不要で送信します。

参考

- ここでは，入力メッセージの送信元として HA サーバを仮定しています。CL サーバから別のクラスタグループの CL サーバへマルチキャストでメッセージを送信することもできます。その場合，HA サーバが CL サーバになります。
- HA サーバは，マルチキャストによって CL サーバの各サーバにメッセージを送信し，送達確認メッセージを CL サーバの実行系から受け取ります。このように，HA サーバは，CL サーバの各サーバにメッセージを送信しているため，どのサーバが実行系であるかを意識する必要がありません。

7.7.2 CL 同期メッセージの処理の流れ

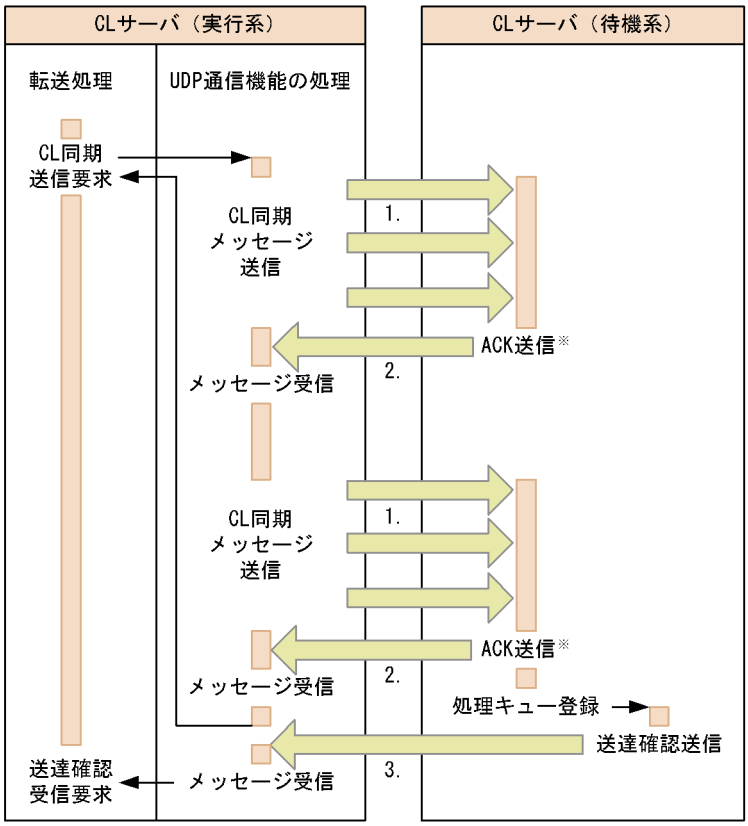
通常，CL サーバの待機系では，実行系からの CL 同期メッセージを受信した場合，処理キューを経由して CL 同期メッセージを受信したスレッドとは別のスレッドで実行系へ送達確認メッセージを送信します。

mch_clreq_rcvthd オペランドに Y を指定した場合は，CL 同期メッセージ受信完了時，処理キューを経由しないで，実行系へ送達確認メッセージを送信するため，実行系での同期処理時間を短縮できます。

ただし，実行系から転送する処理スレッドが多い場合や，転送するデータサイズが大きい場合は実行系での同期処理時間が短縮できない場合もあります。

mch_clreq_rcvthd オペランドに N を指定したときの CL 同期メッセージの処理の流れを次の図に示します。

図 7-11　CL 同期メッセージの処理の流れ



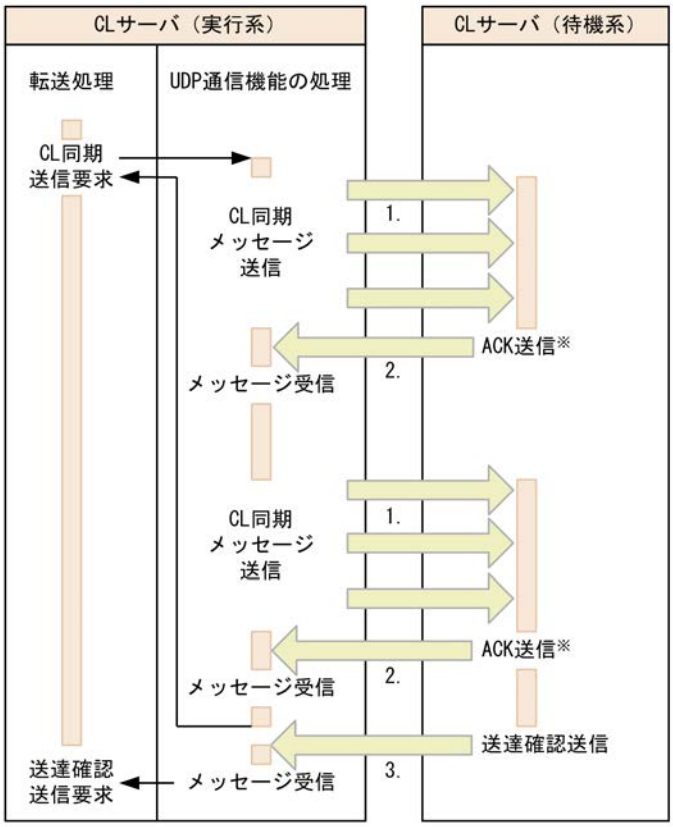
注※
ACK送信の有無は，転送されたデータサイズに応じて決定されます。
詳細については，RPC関連定義のrpc_udp_lmsg_deliverychk_sizeオペランドの
説明を参照してください。

説明

1. 実行系は，リソースを一括して待機系に送信します。
2. 待機系は，送信されたデータサイズに応じて肯定応答（ACK）を送信します。
3. 待機系は，すべての CL 同期メッセージを受信したあと，実行系に送達確認メッセージを肯定応答不要で送信します。

mch_clreq_rcvthd オペランドに Y を指定したときの CL 同期メッセージの処理の流れを次の図に示します。

図 7-12　CL 同期メッセージの処理の流れ（mch_clreq_rcvthd オペランドに Y を指定したとき）



（凡例）

：制御メッセージの流れ

：制御の流れ

ACK：肯定応答

注※
ACK送信の有無は，転送されたデータサイズに応じて決定されます。
詳細については，RPC関連定義のrpc_udp_lmsg_deliverychk_sizeオペランドの説明を参照してください。

説明

mch_clreq_rcvthd オペランドに Y を指定したときは，CL 同期メッセージの肯定応答（ACK）送信後，処理キューを経由せずに同一スレッドから送達確認メッセージを送信します。

なお，送達確認メッセージの送信タイミング以外は，通常時の処理の流れと同じです。

7.8　転送処理での時間監視機能

ここでは，転送処理での時間監視機能について説明します。転送処理での時間監視機能では，次に示すことを監視できます。

- 送信メッセージの監視
- 待機系で仕掛かり中のリソースの監視

7.8.1　送信メッセージの監視

実行系の XTC では，メッセージ送信処理が正常に行われたかどうかを，次に示す二つの監視時間によって判断します。

- CL 同期待ち監視時間
- CL 同期後続待ち監視時間

これらの時間監視でタイムアウトが発生した場合，XTC はメッセージ送信処理が正常に行われなかったと判断し，エラー処理を行います。タイムアウト発生時の処理については，「7.9　転送処理で障害が発生したときの XTC の処理」を参照してください。

(1) CL 同期待ち監視時間の設定

実行系の XTC では，CL 同期待ち監視時間として次に示す時間を監視します。

- 実行系が CL 同期メッセージを送信してから，待機系からの送達確認メッセージを最初に受信するまでの時間
- 実行系が一方送信メッセージを受信してから，すべての待機系からの送達確認メッセージを受信するまでの時間
- 実行系が待機系からの送達確認メッセージを受信してから，一方送信メッセージを受信するまでの時間

CL 同期待ち監視時間は，クラスタ連携関連定義の mch_clsend_ack_timer オペランドで指定します。

(2) CL 同期後続待ち監視時間の設定

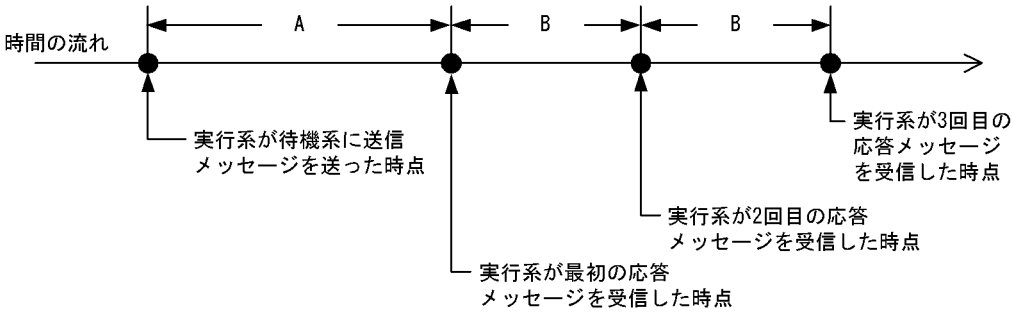
実行系の XTC では，CL 同期後続待ち監視時間として次に示す時間を監視します。

- 実行系が待機系からの送達確認メッセージを受信してから，後続の送達確認メッセージを受信するまでの時間

CL 同期後続待ち監視時間は，クラスタ連携関連定義の mch_clsend_next_timer オペランドで指定します。

CL 同期待ち監視時間と CL 同期後続待ち監視時間の関係を次の図に示します。

図 7-13　CL 同期待ち監視時間と CL 同期後続待ち監視時間の関係



（凡例）　A：CL同期待ち監視時間
　　B：CL同期後続待ち監視時間

7.8.2　待機系で仕掛かり中のリソースの監視

待機系の XTC では，待機系で仕掛かり中のリソースがあるかどうかを一定間隔でチェックしています。仕掛かり中の時間が監視時間に達した場合，仕掛かり中のリソースに関する KFSB88300-I メッセージを出力します。

監視時間は，クラスタ連携関連定義の mch_commit_wait_msg_interval オペランドで指定します。また，監視対象とする範囲を監視レベルとして，mch_commit_wait_msg_level オペランドで指定します。監視レベルには次に示す二つの種類があります。

- **監視レベル** 1
  トランザクション決着後の CL 同期メッセージの受信から，CL 同期済み通知メッセージの受信による処理キュー登録までの時間を監視対象とします。
- **監視レベル** 2
  CL 同期済み通知メッセージの受信から，CL 同期済み通知メッセージの受信による処理キュー登録までの時間を監視対象とします。

7.9　転送処理で障害が発生したときの XTC の処理

転送処理で障害が発生したときの XTC の処理を次の表に示します。

表 7-3　転送処理で障害が発生したときの XTC の処理

項番	障害要因	XTC の処理
1	HA サーバからの一方送信メッセージを受信したあとに，待機系からの送達確認メッセージの受信待ちタイムアウトが発生した	• 送信元に送達確認エラーメッセージ（CL 同期エラーメッセージ）を送信します。 • 受信メッセージを破棄します。 • 待機系に受信メッセージを破棄するメッセージを送信します。
2	待機系から送達確認メッセージを受信したあとに，HA サーバからの一方送信メッセージの受信待ちタイムアウトが発生した	• HA サーバに送達確認エラーメッセージ（CL 同期エラーメッセージ）を送信します。 • 待機系に受信メッセージを破棄するメッセージを送信します。
3	HA サーバへの送達確認メッセージの送信エラー（リトライ失敗）が発生した	処理キューの登録後，送信エラーのメッセージを出力します。HA サーバに送達確認メッセージが届いている場合があるため，受信メッセージは破棄しません。
4	待機系への CL 同期メッセージの送信エラー（リトライ失敗）が発生した	• 送信エラーのメッセージを出力します。 • 実行系孤立状態になったときの XTC の処理については，「7.4　実行系孤立状態になった場合の対処」を参照してください。
5	待機系への CL 同期済み通知メッセージの送信エラー（リトライ失敗）が発生した	

参考

ここでは，入力メッセージの送信元として HA サーバを仮定しています。CL サーバから別のクラスタグループの CL サーバへマルチキャストでメッセージを送信することもできます。

転送処理で障害が発生したときの XTC の処理をケース別に説明します。

(1) 一方送信メッセージが実行系に届かない場合

ネットワーク障害などによって，HA サーバからの一方送信メッセージが実行系に届かないときの処理を次に示します。

■実行系の処理

- 待機系からの送達確認メッセージを受信したあと，CL 同期待ち監視時間のタイムアウトを検知し，HA サーバに送達確認エラーメッセージ（CL 同期エラーメッセージ）を送信します。

- 受信メッセージを破棄するメッセージをマルチキャストで待機系に送信します。

■待機系の処理

- HA サーバからの一方送信メッセージを受信したあと，実行系に送達確認メッセージを送信します。
- 実行系からの，受信メッセージを破棄するメッセージを受信したあと，HA サーバからの受信メッセージを破棄します。

■ HA サーバの処理

実行系が送信した送達確認エラーメッセージを受信します。送達確認エラーメッセージを受信したときの HA サーバの処理については，「2.2.8　障害処理」を参照してください。

（2）一方送信メッセージが待機系に届かない場合

ネットワーク障害などによって，HA サーバからの一方送信メッセージがすべての待機系に届かないときの処理を次に示します。

■実行系の処理

- HA サーバからの一方送信メッセージが待機系に届かなかったことを，クラスタ連携関連定義の mch_clsend_ack_timer オペランドの時間監視によるタイムアウトで検知し，HA サーバに送達確認エラーメッセージ（CL 同期エラーメッセージ）を送信します。
- HA サーバからの受信メッセージを破棄します。
- 受信メッセージを破棄するメッセージをマルチキャストで待機系に送信します。

■待機系の処理

実行系からの，受信メッセージを破棄するメッセージを受信したあと，HA サーバからの受信メッセージを破棄します。
なお，HA サーバからの受信メッセージを破棄するのは，HA サーバからの一方送信メッセージを受信した待機系だけです。

■ HA サーバの処理

実行系が送信した送達確認エラーメッセージを受信します。送達確認エラーメッセージを受信したときの HA サーバの処理については，「2.2　高速メッセージ送信制御」を参照してください。

（3）一方送信メッセージに対する送達確認メッセージが HA サーバに届かない場合

一方送信メッセージに対する送達確認メッセージが，ネットワーク障害などによって HA サーバに届かないときの処理を次に示します。

■実行系の処理

- 障害によって送達確認メッセージが HA サーバに送信できない場合，高速メッセージ送信関連定義の mch_send_retry_count オペランドの指定に従って送信リ

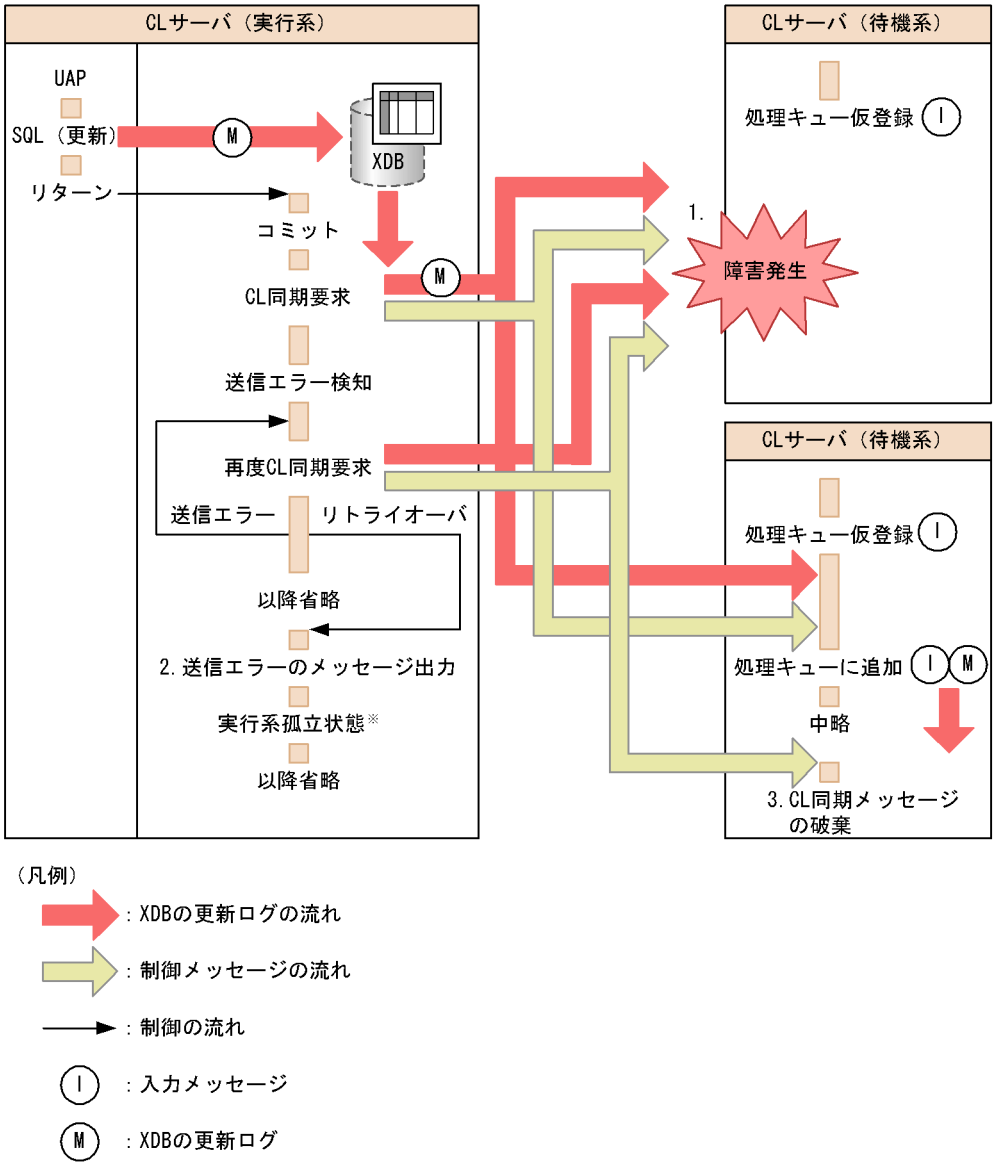
トライ処理を行います。

- 送信リトライ処理に失敗した場合，送信エラーのメッセージを出力します。

（4） CL 同期メッセージまたは CL 同期済み通知メッセージの送信に失敗した場合

CL 同期メッセージまたは CL 同期済み通知メッセージの送信がエラーとなった場合，クラスタ連携関連定義の mch_clsend_retry_count オペランドの指定に従ってメッセージを再送します。すべての待機系への再送が失敗した場合は，待機系を切り離して実行系孤立状態となります。CL 同期メッセージまたは CL 同期済み通知メッセージの送信に失敗した場合の処理の流れを次の図に示します。

図 7-14　CL 同期メッセージまたは CL 同期済み通知メッセージの送信に失敗した場合の処理の流れ



注※　すべての待機系を切り離した場合に，実行系孤立状態になります。

説明

1. 実行系からの CL 同期メッセージの送信に失敗した場合，クラスタ連携関連定義の mch_clsend_retry_count オペランドの指定に従ってメッセージの送信リトライ処理を行います。
2. 送信リトライ処理に失敗した場合，送信エラーのメッセージを出力し，該当する待機系を切り離して処理を続行します。ただし，すべての待機系を切り離した場

合は，実行系孤立状態となります。

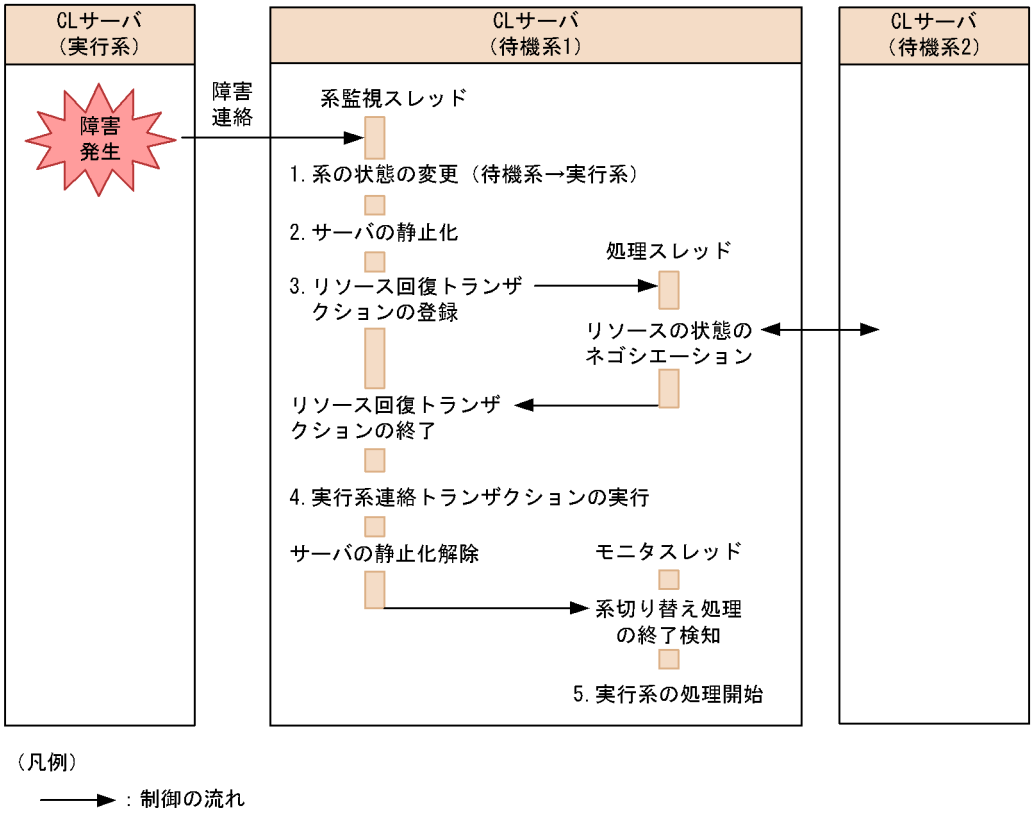
3. 再送された CL 同期メッセージが待機系で受信済みの場合は，受信した CL 同期メッセージを破棄します。

7.10 系切り替えが発生したときの XTC の処理

系切り替えが発生したタイミングによっては，仕掛かり中のリソースの状態が，二つの待機系の間で異なることがあります。この場合，XTC では，リソースの状態を一致させてから，系切り替え後の実行系で業務処理を開始します。これら一連の処理（リソースの状態を一致させる一連の処理）を**キューログ回復処理**といいます。

系切り替えが発生したときのキューログ回復処理を次の図に示します。

図 7-15 系切り替えが発生したときのキューログ回復処理



説明

1. HA モニタからの障害連絡を待機系の系監視スレッドが受け付けると，系の状態が待機系から実行系に変わります。
2. リソースの状態を一致させる前に，リソースの状態の更新を中断します。また，外部からのメッセージの受信を抑止したあと，実行中のすべての処理が終了するのを待ち合わせます。この処理を**サーバの静止化**といいます。
   サーバの静止化中は受信メッセージを破棄しますが，メッセージ送信元が静止化を解除するまで送信リトライするため，静止化を解除したあとにメッセージを受け取ることができます。

3. リソース回復トランザクションを登録します。このトランザクションによって，待機系（図中の待機系 2）とリソースの状態を一致させます。このリソースの状態を合わせる処理を**ネゴシエーション**といいます。
   なお，待機系がない場合は，この処理は行われません。
4. リソース回復トランザクションの終了後，系監視スレッドは実行系連絡トランザクション（UI）を実行し，サーバの静止化を解除します。
5. すべての系切り替え処理が終了したあとにメッセージの受信抑止を解除します。

7.11 CL サーバで実行できる HA モニタのコマンド

CL サーバでは，実行できる HA モニタのコマンドが限定されています。CL サーバで実行できる HA モニタのコマンドを次の表に示します。

表 7-4 CL サーバで実行できる HA モニタのコマンド

項番	HA モニタのコマンド	コマンドの説明
1	monact	待ち状態のサーバを実行系のサーバとして起動します。
2	monchange	HA モニタやサーバの稼働中に設定を変更します。
3	moncheck	定義チェックを実施します。
4	mondeact	待ち状態のサーバを停止します。
5	monlink	HA モニタとほかの HA モニタを接続します。
6	monmp	MP の状態を表示します。
7	monodrshw	サーバの切り替え順序制御の状態を表示します。
8	monpath	監視パスの状態を表示します。
9	monsetup	HA モニタの環境設定をします。
10	monshow	サーバと系の状態を表示します。
11	monstart	HA モニタを起動します。
12	monstop	HA モニタを停止します。
13	monswap	計画系切り替えをします。
14	monts	HA モニタのトラブルシュート情報を収集します。

各コマンドの詳細については，マニュアル「高信頼化システム監視機能 HA モニタ Linux(R) 編」を参照してください。

8 障害時の運用

この章では，障害時の運用について説明します。

8.1 想定される障害

ここでは，TP1 キャッシュ機能使用時の運用で想定される障害と対処方法について説明します。

8.1.1 XTC の開始時に発生する障害

CL サーバでの XTC の開始時に発生する障害について説明します。

(1) 実行系の開始処理中での障害

実行系の開始処理中に障害が発生した場合，系切り替えは発生しません。そのため，実行系の開始処理中に障害が発生した場合に待機系が起動していると，クラスタグループ内は待機系だけとなります。この場合，ユーザの運用によって待機系をすべて終了させる必要があります。クラスタグループ内で待機系だけが起動している状態では，実行系を起動できません。

8.1.2 XTC の終了時に発生する障害

CL サーバでの XTC の終了時に発生する障害について説明します。

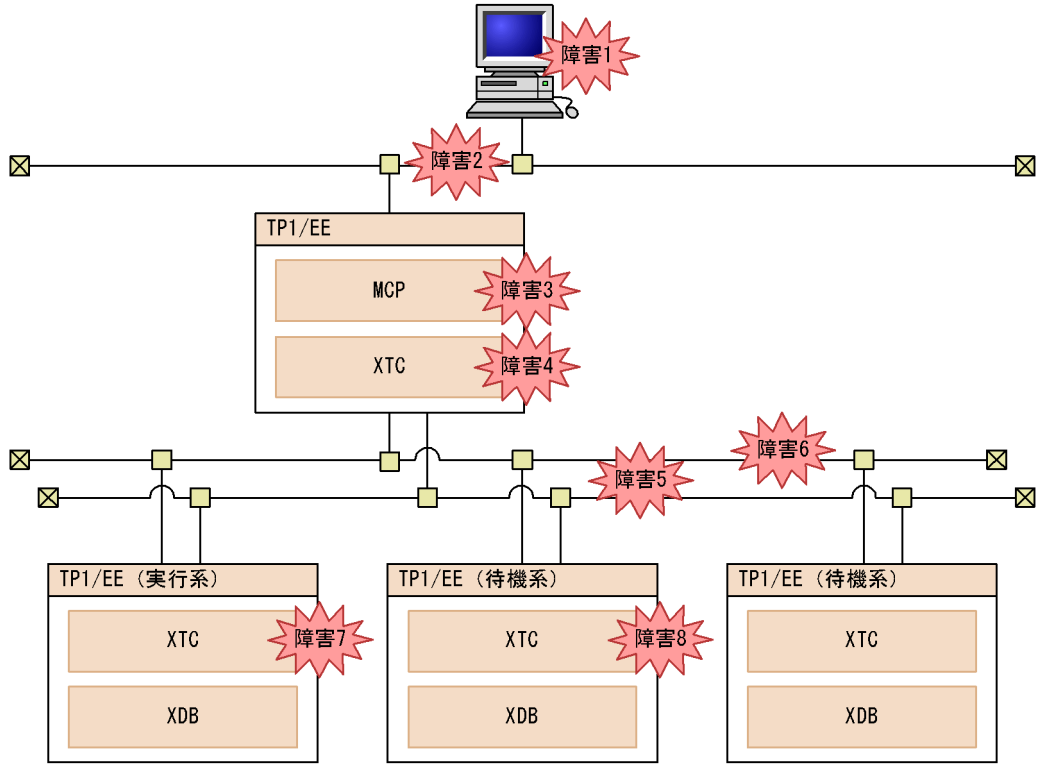
(1) 実行系の終了検知エラー

実行系の終了時，何らかの障害によって待機系で実行系の終了が検知できなかった場合，CL サーバ内は待機系だけとなります。このような場合，ユーザの運用によって CL サーバ内に残っている待機系を終了させる必要があります。待機系を終了させないで実行系を起動することはできません。

8.1.3 メッセージの送受信で発生する障害

メッセージの送受信で発生する障害を次の図に示します。

図 8-1　メッセージの送受信で発生する障害



説明

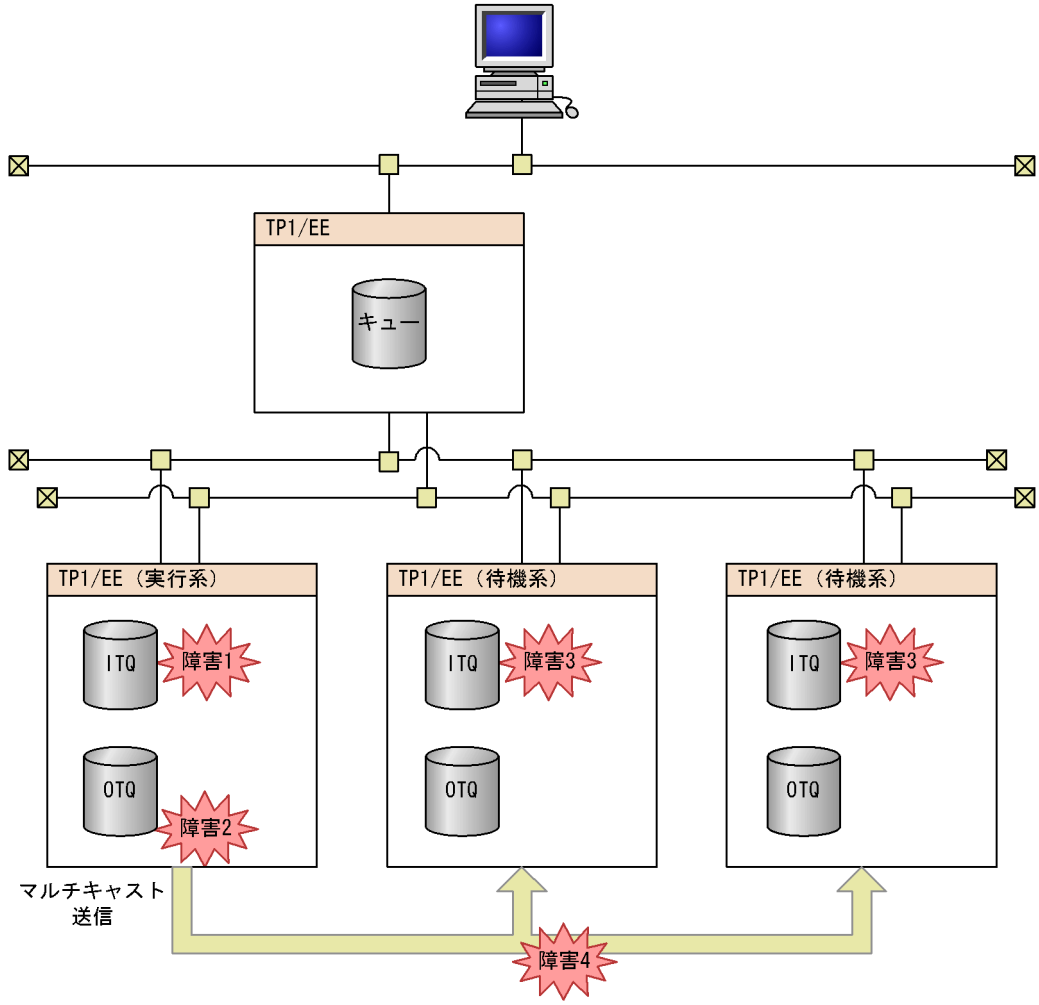
障害	障害内容	障害時の動作	システムまたはユーザの対応	ユーザによる対処
1	端末障害	MCP ではメッセージが受信できません。端末が起動できない場合は，送信メッセージを破棄します。	MCP では，リトライによる接続を試みます。リトライ失敗時はメッセージを破棄します。	出力されたメッセージを基に，原因を取り除く必要があります。
2	端末との LAN 障害	• メッセージの動作は障害 1 と同様です。 • TCP/IP による通信の場合，通信障害が発生して，通信回線上でメッセージが消失するおそれがあります。		
3	MCP の障害（HA サーバ）	プロセスダウンになります（UOC を含みます）。	障害原因を取り除く必要があります。	

障害	障害内容	障害時の動作	システムまたはユーザの対応	ユーザによる対処
4	XTC の障害（HA サーバ系）	プロセスダウンになります。		
5	LAN 障害（1LAN）	W-send 機能によって，処理を継続できます。業務には影響ありません。	XTC での対応はありません。LAN の起動が必要です。	
6	LAN 障害（2LAN）	• 転送ができなくなり，エラーが発生します。 • 出力キュー（OTQ）を閉塞します。	XTC では，リトライによる再送を試みます。リトライ失敗時はメッセージを破棄します。	
7	XTC の障害（CL サーバの実行系）	待機系が起動している場合は，系切り替えが発生します。	障害原因を取り除く必要があります。	ユーザによる対処がなくても，オンラインを継続できます。 障害のあった実行系については，必要に応じて障害対策を実施してください。
8	XTC の障害（CL サーバの待機系）	待機系が停止します。メモリの内容は消失します。		出力されたメッセージを基に，原因を取り除く必要があります。

8.1.4 キュー満杯時に発生する障害

キュー満杯時に発生する障害を次の図に示します。

図 8-2　キュー満杯時に発生する障害



（凡例）
ITQ：入力キュー
OTQ：出力キュー

説明

障害	障害内容	障害時の動作	システムまたはユーザの対応	ユーザによる対処
1	処理キュー満杯（実行系）	リトライ処理に失敗すると，送信元はエラーとなります。	XTC での対応はありません。	出力されたメッセージを基に，原因を取り除く必要があります。

障害	障害内容	障害時の動作	システムまたはユーザの対応	ユーザによる対処
2	マルチキャストメッセージ送信時の出力キュー（OTQ）満杯			
3	処理キュー満杯（待機系）	実行系だけが受信していた場合には，待機系を切り離します。		
4	CL 同期の転送障害	リトライ処理に失敗すると，同期点でエラーが発生し，待機系を切り離します。	XTC では，リトライによる再送を試みます。リトライ失敗時はメッセージを破棄します。	

8.1.5 システムダウン時の動作

障害発生によって CL サーバのオンラインが 1 台となった場合，1 台構成となったことを検知した時点でオンラインが停止します。この場合，キューダンプファイルおよびメモリダンプファイルを取得します。

詳細については，「7. CL 連携による系切り替え機能」を参照してください。

8.1.6 UAP の障害（SQL での問題）

UAP に障害が発生した場合，トラブルシュート情報を利用して障害要因を調査します。

TP1 キャッシュ機能使用時は，TP1/EE のトラブルシュート情報に加えて，XDB による XDB トレース情報を取得できます。XDB トレース情報によって，UAP の障害が SQL での問題だった場合に対処できます。詳細については，マニュアル「TP1/EE/Extended Data Cache 使用の手引」を参照してください。

8.2　障害に備えて取得する情報

障害に備えて，TASKTM ファイルや回線トレースファイルなどを TP1/EE で取得します。障害時に取得する情報の詳細については，マニュアル「TP1/Server Base Enterprise Option 使用の手引」を参照してください。

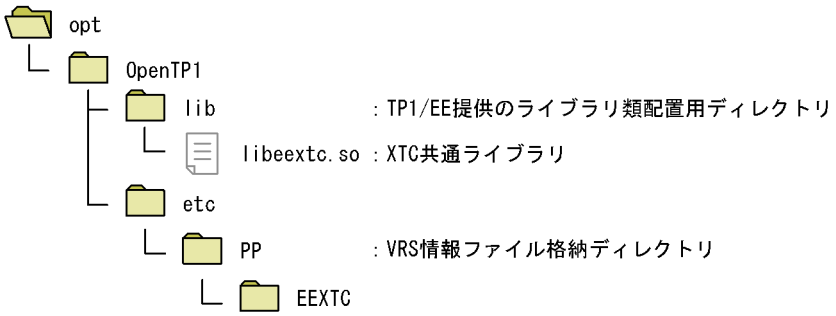
ただし，一部のファイルでは容量見積もりの方法が異なります。「3.　XTC が扱うファイルの種類と容量見積もり」もあわせて参照してください。

付録

付録 A　インストールディレクトリとファイル

XTC のインストールディレクトリの構成とファイルを次の図に示します。

図 A-1　XTC のインストールディレクトリの構成とファイル



付録 B　各バージョンの変更内容

各バージョンの変更内容を示します。

(1) 変更内容（3000-3-F54-10）

次の製品の変更内容（3000-3-F54-10）を表に示します。

- uCosminexus TP1/EE/Extended Transaction Controller　01-02

追加・変更内容
タイマ監視範囲について，説明を追加した。
一方送信メッセージを一括送信できるようにした。 これに伴い，次のオペランドを追加した。 高速メッセージ送信関連定義 　mch_send_lump_count また，次のオペランドの説明を変更した。 RPC 関連定義 　rpc_udp_lmsg_deliverychk_size
RPC メッセージ，およびタイマメッセージを，ee_scd_msg_receive 関数で受信できるようにした。
RPC メッセージ，および MCP メッセージでも，永続化できるようにした。 これに伴い，次のオペランドを追加した。 RPC 関連定義 　rpc_recv_permanence 次のオペランドを CL サーバで指定できるようにした。 プロセス関連定義 　mcp_use トランザクション関連定義 　trn_expiration_time_rl また，次のコマンドの説明を変更した。 • eetrbqueed
XTC を配置した実行系 1 台で，TP1 キャッシュ機能を実現できる，CL 単独起動機能をサポートした。 これに伴い，次の定義の説明を変更した。 クラスタ連携関連定義 　cluster_test_mode 　standby_start_watch_time UDP グループ情報関連定義（コマンド形式） 　myudprcvdef 定義コマンド 　clgrpdef 定義コマンド また，次のコマンドの説明を変更した。 • eehamls
回線トレース情報の取得量を変更できるようにした。 これに伴い，次のオペランドの指定値を追加した。 RPC 関連定義 　rpc_udp_linetrace
次のオペランドの説明に，永続指定の一方送信メッセージの送信先に CL サーバが含まれる場合の説明を追加した。 高速メッセージ送信関連定義 　mch_send_ack_timer

追加・変更内容
CL サーバで実行できる HA モニタのコマンドから，monhelp コマンドを削除した。また，monchange コマンドを追加した。

付録 C　用語解説

このマニュアルで使用している用語について説明します。

（英字）

CL サーバ

メッセージや RPC の受信側となって，高速データ処理を行うサーバです。実行系 1 台，待機系 2 台の 3 台で構成します。
CL 単独起動機能を使用する場合は，実行系 1 台だけで構成することもできます。

CL サーバのシミュレート機能

UAP の動作確認テストを支援する機能です。この機能を使用すると，待機系の動作をシステムがシミュレートして稼働するため，実行系 1 台だけで UAP の動作確認テストができます。
HA モニタおよび CL サーバの待機系の環境設定が必要ないため，テスト環境の構築が容易になります。

CL 同期

CL サーバ内の各サーバのリソースの状態を一致させることです。

CL 連携

XTC と HA モニタの連携によって，高信頼性と高可用性を実現するクラスタリングのことです。

HA サーバ

メッセージや RPC の送信側となるサーバです。実行系 1 台，待機系 1 台の 2 台で構成します。

HA モニタ情報ファイル

XTC が HA モニタに接続したときに作成されるファイルです。XTC と HA モニタの間で使用されます。

TP1 キャッシュ機能

インメモリデータ処理技術を中心とした高速トランザクション処理機能の総称です。TP1 キャッシュ機能を使用することで，トランザクション処理の高速化と，高信頼性，高可用性を同時に実現するオンライントランザクション処理システムを構築できます。

W-send 機能

ネットワーク経路を多重化した構成で hbonding ドライバを適用し，二つの経路から同一のパケットを送受信する機能です。W-send 機能を使用するためには，hbonding ドライバのバージョンが 02-01 以降であり，hbonding ドライバの動作モードに broadcast モードを選択する必要があります。

XDB 用ワーク領域（XDBPOOL）

XDB が利用するワーク領域です。

XTC 用ワーク領域（XTCPOOL）
XTC が利用するワーク領域です。

（ア行）

永続メッセージ
CL サーバに送信するメッセージのうち，待機系で保証するメッセージです。

（カ行）

キューダンプ機能
オンライン終了時にキューに滞留しているメッセージのファイル出力および編集をする機能です。

キューダンプファイル
キューダンプを取得するファイルです。

クラスタグループ
CL 連携を行うサーバのグループのことです。クラスタグループは，UDP グループ情報関連定義の clgrpdef 定義コマンドで定義します。

計画系切り替え
HA モニタの monswap コマンドを実行して系を切り替えることです。

更新ログ
実行系の XDB のデータベースの更新情報です。

高速メッセージ送信制御
TP1 キャッシュ機能使用時の，UDP プロトコルによるシステム間通信です。

孤立終了モード
実行系孤立状態になったときに管理者が選択する XTC の終了形態です。孤立終了モードには，孤立終了モード A と孤立終了モード B の二つがあります。

（サ行）

実行系孤立状態
待機系がなくなり，実行系だけとなった状態です。

自動系切り替え
実行系で障害が発生した場合に，自動的に系を切り替えることです。

ステータスファイルレス機能
ステータスファイルに取得するシステム制御情報を限定する機能です。

（タ行）

滞留メッセージ

オンライン終了時に，キューに滞留したメッセージのことです。

転送処理

実行系で更新されたリソースの情報を待機系に逐次転送して，実行系のリソース更新結果を待機系で保証する処理です。

（ハ行）

非永続メッセージ

CL サーバに送信するメッセージのうち，待機系で保証しないメッセージです。

（マ行）

マルチスタンバイ機能

一つの実行系に対して，複数の待機系を準備する機能です。マルチスタンバイ機能は HA モニタの機能です。

（ラ行）

連動系切り替え

複数のクラスタグループをグループ化しておき，障害が発生した場合，グループ内の全クラスタグループを待機系に切り替える機能です。

索引

C

E

H

M

R

S

T

U

W

X

い

う

え

か

き

く

け

こ

さ

し

す

そ

た

つ

て

と

に

ね

は

ひ

ふ

ほ

ま

め

ゆ

よ

り

れ

ろ